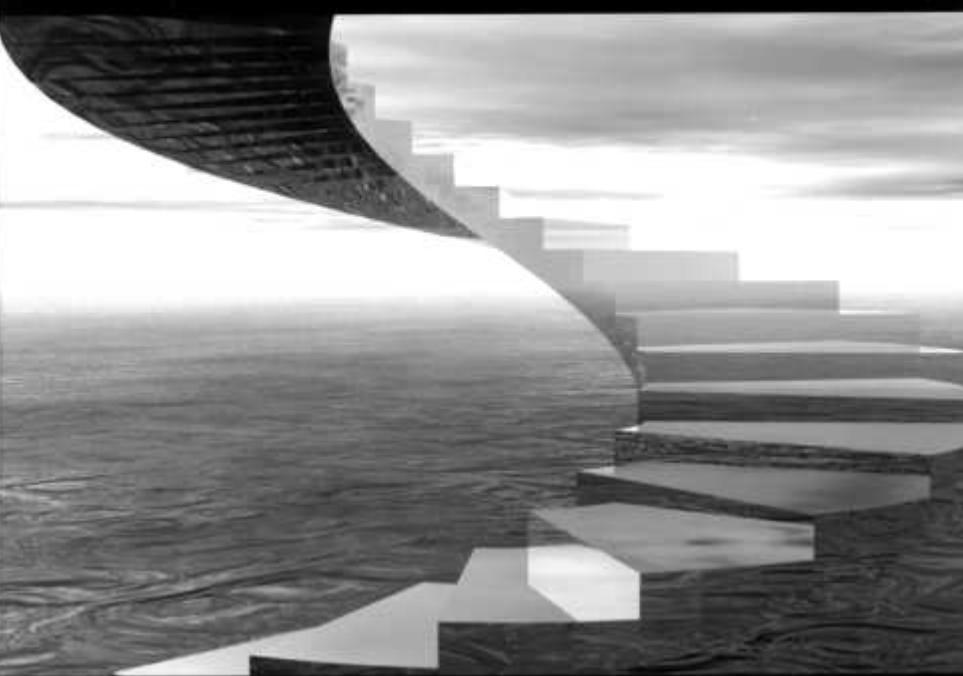


Virtual Reality Technology

Grigore C. Burdea, Philippe Coiffet

GRIGORE C. BURDEA • PHILIPPE COIFFET



Virtual Reality TECHNOLOGY

SECOND EDITION



VIRTUAL REALITY TECHNOLOGY

VIRTUAL REALITY TECHNOLOGY

Second Edition

GRIGORE C. BURDEA

Rutgers-The State University of New Jersey

PHILIPPE COIFFET

University of Versailles, France



A John Wiley & Sons, Inc., Publication

Copyright © 2003 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400, fax 978-750-4470, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, e-mail permreq@wiley.com.

Limit of Liability/Disclaimer of Warranty While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993 or fax 317-572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print, however, may not be available in electronic format.

Library of Congress Cataloging-in-Publication Data

Burdea, Grigore.

[Réalité virtuelle. English]

Virtual reality technology / Grigore Burdea, Philippe Coiffet.—2nd ed.

p. cm.

“A Wiley-Interscience publication.”

Includes bibliographical references and index.

ISBN 0-471-36089-9 (cloth)

1. Human-computer interaction. 2. Virtual reality. I. Coiffet, Philippe. II. Title.

QA76.9.H85 B8713 2003
006—dc21

2002038088

Printed in the United States of America

10 9 8 7 6 5 4 3 2

**¶Due to the nature of digital conversion, some of the
images included in this e-book may lack the detail
and clarity of the originals.**

To my friends in need, Iris and Marty Hoffert Grigore Burdea

To my wife, Danielle Gadelin Philippe Coiffet

CONTENTS

FOREWORD xiii

PREFACE xv

1 INTRODUCTION 1

1.1 The Three I's of Virtual Reality/ 3

1.2 A Short History of Early Virtual Reality/ 3

1.3 Early Commercial VR Technology/ 8

1.4 VR Becomes an Industry/ 10

1.5 The Five Classic Components of a VR System/ 12

1.6 Review Questions / 13

References / 14

2 INPUT DEVICES: TRACKERS, NAVIGATION, AND GESTURE
INTERFACES 16

2.1 Three-Dimensional Position Trackers / 17

2.1.1 Tracker Performance Parameters / 19

2.1.2 Mechanical Trackers / 21

2.1.3 Magnetic Trackers / 24

2.1.4 Ultrasonic Trackers / 32

2.1.5 Optical Trackers / 35

2.1.6 Hybrid Inertial Trackers / 38

2.2 Navigation and Manipulation Interfaces / 41

2.2.1 Tracker-Based Navigation/Manipulation Interfaces / 42

2.2.2 Trackballs / 44

2.2.3 Three-Dimensional Probes / 45

2.3 Gesture Interfaces / 46

2.3.1 The Pinch Glove / 48

2.3.2 The 5DT Data Glove / 49

2.3.3 The Didjiglove / 51

2.3.4 The CyberGlove / 53

2.4 Conclusion / 54

2.5 Review Questions / 54

References / 54

3 OUTPUT DEVICES: GRAPHICS, THREE-DIMENSIONAL SOUND,
AND HAPTIC DISPLAYS 57

3.1 Graphics Displays / 58

3.1.1 The Human Visual System / 58

3.1.2 Personal Graphics Displays / 60

3.1.3 Large-Volume Displays / 70

3.2 Sound Displays / 84

3.2.1 The Human Auditory System / 84

3.2.2 The Convolvotron / 88

3.2.3 Speaker-Based Three-Dimensional Sound / 90

3.3 Haptic Feedback / 92

3.3.1 The Human Haptic System / 93

3.3.2 Tactile Feedback Interfaces / 97

3.3.3 Force Feedback Interfaces / 102

3.4 Conclusion / 110

3.5 Review Questions / 110

References / 111

4 COMPUTING ARCHITECTURES FOR VR 116

4.1 The Rendering Pipeline / 117

4.1.1 The Graphics Rendering Pipeline / 117

4.1.2 The Haptics Rendering Pipeline / 125

4.2 PC Graphics Architecture / 126

4.2.1 PC Graphics Accelerators / 129

4.2.2 Graphics Benchmarks / 133

4.3 Workstation-Based Architectures / 135

4.3.1 The Sun Blade 1000 Architecture / 136

4.3.2 The SGI Infinite Reality Architecture / 137

4.4 Distributed VR Architectures / 139

4.4.1 Multipipeline Synchronization / 140

4.4.2 Colocated Rendering Pipelines / 143

4.4.3 Distributed Virtual Environments / 149

4.5 Conclusion / 153

4.6 Review Questions / 154

References / 155

5 MODELING 157

5.1 Geometric Modeling / 158

5.1.1 Virtual Object Shape / 158

5.1.2 Object Visual Appearance / 164

5.2 Kinematics Modeling / 172

5.2.1 Homogeneous Transformation Matrices / 172

5.2.2 Object Position / 172

5.2.3 Transformation Invariants / 175

5.2.4 Object Hierarchies / 176

5.2.5 Viewing the Three-Dimensional World / 178

5.3 Physical Modeling / 180

5.3.1 Collision Detection / 180

5.3.2 Surface Deformation / 183

5.3.3 Force Computation / 184

5.3.4 Force Smoothing and Mapping / 190

5.3.5 Haptic Texturing / 192

5.4 Behavior Modeling / 194

5.5 Model Management / 197

5.5.1 Level-of-Detail Management / 198

5.5.2 Cell Segmentation / 202

5.6 Conclusion / 205

5.7 Review Questions / 206

References / 206

6 VR PROGRAMMING 210

6.1 Toolkits and Scene Graphs / 210

6.2 WorldToolKit / 214

6.2.1 Model Geometry and Appearance / 214

6.2.2 The WTK Scene Graph / 215

6.2.3 Sensors and Action Functions / 218

6.2.4 WTK Networking / 220

6.3 Java 3D / 221

6.3.1 Model Geometry and Appearance / 222

<u>6.3.2 Java 3D Scene Graph /</u>	223
<u>6.3.3 Sensors and Behaviors /</u>	225
<u>6.3.4 Java 3D Networking /</u>	227
<u>6.3.5 WTK and Java 3D Performance Comparison /</u>	227
<u>6.4 General Haptics Open Software Toolkit /</u>	231
<u>6.4.1 GHOST Integration with the Graphics Pipeline /</u>	231
<u>6.4.2 The GHOST Haptics Scene Graph /</u>	232
<u>6.4.3 Collision Detection and Response /</u>	234
<u>6.4.4 Graphics and PHANToM Calibration /</u>	234
<u>6.5 PeopleShop /</u>	235
<u>6.5.1 DI-Guy Geometry and Path /</u>	236
<u>6.5.2 Sensors and Behaviors /</u>	237
<u>6.5.3 PeopleShop Networking /</u>	238
<u>6.6 Conclusion /</u>	239
<u>6.7 Review Questions /</u>	239
<u>References /</u>	240
<u>7 HUMAN FACTORS IN VR</u>	243
<u>7.1 Methodology and Terminology /</u>	244
<u>7.1.1 Data Collection and Analysis /</u>	246
<u>7.1.2 Usability Engineering Methodology /</u>	250

7.2 User Performance Studies / 253

7.2.1 Testbed Evaluation of Universal VR Tasks / 253

7.2.2 Influence of System Responsiveness on User Performance / 256

7.2.3 Influence of Feedback Multimodality / 260

7.3 VR Health and Safety Issues / 266

7.3.1 Direct Effects of VR Simulations on Users / 267

7.3.2 Cybersickness / 269

7.3.3 Adaptation and Aftereffects / 274

7.3.4 Guidelines for Proper VR Usage / 276

7.4 VR and Society / 277

7.4.1 Impact on Professional Life / 278

7.4.2 Impact on Private Life / 278

7.4.3 Impact on Public Life / 279

7.5 Conclusion / 280

7.6 Review Questions / 280

References / 282

8 TRADITIONAL VR APPLICATIONS 285

8.1 Medical Applications of VR / 287

8.1.1 Virtual Anatomy / 287

8.1.2 Triage and Diagnostic / 289

8.1.3 Surgery / 296

8.1.4 Rehabilitation / 304

8.2 Education, Arts, and Entertainment / 314

8.2.1 VR in Education / 314

8.2.2 VR and the Arts / 319

8.2.3 Entertainment Applications of VR / 324

8.3 Military VR Applications / 328

8.3.1 Army Use of VR / 328

8.3.2 VR Applications in the Navy / 334

8.3.3 Air Force Use of VR / 338

8.4 Conclusion / 342

8.5 Review Questions / 342

References / 343

9 EMERGING APPLICATIONS OF VR 349

9.1 VR Applications in Manufacturing / 349

9.1.1 Virtual Prototyping / 350

9.1.2 Other VR Applications in Manufacturing / 358

9.2 Applications of VR in Robotics / 362

9.2.1 Robot Programming / 363

9.2.2 Robot Teleoperation / 365

9.3 Information Visualization / 371

9.3.1 Oil Exploration and Well Management / 374

9.3.2 Volumetric Data Visualization / 376

9.4 Conclusion / 382

9.5 Review Questions / 382

References / 383

INDEX 387

FOREWORD

It is insight into human nature that is the key to the communicator's skill. For whereas the writer is concerned with what he puts into writings, the communicator is concerned with what the reader gets out of it.

-William Bernbach

advertising legend, (1911-1982)

Burdea and Coiffet speak of three I's that their book centers on: interaction, immersion, and imagination. But it is a fourth I, insight, that makes this work so unique. Through insight into readers' diverse mindsets, Virtual Reality Technology has the versatility to capture the interests of the enthusiast or hobbyist, while at the same time presenting scientific rigor to satisfy the eager student or skilled technologist. The neophyte will come away with the basics of what constitutes a virtual environment, while the veteran will gain insights into how to tackle challenging problems such as specifying tracker performance requirements, resolving luminosity concerns in tiled displays, and achieving multipipeline synchronization.

Virtual Reality (VR) offers the enticing premise of a rich sensory experience. For years, proponents have heralded the limitless potential of VR to surpass predecessor interactive technologies. Those who have tried to communicate its potential, however, have generally leveraged static text, which often reduces what is supposed to be a stimulating, multimodal encounter to a one-dimensional exercise that underwhelms at best. Burdea and Coiffet's innovative multimedia presentation, with its enticing videos, abundant illustrations, and VRML and Java 3D-based interactive laboratory activities, is a novel and fresh approach to the presentation of large volumes of technical information. Based on my own experiences in teaching and researching VR-related subject matter, I believe this format is highly appropriate for the treatment of VR issues, which often rely heavily on

visual and auditory modalities. Just as VR applications have expanded far beyond what was originally envisioned due to the creative use of this multifaceted technology, Burdea and Coiffet have expanded the possibilities for understanding the complex, intertwined issues associated with VR technology through their vibrant format.

VR applications have branched out into numerous domains, from education to medicine to the alluring world of entertainment. Each of these domains brings challenges to developers, who will be able to turn to this book for answers. Developers will be guided in which HMD, tracker and I/O devices to use, the type of architecture to support their integration, types of authoring tools and techniques, how to address human factors concerns (i.e., health and safety issues), and, in some regards, how much one can anticipate having to invest to bring their virtual world to life. Developers can also learn from the invaluable experiences of others, which have been well illustrated in the diverse compendium of applications.

Taken together, the chapters present systematic and comprehensive coverage of VR technology, from its requisite constituents to its administration woes. As Burdea and Coiffet likely intended, what the reader is certain to get out of this second edition is up-to-date knowledge garnered via an engaging and interactive learning experience. It is a work destined to be highly regarded by practitioners, hobbyists, and students alike.

KAY M. STANNEY, PH.D.

Orlando, Florida

February 2003

PREFACE

Almost ten years have past since we wrote the first edition of Virtual Reality Technology. Since then Virtual Reality has proven to be fast-changing, largely misunderstood (outside the technical community), and still awaiting full-spread application.

Computing technology, especially the Personal Computer, has seen dramatic improvements, and laid the foundation for worldwide VR development. Formal education in colleges and universities, aimed at preparing specialists capable of creating useful VR applications, has unfortunately lagged behind. This is due to the absence of good VR textbooks, as well as a lack of instructors with first-hand knowledge of the subject.

In writing this book, we were aware of the fast-changing technology of the field, and consequently, we updated the material presented in the first edition and included new developments in hardware, software, and applications. Haptics merits a particular mention, since it has now entered "mainstream" VR and is treated in an integrative manner in our book.

Our main motivation in writing a second edition was, however, to offer a complete, structured book that follows the rigors of texts written in related fields, such as Mathematics, Electrical Engineering, or Computer Science. Therefore Virtual Reality Technology, Second Edition includes definitions, mathematical formulae, and review questions. It also provides the reader with video clips, because we felt that seeing examples of actual simulations will allow the student to better understand the concepts. These video clips are found on the CD accompanying the book, and are cross-referenced in text with the special icon . A laboratory manual with six chapters, each intended for a different experiment and programming skill is also included on the CD. The Laboratory Manual uses VRML and Java 3D, since they are

freely available on the Web, without additional license expenses for schools and universities. The experiments are designed for PC-based VR stations incorporating sensing gloves, stereo glasses, 3-D trackers, and haptic joysticks, in order to give the student a true experience in VR programming. Each Laboratory Manual chapter has homework and programming assignments for undergraduate and graduate students.

The textbook is intended for use in a one-semester VR course and can also be used in "subspecialty" classes. Its first four chapters—"Introduction," "Input Devices," "Output Devices," and "Computer Architectures for VR"—can form the core of a hardware-intensive course targeted mainly for the engineer. The next three chapters—"Modeling," "VR Programming" and "Human Factors in VR"—are appropriate for an advanced course in human-machine interfaces, more familiar to the computer scientist and experimental psychologist. The last two chapters—"Traditional VR Applications" and "Emerging Applications of VR"—are of interest to programmers, application developers, and others, and thus generally applicable. An Instructor's Resource Page has been set up at www.vrtechnology.org. This site has additional teaching materials to be used in connection to this book.

We want to thank first ChanSu Lee who co-authored the laboratory manual. We are grateful to the hundreds of students in the School of Engineering at Rutgers who took the Virtual Reality course and its associated laboratory over the past few years. Their enthusiasm for the subject, timely feedback and willingness to test the unknown helped make this textbook a better one. Our thanks also go to the CAIP Center at Rutgers University and its system support staff for providing computing and printing resources, and fixing up glitches. We are grateful to the many companies, laboratories, and individuals who contributed materials for this book. Special thanks go to Professor Steve Feiner and the Computer Science Department at Columbia University where Grigore Burdea spent part of his writing sabbatical. Professor Kay Stanney took time off from her busy schedule at the University of Central Florida, and wrote the Foreword to this edition, for which we are grateful. Last, but not least, we want to thank

our families, for their support, encouragement, and willingness to sacrifice "quality time" so that this book transits from the virtual to the real.

GRIGORE C. BURDEA

PHILIPPE COIFFET

Piscataway, New Jersey

Versailles, France

March 2003

CHAPTER 1

INTRODUCTION

The scientific community has been working in the field of virtual reality (VR) for decades, having recognized it as a very powerful human-computer interface. A large number of publications, TV shows, and conferences have described virtual reality in various and (sometimes) inconsistent ways. This has led to confusion, even in the technical literature.

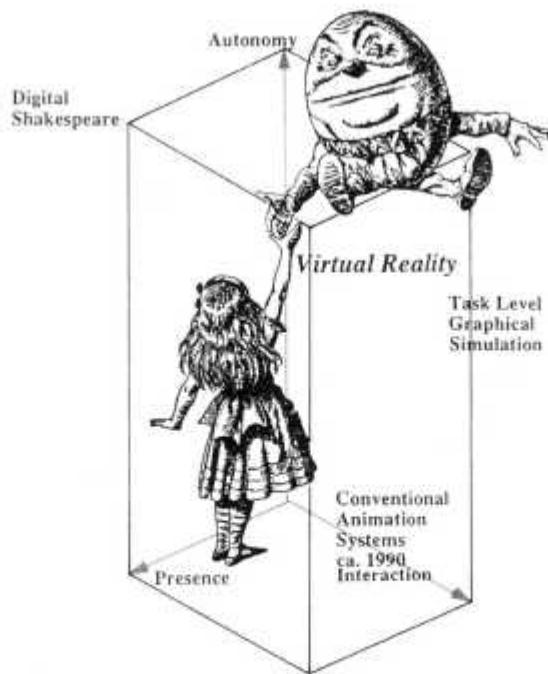
Before we attempt to define virtual reality, we should first say what it is not. Some researchers refer to telepresence, in which a user is immersed in a remote environment. This is very useful in telerobotics [Sheridan, 1992], where we attempt to control robots at a distance and where knowledge of what is happening around the robot is critical. Others have used the name enhanced reality or augmented reality (AR) [Muller, 1999], where certain computer graphics, or text, is overlaid on top of real images. A technician attempting to repair an installation may look at a photo in which overlaid graphics makes apparent otherwise occluded components [Bejczy, 1993]. Both telepresence and augmented reality incorporate images that are real, so neither is virtual reality in its strictest sense.

Technologists have been joined by artists and journalists in trying to define the field. The cover of the book *The World of Virtual Reality* published in Japan [Hattori, 1991] depicts Alice in the Wonderland, as shown in Figure 1.1. This is more eye-catching and amusing than scientific. Others have referred to virtual reality in terms of the devices it uses and not its purpose and function. The general public tends to associate virtual reality simulations with head-mounted displays (sometimes called "goggles") and sensing gloves, just because these were the first devices used in simulation. This is not a good definition either. Virtual reality today is done mostly without headmounted displays, by using large projection screens or desk-top PCs [Robertson et al., 1993]. Similarly, gloves can be replaced with much simpler trackballs or joysticks [Schmult and Jebens,

1993]. Conversely, sensing gloves can be used in other tasks than VR, such as in telerobotics [Clark et al., 1989]. Therefore describing virtual reality in terms of the devices it uses is also not an adequate definition.

HATTORI KATSURA
人工 現実感の 世界
服部 桂

What's Virtual Reality ?



工業調査会

Fig. 1.1 The cover of The World of Virtual Reality. From Hattori [1991]. Reprinted by permission.

Then what is virtual reality? Let us first describe it in terms of functionality. It is a simulation in which computer graphics is used to create a realistic-looking world. Moreover, the synthetic world is not static, but responds to the user's input (gesture, verbal command, etc.). This defines a key feature of virtual reality, which is real-time interactivity. Here real time means that the computer is able to detect a user's input and modify the virtual world instantaneously. People like to see things change on the screen in response to their commands and become captivated by the simulation. Anybody who doubts the spell-binding power of interactive graphics has only to look at children playing video games. It was reported that two youngsters in the United Kingdom continued to play Nintendo even though their house was on fire!

Interactivity and its captivating power contributes to the feeling of immersion, of being part of the action on the screen, that the user experiences. But virtual reality pushes this even further by using all human sensorial channels. Indeed, users not only see and manipulate graphic objects on the screen, they also touch and feel them [Burdea, 1996]. Researchers are also talking of the senses of smell and taste, although these sensorial modalities are less used at this time. In summary we give the following definition:

Definition Virtual reality is a high-end user-computer interface that involves realtime simulation and interactions through multiple sensorial channels. These sensorial modalities are visual, auditory, tactile, smell, and taste.

Virtual reality can also be described from the simulation content point of view as unifying realistic (or veridical [Codella et al., 1993]) realities with artificial reality. This is a synthetic environment, for which there is no real counterpart (or antecedent) [Krueger, 1991]. For the rest of this book we use the term virtual reality to encompass all the other terminology described earlier.

1.1 THE THREE I's OF VIRTUAL REALITY

It is clear from the foregoing description that virtual reality is both interactive and immersive. These features are the two I's that most people are familiar with. There is, however, a third feature of virtual reality that fewer people are aware of. Virtual reality is not just a medium or a high-end user interface, it also has applications that involve solutions to real problems in engineering, medicine, the military, etc. These applications are designed by virtual reality developers. The extent to which an application is able to solve a particular problem, that is, the extent to which a simulation performs well, depends therefore very much on the human imagination, the third "I" of VR. Virtual reality is therefore an integrated trio of immersioninteraction-imagination, as shown in Figure 1.2. The imagination part of VR refers also to the mind's capacity to perceive nonexistent things. The triangle in Figure 1.2, for example, is easily "seen" by the reader, yet it only exists in his or her imagination.

1.2 A SHORT HISTORY OF EARLY VIRTUAL REALITY

Virtual reality is not a new invention, but dates back more than 40 years. In 1962, U.S. Patent #3,050,870 was issued to Morton Heilig for his invention entitled Sensorama Simulator, which was the first virtual reality video arcade. As shown in Figure 1.3, this early virtual reality workstation had three-dimensional (3D) video feedback (obtained with a pair of side-by-side 35-mm cameras), motion, color, stereo sound, aromas, wind effects (using small fans placed near the user's head), and a seat that vibrated. It was thus possible to simulate a motorcycle ride through New York, where the "rider" sensed the wind and felt the pot-holes of the road as the seat vibrated. The rider could even smell food when passing by a store.

VIRTUAL REALITY TRIANGLE

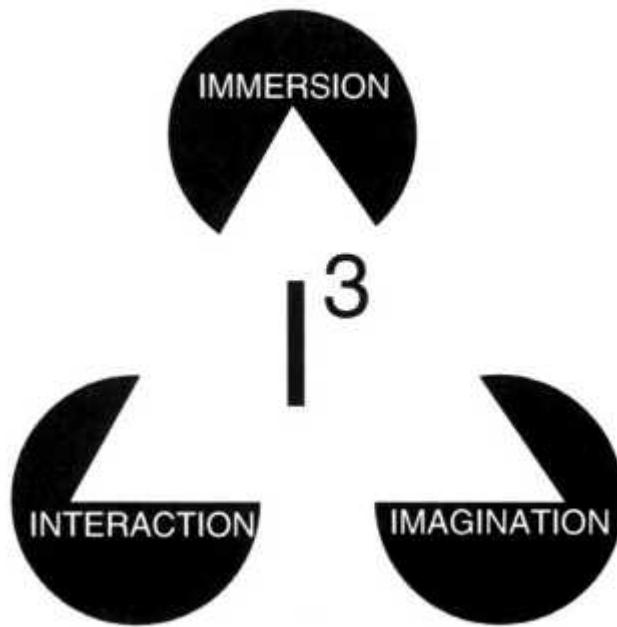


Fig. 1.2 The three I's of virtual reality, immersion-interaction-imagination.
Adapted from Burdea [1993]. Reprinted by permission.

Heilig also realized the possibilities of head-mounted television. He designed a simulation mask that included 3D slides with wide peripheral effects, focusing controls and optics, stereophonic sound, and the capability to include smell. A depiction from U.S. Patent #2,955,156 issued to him on October 4, 1960, is shown in Figure 1.4. Heilig, a cinematographer by profession, was well ahead of his time. Like Jules Verne, he imagined a new machine that would replace the classical cinematographic experience of today. He was also like Thomas Edison, an inventor who not only dreamed ideas, but also transformed them into real machines. At the time of Heilig's inventions, nobody realized the revolutionary technological progress they represented.

Heilig's initial work on head-mounted displays (HMD) was continued by Ivan Sutherland. In 1966 Sutherland used two cathode ray tubes (CRTs) mounted along the user's ears. Today's high-end HMDs use miniature CRTs mounted in the same configuration. Since the tubes available in 1966 were much heavier than those in use today, Sutherland had to rely on a

mechanical arm to support the weight of the display. This mechanical arm had potentiometers that measured the user's view direction. Most of today's HMDs use noncontact position tracking (magnetic or ultrasound), but this technology was not available in the 1960s.

While working on his head-mounted display, Sutherland realized that he could use computer-generated scenes instead of analog images taken by cameras, and began to design such a scene generator. This was the precursor of the modern graphics accelerator, a key part of VR hardware. The computer generates a sequence of scenes each a bit different and each displayed in a fraction of a second. The overall effect is that of animations such as flyby simulations used to train pilots. Early graphics scene generators produced by Evans and Sutherland around 1973 could display simple scenes of only 200-400 polygons. Each scene took about 1/20 sec to compute and display, so that about 20 scenes (or frames) were displayed every second. More complex scenes were made of more polygons, took a longer time to display, and therefore consisted of fewer frames per second. As a consequence, animation (which requires more than 16 frames/sec to be smooth) suffered.



Fig. 1.3 The Sensorama Simulator prototype. Courtesy of M. Heilig.

Sutherland's vision of an "ultimate display" to the virtual world was not limited to graphics. In 1965 he predicted that the sense of touch (or haptics) would be added in order to allow users to feel the virtual objects they saw

[Sutherland, 1965]. This idea was made reality by Frederick Brooks, Jr., and his colleagues at the University of North Carolina at Chapel Hill. By 1971 these scientists demonstrated the ability to simulate two-dimensional continuous force fields associated with molecular docking forces [Batter and Brooks, 1971]. Later they simulated three-dimensional collision forces using a surplus robotic arm normally used in nuclear material handling. Most of today's haptic technology is based on miniature robotic arms.

The military was very eager to test the new digital simulators, since they hoped to replace very expensive analog ones. Flight simulators were hardware designed for a particular airplane model. When that airplane became obsolete, so did its simulator, and this was a constant drain of funds. If the simulation could be done in software on a general-purpose platform, then a change in airplane models would only require software upgrades. The advantage seems obvious. Much research went on in the 1970s and early 1980s on flight helmets and modem simulators for the military, but much of this work was classified and was not published. This changed when funds for defense were cut and some researchers migrated to the civilian sector.

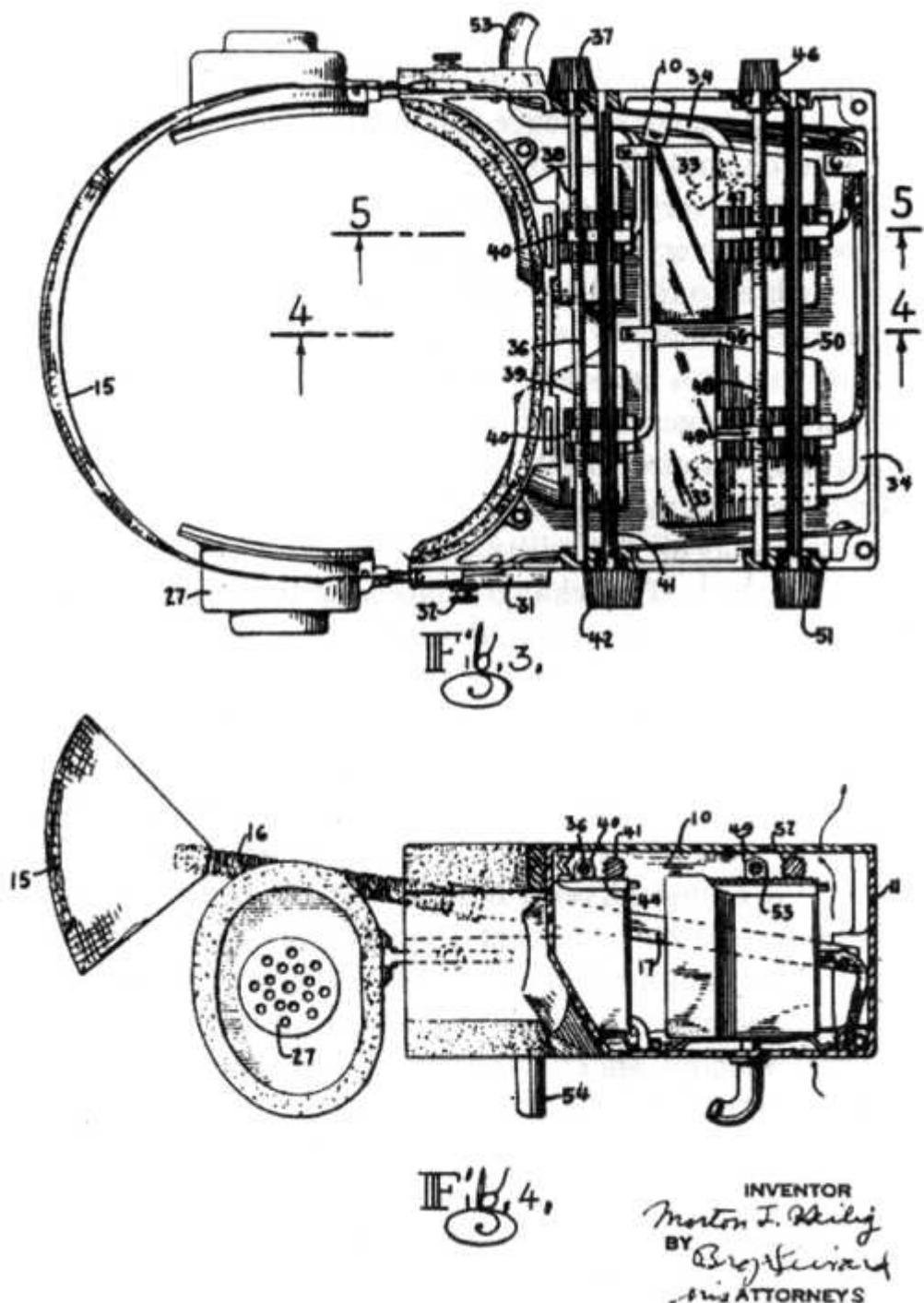


Fig. 1.4 Heilig's early head-mounted display patent. From Heilig [1960]. Reprinted by permission.

The National Aeronautics and Space Agency (NASA) was another agency of the American government interested in modern simulators. It

needed simulations for astronaut training, as it was difficult or impossible to otherwise recreate conditions existing in outer space or on distant planets. In 1981, on a very small budget, NASA created the prototype of a liquid crystal display (LCD)-based HMD, which they named the Virtual Visual Environment Display (VIVED). NASA scientists simply disassembled commercially available Sony Watchman TVs and put the LCDs on special optics. These optics were needed to focus the image close to the eyes without effort. The majority of today's virtual reality head-mounted displays still use the same principle. NASA scientists then proceeded to create the first virtual reality system by incorporating a DEC PDP 11-40 host computer, a Picture System 2 graphics computer (from Evans and Sutherland), and a Polhemus noncontact tracker. The tracker was used to measure the user's head motion and transmit it to the PDP 11-40. The host computer then relayed these data to the graphics computer, which calculated new images displayed in stereo on the VIVED.

In 1985 the project was joined by Scott Fisher, who integrated a new kind of sensing glove into the simulation. The glove was developed earlier by Thomas Zimmerman and Jaron Lanier as a virtual programming interface for nonprogrammers. A photo of Fisher experimenting with the VIVED system is shown in Figure 1.5. By 1988 Fisher and Elizabeth Wenzel created the first hardware capable of manipulating up to four 3D virtual sound sources. These are sounds that remain localized in space even when the user turns his or her head. This represented a very powerful addition to the simulation. The original VIVED project became VIEW (for Virtual Interface Environment Workstation) and the original software was ported to a newer HewlettPackard 9000, which had sufficient graphics performance to replace the wireframe rendering used in VIVED with more realistic flat-shaded surfaces.

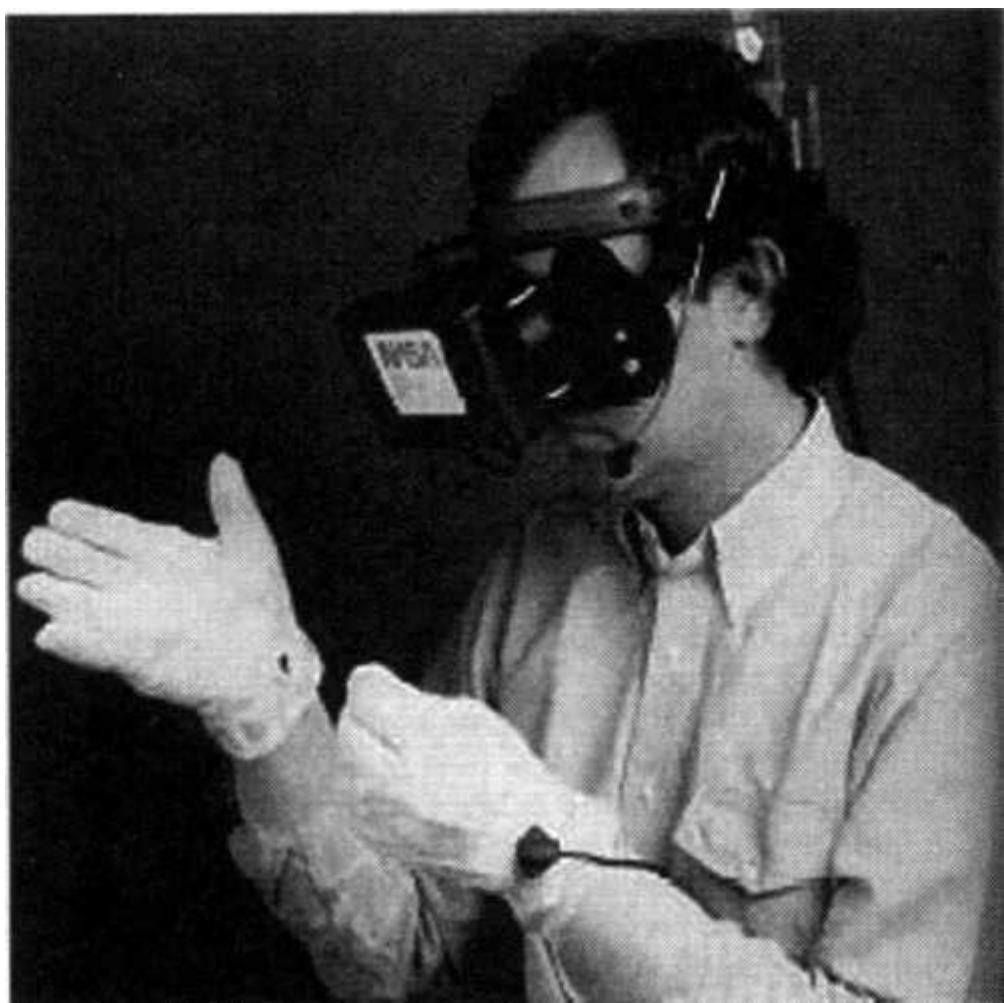


Fig. 1.5 NASA VIVED prototype. Courtesy of Telepresence Research Inc.

With all the aforementioned technological developments, scientific exchange of information among the small group of specialists of the time followed. France was one of the first countries to organize a major international conference on the subject, held in Montpellier in March 1992. The name of this conference was Interfaces for Real and Virtual Worlds, and it drew hundreds of papers and many vendors. Later the same year the United States organized the first conference on Medicine Meets Virtual Reality. In San Diego, about 180 medical practitioners met with 60 scientists and engineers to discuss the great potential of virtual reality as a tool for medicine. In September 1993 the world's largest professional society, the Institute of Electrical and Electronics Engineers (IEEE), organized its first VR conference in Seattle. Virtual reality had become part of the mainstream scientific and engineering community.

1.3 EARLY COMMERCIAL VR TECHNOLOGY

The first company to sell VR products was VPL Inc., headed by Jaron Lanier. Until its demise in 1992 this company produced the first sensing glove, called the DataGlove (Figure 1.6a) [VPL, 1987]. The standard interfaces of the time (and still today) were the keyboard and the mouse. Compared to these, the VPL DataGlove represented a quantum improvement in the natural way one could interact with computers. Its fiber-optic sensors allowed computers to measure finger and thumb bending, and thus interaction was possible through gestures. Its drawbacks were high price (thousands of dollars), lack of tactile feedback, and difficulty in accommodating different hand sizes.

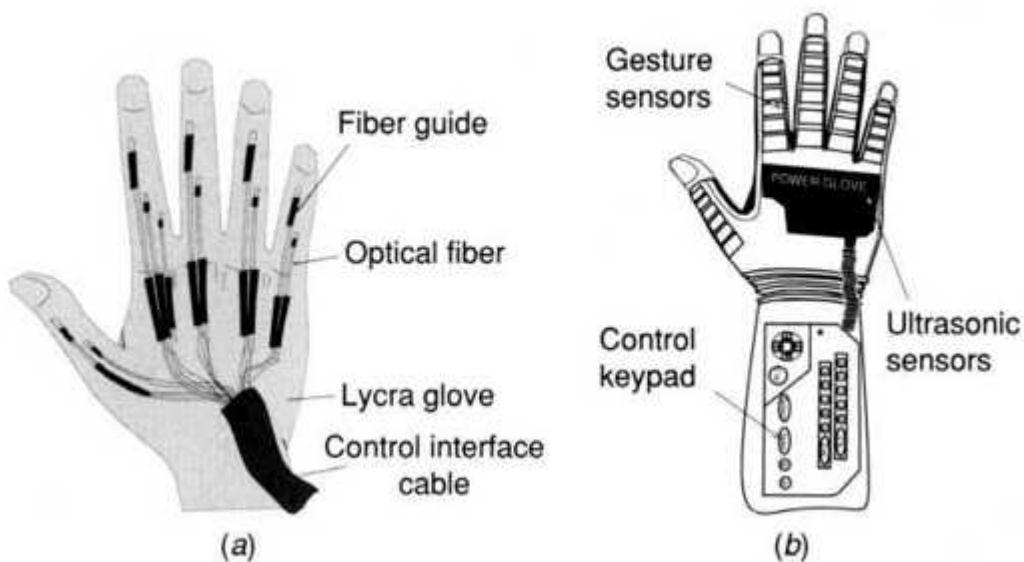


Fig. 1.6 Early sensing glove technology: (a) the VPL DataGlove; (b) the PowerGlove. From Burdea [1993]. Reprinted by permission.

Shortly after the appearance of the VPL DataGlove, the game company Nintendo introduced the much cheaper PowerGlove, shown in Figure 1.6b [Burdea, 1993]. It used ultrasonic sensors to measure wrist position relative to the PC screen and conductive ink flex sensors to measure finger bending. In 1989 almost one million such new game consoles were sold in a consumer frenzy that was later repeated with the introduction of Sony

Play Station. The downfall of the PowerGlove was lack of sufficient games that used it, such that by 1993 its production had stopped.

The first commercial head-mounted displays, called EyePhones, were introduced by VPL in the late 1980s. These HMDs used LCD displays to produce a stereo image, but at extremely low resolution (360 x 240 pixels), such that virtual scenes appeared blurred. Other drawbacks were high price (\$11,000 each) and large weight (2.4 kg).

Researchers now had an initial set of specialized hardware with which to start developing applications. However, they first had to solve various integration issues as well as develop most of the required software from scratch. The idea of a turnkey VR system originated with VPL as well. Its RB2 Model 2 offered a rack assembly housing the EyePhone HMD interface, the VPL DataGlove Model 2 electronic unit, a spatial tracking unit for the HMD, a design and control workstation, as well as connections to an SGI 4D/3 10 VGX graphics renderer and to an optional 3D sound system.

The next step in integration was to shrink each of these components and put them on a board in a single desk-side cabinet. In early 1991 a company in the United Kingdom, Division Ltd., introduced the first integrated commercial VR workstation. It was called Vision and was followed by the more powerful Provision 100 [Grimsdale, 1992], which is illustrated in Figure 1.7. The Provision 100 parallel architecture had multiple processors, stereo display on an HMD, 3D sound, hand tracking, and gesture recognition. The architecture also had an input/output (I/O) card and was scalable, allowing additional I/O processors to be added. Its i860 dedicated geometry processor with a custom polygon accelerator provided 35,000 Gouraud-shaded and Z-buffered polygons per second. This was a clear improvement over the speed of the HP 9000 computer used in NASA's VIEW system, but it came at a high price (\$70,000).

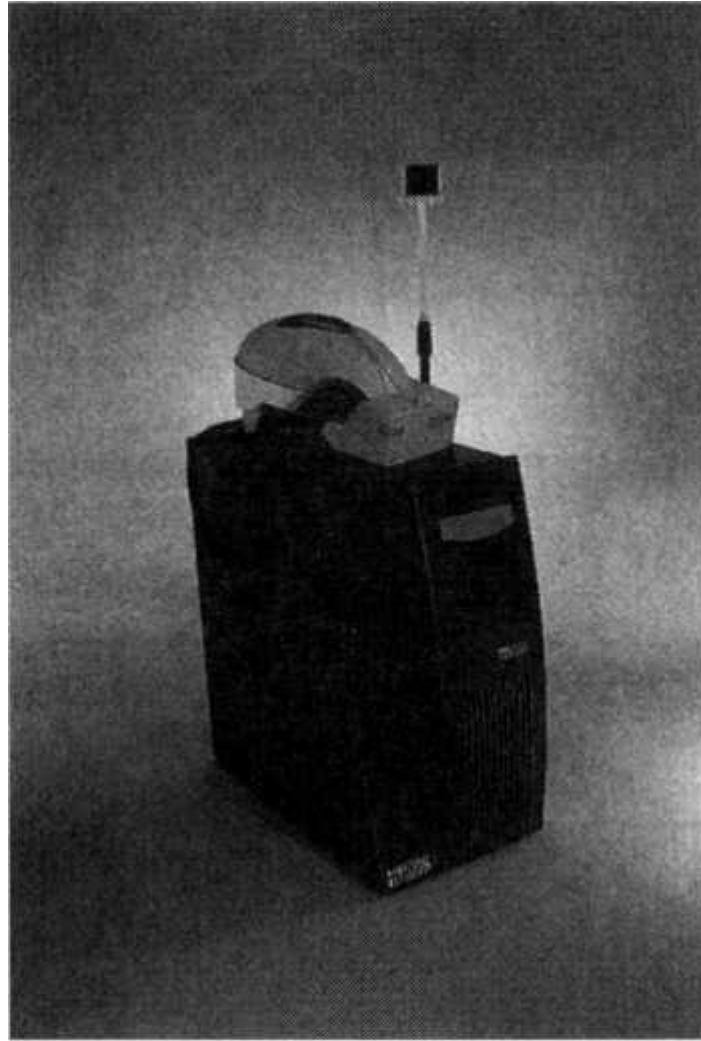


Fig. 1.7 The Provision 100 VR workstation. Courtesy of Division Ltd.

Although turnkey hardware systems appeared in early 1990s, the VR software development and debugging time continued to be a problem. In 1992, the small U.S. company Sense8 Co. developed the first version of its WorldToolKit (WTK) [Sense8 Co., 1992], a library of C functions written specifically for VR applications. With this toolkit developing VR software became more like a science, and debugging time was reduced significantly.

Another popular toolkit of the 1990s was the Virtual Reality Toolkit (VRT3), developed in the United Kingdom by Dimension International (the company later became Superscape PLC) [Dimension International, 1993]. Unlike WTK, VRT3 was designed to run on multiple computing platforms without the need for (then) pricey graphics accelerators. Also, unlike WTK,

which is text-based, VRT3 used graphical programming through menus and icons. This made programming easier to learn, but less rich in functionality, owing to the limited menu of available functions and the limited number of supported interfaces.

1.4 VR BECOMES AN INDUSTRY

The fastest graphics workstation in 1993, the Silicon Graphics Inc. Reality Engine, cost over \$100,000. This points to a very important drawback of early virtual reality hardware: It was expensive. Only large corporations, governments, and highly endowed universities could afford it. The VR market was small, estimated at \$50 million [Donovan, 1993]. Most VR pioneering companies were also small, lacking adequate resources to invest in fixing some of the problems with their first VR products. These companies relied mostly on private and venture capital. At the same time the public expectations were unrealistically high due to sustained media hype. It became clear VR could not deliver overnight all that was expected of it, and funding dried up. As a consequence many VR companies such as VPL, Division Ltd., Superscape PLC, and others disappeared. This in turn compounded the problems for VR application developers. Not only was the market for such applications untested, there was no technical support and no possibility for equipment upgrades.

By the mid 1990s VR had reached a critical point. While public attention and funding migrated to the emerging Internet and Web applications, a small group of scientists continued their VR research. Steady progress lead to a rebirth of VR in the late 1990s [Rosenblum et al., 1998]. What were some of the contributing factors? One of the most important changes was the tremendous improvement in PC hardware. Not only did the central processing unit (CPU) speed get faster, but so did the speed of PC-based graphics accelerators. According to the well-known Moore's law, CPU performance doubles every 18 months. The same law could be applied to PC graphics accelerators. In the early 1990s the speed of PC graphics boards was very poor, at 7000-35,000 polygons rendered in a second. Based on that, Moore's law predicts a speed of 20 million polygons rendered every second by 2003 [3Dlabs Inc., 2001]. Rapid technological

advances in 3D rendering chipsets made this happen much earlier, such that by 2001 the performance of PC graphics matched or exceeded that of high-end SGI graphics supercomputers, as illustrated in Figure 1.8. The tremendous reduction in price for the same performance meant that interactive 3D graphics became available to almost everyone.

Other technological advances occurred in the very important area of VR I/O interfaces. Early LCD-based color HMDs were very heavy and had poor resolution. For example, the Flight Helmet shown in Figure 1.7 had a weight of 2 kg and a resolution of only 360 x 240 pixels. By 1997 the LCD-based HMD resolution increased to 640 x 480 (VGA). A breakthrough in HMD ergonomics occurred in 1998 when Sony introduced a much slimmer Glasstron with a weight of only 310 grams. Shortly thereafter Kaiser Electro-Optics launched the first LCD-based HMD with extended graphics array (XGA) resolution (1024 x 768 pixels). Such resolution was previously available only on much more expensive and heavier CRT-based HMDs or on desktop color monitors. The steady increase in HMD resolution during the 1990s meant much sharper images without the unwanted jagged pixelation effect of earlier models. The significant improvement in HMD image resolution in the 1990s is illustrated in Figure 1.9.

Another factor in the rebirth of VR was the introduction in mid 1990s of large volume displays capable of much larger images than those available on even the most modern HMD. With such wall-size images, more users could participate in the same simulation. Fortunately for the VR manufacturers, such large-volume displays were very expensive and the VR market increased from \$500 million in 1996 to \$1.4 billion in 2000. Carl Machover, a consultant following computer graphics industry trends, predicted that 3D VR would enjoy a healthy growth at a yearly rate of 21%, and estimated that by 2005 the VR market would reach \$3.4 billion [Machover, 2000]. Figure 1.10 illustrates the growth in VR industry size, excluding the related fields of scientific visualization and real-time multimedia simulations.

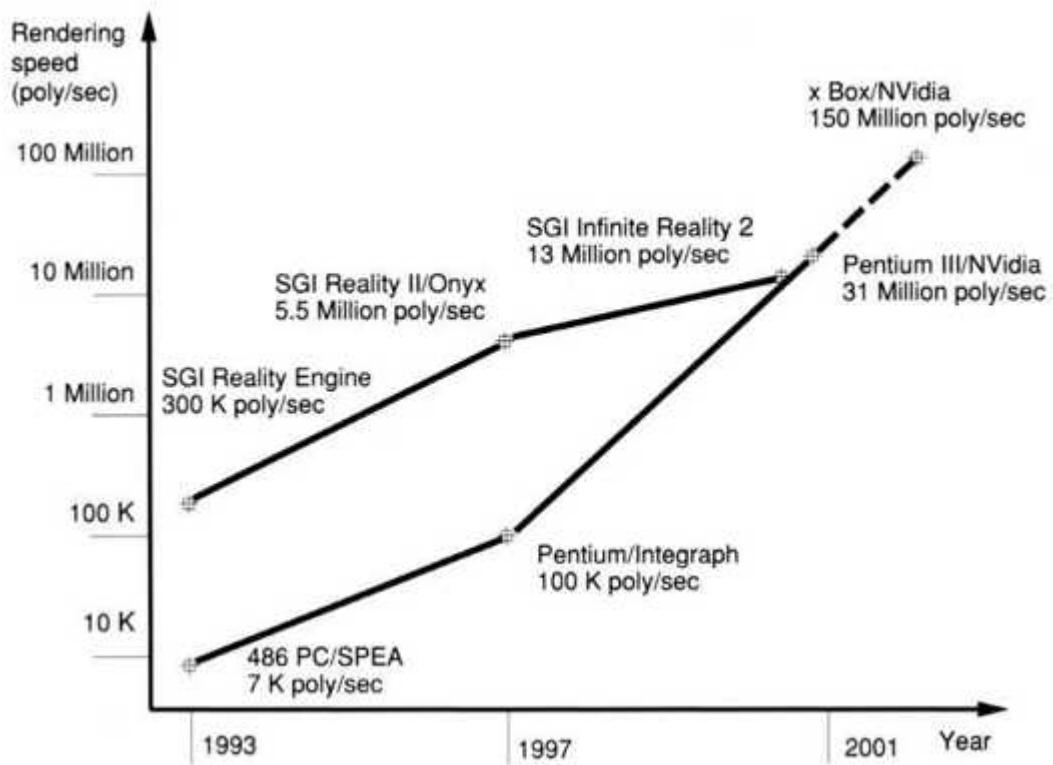


Fig. 1.8 Graphics performance comparison between PCs and high-end SGI workstations. Speed is given in polygons per second. Adapted from Burdea [2000]. Reprinted by permission.

1.5 THE FIVE CLASSIC COMPONENTS OF A VR SYSTEM

The discussion in this book will focus on the five classic components of a VR system, as depicted in Figure 1.11 [Burdea and Coiffet, 1993]. Chapters 2 and 3 concentrate on the very important UO devices used either for user input (such as trackers, gloves, or mice) or output (such as HMDs, large-volume displays, force feedback robotic arms, etc.). The special-purpose computer architecture designed to match the high UO and computation demands of real-time VR simulations is presented in Chapter 4. Chapter 5 deals with software for virtual object modeling, involving its geometry, texture, intelligent behavior, and physical modeling of hardness, inertia, surface plasticity, etc. Chapter 6 reviews a number of powerful programming packages, such as WorldToolKit and Java 3D, designed to help the virtual reality application developer. In Chapter 7 we analyze

several human factors issues affecting simulation efficiency as well as user comfort and safety. Chapter 8 discusses traditional VR applications. These integrate all the aspects discussed in earlier chapters and make VR a tool for solving various practical problems in medical care, education, arts, entertainment, and the military. A look at emerging VR applications in manufacturing, robotics, and information visualization makes up the subject of Chapter 9, which concludes the book.

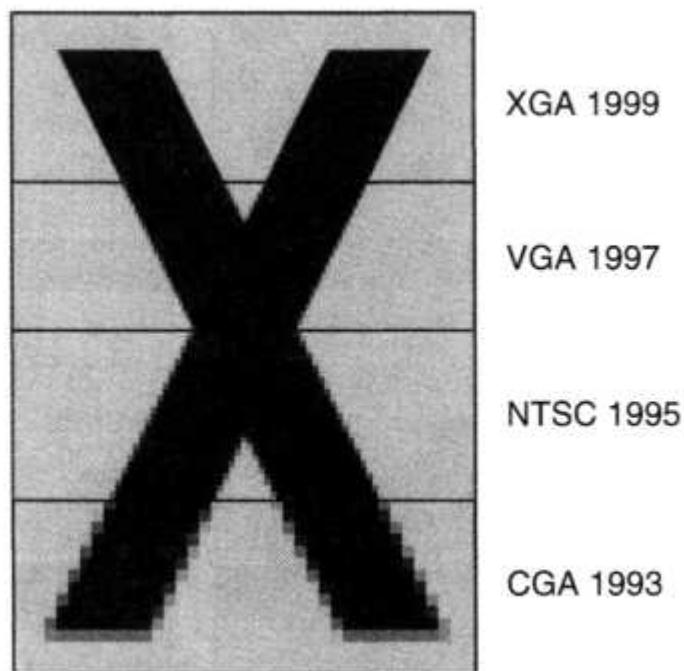


Fig. 1.9 LCD HMD image resolution evolution in the 1990s. XGA, Extended graphics array; VGA, video graphics array; NTSC, National Television System Committee; CGA, color graphics adaptor. Courtesy of Kaiser Electro-Optics Co. Reprinted by permission.

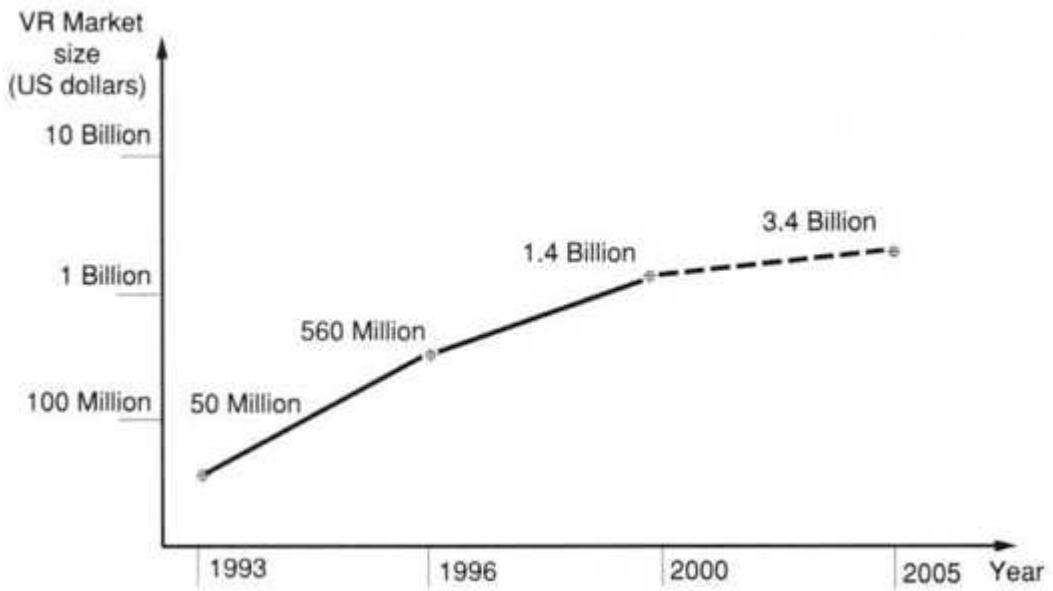


Fig.1.10 Growth of the virtual reality industry since 1993. Based on Donovan [1993], Delaney [1997], and Machover [2000].

1.6 REVIEW QUESTIONS

1. What is virtual reality?
2. How does virtual reality differ from augmented reality and telepresence?

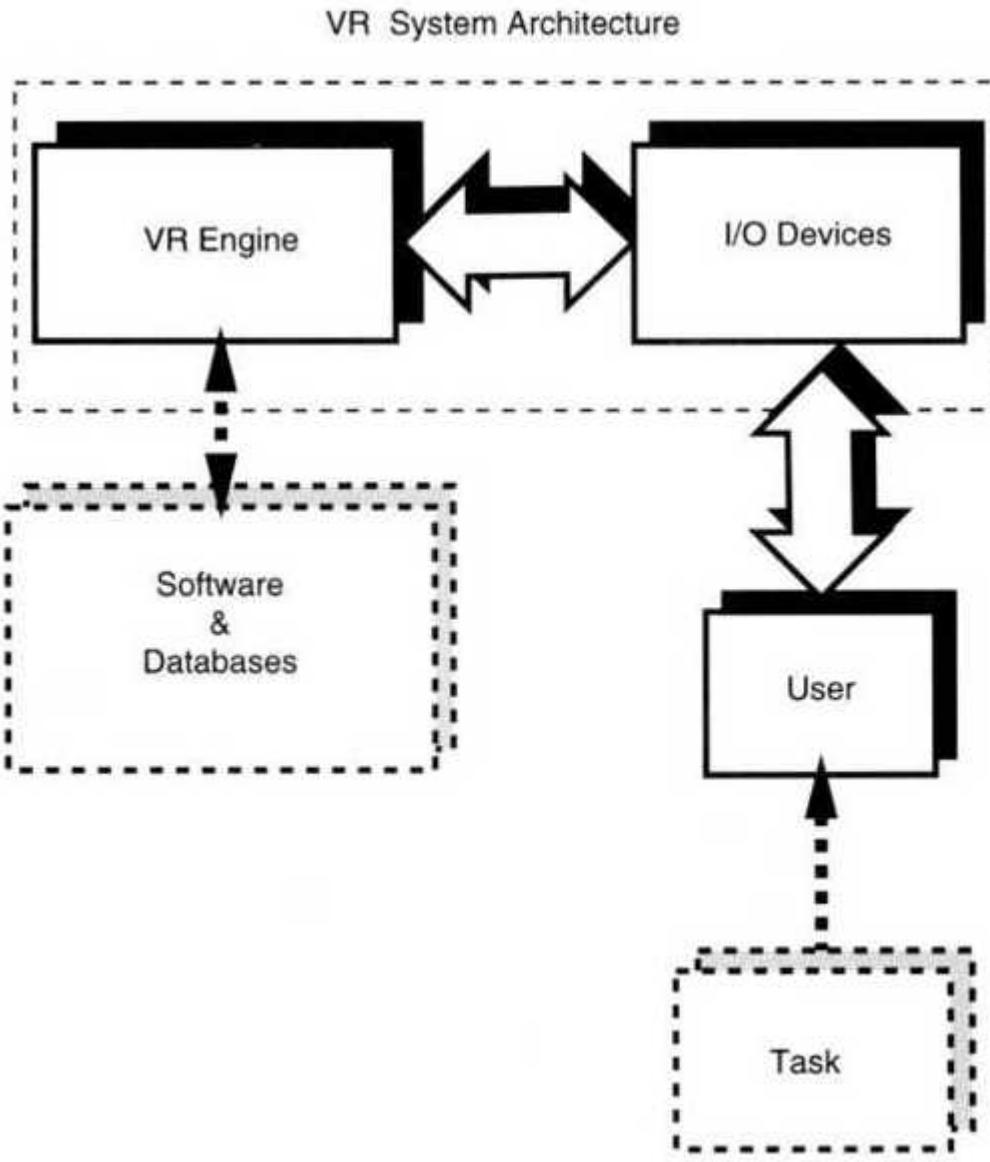


Fig. 1.11 The five classic components of a VR system. From Burdea and Coiffet [1993]. © Editions Hermes. Reprinted by permission.

3. What are commonalities and differences between virtual reality and 3D computer graphics?
4. What was Heilig's role in the development of VR?
5. What was NASA's role in early VR developments and why was it interested?

6. What were the first commercial VR products?
7. What happened with the VR industry in the 1990s?
8. What are the five classic components of a VR system?

REFERENCES

- 3Dlabs Inc., 2001, "Wildcat: Resetting Moore's Law," online at www.3dlabs.com/product/technology/mooresla.htm.
- Batter, J., and F. Brooks, Jr., 1971, "GROPE-I: A Computer Display to the Sense of Feel," in Proceedings of the ITIP Congress, pp. 759-763.
- Bejczy, A., 1993, "State-of-the-Art in Remote Manipulation Using Virtual Environment Display," in IEEE Workshop on Force Display on Virtual Environments and its Application to Robotic Teleoperation, Atlanta, GA, May.
- Burdea, G., 1993, "Virtual Reality Systems and Applications" [Short Course], in Electro '93 International Conference, Edison, NJ.
- Burdea, G., 1996, Force and Touch Feedback for Virtual Reality, Wiley, New York.
- Burdea, G., 2000, "Available Virtual Reality Techniques Now and in the Near Future" [Keynote Address], in NATO Workshop on "What is Essential for Virtual Reality to Meet Military Human Performance Goals?", April 13-15, Hague, The Netherlands, pp. 91-102 (CD ROM Proceedings March 2001).
- Burdea, G., and P. Coiffet, 1993, La Realite Virtuelle, Hermes, Paris.
- Clark, D., J. Demmel, J. Hong, G. Lafferriere, L. Salkind, and X. Tan, 1989, "Teleoperation Experiments with a Utah/MIT Hand with a VPL DataGlove," in Proceedings NASA Conference on Space Telerobotics, Pasadena, CA, Vol. V, pp. 81-89.

Codella, C., R. Jalili, L. Koved, and J. Lewis, 1993, "A Toolkit for Developing Multi-User, Distributed Virtual Environments," in IEEE Virtual Reality Annual International Symposium, Seattle, WA, September, pp. 401-407.

Delaney, B., 1997, "Taking My Best Guess," CyberEdge Journal, 1997 (January/February), pp. 18-19.

Dimension International, 1993, Virtual Reality Systems Guide, Berkshire, England.

Division Ltd., 1992, Company brochure, Division Ltd, Bristol, U.K.

Donovan, J., 1993, "Market Overview and Market Forecasts for the VR Business," in Proceedings of Virtual Reality Systems '93 Conference, SIG-Advanced Applications, New York, pp. 25-28.

Grimsdale, C., 1992, "Virtual Reality Evolution or Revolution," Division Ltd, Bristol, U.K.

Hattori, K., 1991, The World of Virtual Reality, Kogyochosakai, Tokyo, Japan.

Heilig, M., 1960, "Stereoscopic-Television Apparatus for Individual Use," U.S. Patent No. 2,955,156, October 4, 1960.

Krueger, M., 1991, Artificial Reality II, Addison-Wesley, Reading, MA.

Machover, C., 2000, "The Business of Computer Graphics," IEEE Computer Graphics and Applications, 2000 (January/February), pp. 44-45.

Muller, S., 1999, "Virtual Reality and Augmented Reality," in International Conference on Visual Computing, Goa, India.

Robertson, G., S. Card, and J. Mackinlay, 1993, "Nonimmersive Virtual Reality," IEEE Computer, Vol, 26(2), pp. 81-83.

Rosenblum, L., G. Burdea, and S. Tachi, 1998, "VR Reborn," IEEE Computer Graphics & Applications [Special Issue on VR], Vol. 18(6), pp. 21-23.

Schmult, B., and R. Jebens, 1993, "A High Performance Force-Feedback Joystick," in Proceedings of Virtual Reality Systems '93 Conference, SIG-Advanced Applications, New York, pp.123-129.

Sense8 Co., 1992, WorldToolKit Version 1.01, Technical Brief, Sense8 Co., Sausalito, CA.

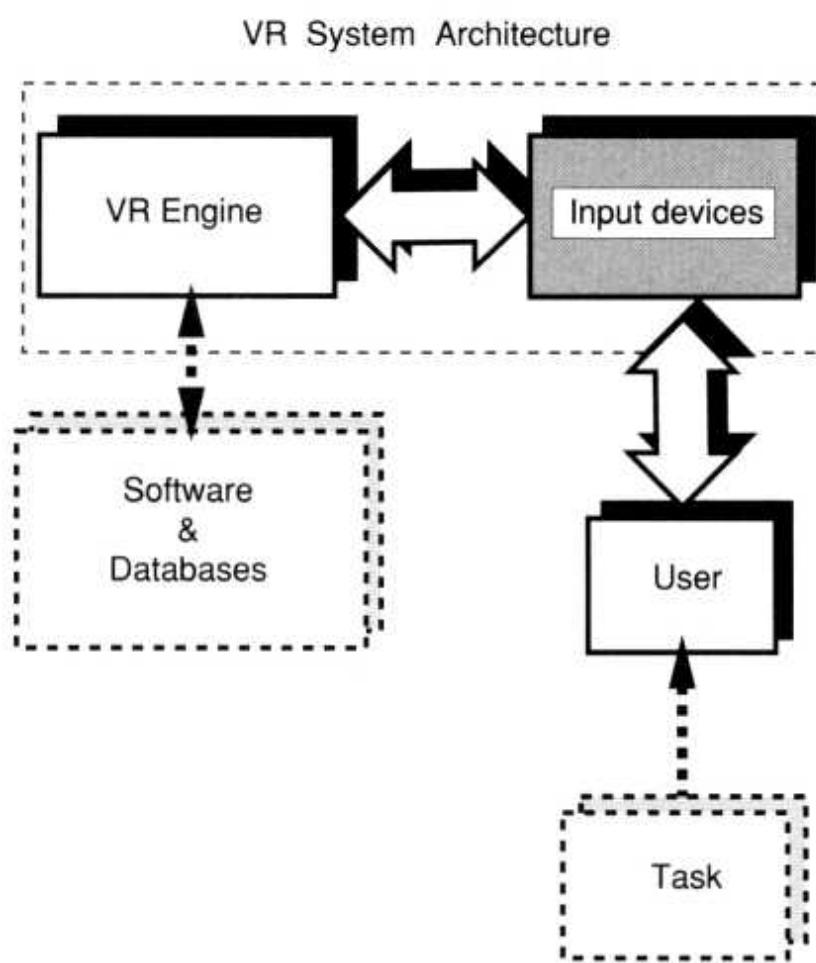
Sheridan, T., 1992, Telerobotics, Automation, and Human Supervisory Control, MIT Press, Cambridge, MA.

Sutherland, I., 1965, "The Ultimate Display," in Proceedings of the International Federation of Information Processing '65, pp. 506-508.

VPL, 1987, DataGlove Model 2 User's Manual, VPL Research, Redwood City, CA.

CHAPTER 2

INPUT DEVICES: TRACKERS, NAVIGATION, AND GESTURE INTERFACES



One of the three I's defining virtual reality stands for interactivity. In order to allow human-computer interaction it is necessary to use special interfaces designed to input a user's commands into the computer and to provide feedback from the simulation to the user. Today's VR interfaces are varied in

functionality and purpose, as they address several human sensorial channels. For example, body motion is measured with 3D position trackers or using sensing suits, hand gestures are digitized by sensing gloves, visual feedback is sent to stereo HMDs and large-volume displays, virtual sound is computed by 3D sound generators, etc. Some of these input/output (I/O) devices are commercially available, some are still prototypes in a field which has become a very active research area. The aim of researchers is to allow faster and more natural ways of interaction with the computer and thus overcome the communication bottleneck presented by the keyboard and the computer mouse.

This chapter describes the VR interfaces used in tracking, VR navigation, and gesture input. Output devices for visual, auditory, and haptic feedback to the user are the focus of the next chapter. Our discussion of VR interface hardware includes functionality, design, and limitations, comparing, whenever possible, alternative available technologies.

2.1 THREE-DIMENSIONAL POSITION TRACKERS

Many computer application domains, such as navigation, ballistic missile tracking, ubiquitous computing, robotics, biomechanics, architecture, computer-aided design (CAD), education, and VR, require knowledge of the real-time position and orientation of moving objects within some frame of reference [Hightower and Borriello, 2001]. These applications have varying requirements in terms of such parameters as measurement range, precision, and temporal update rates. Global positioning system (GPS) transducers, for example, report position with an accuracy of 1-5 m (95% of the time) on a worldwide scale. This is great for car navigation, but VR has much more stringent accuracy requirements. On the other hand, the scale of measured distances for VR interactions is usually not larger than a room or a laboratory.

A moving object in 3D space has six degrees of freedom, three for translations and three for rotations. If a Cartesian coordinate system is attached to the moving object (as illustrated in Fig. 2.1), then its translations are along the X, Y, and Z axes. Object rotations about these axes are called

yaw, pitch, and roll, respectively. These define a dataset of six numbers that need to be measured sufficiently rapidly, as the object may be moving at high speed.

Definition The special-purpose hardware used in VR to measure the real-time change in a 3D object position and orientation is called a tracker.

Virtual reality applications typically measure the motion of the user's head, limbs or hands, for the purpose of view control, locomotion, and object manipulation [Foxlin, 2002]. A newer tracker application in VR is for the control of an avatar, or virtual body, mapped to the user. In the case of the head-mounted display illustrated in Figure 2.2, the tracker receiver is placed on the user's head, so that when the posture of the head changes, so does the position of the receiver. The user's head motion is sampled by an electronic unit and sent to a host computer (in this case a graphics workstation). The computer uses the tracker data to calculate a new viewing direction of the virtual scene and to render an updated image. This scene is then converted to National Television System Committee (NTSC) video signals displayed by the two LCD screens. While this example illustrates the use of a HMD tracker, the display could have been a much larger Immersive Workbench (discussed in Chapter 3), but the requirement to measure the user's viewing direction remains. Without the 3D head tracker the computer could not have changed the spatial view to match the user's head posture, and the "immersion" sensation would have been lost.

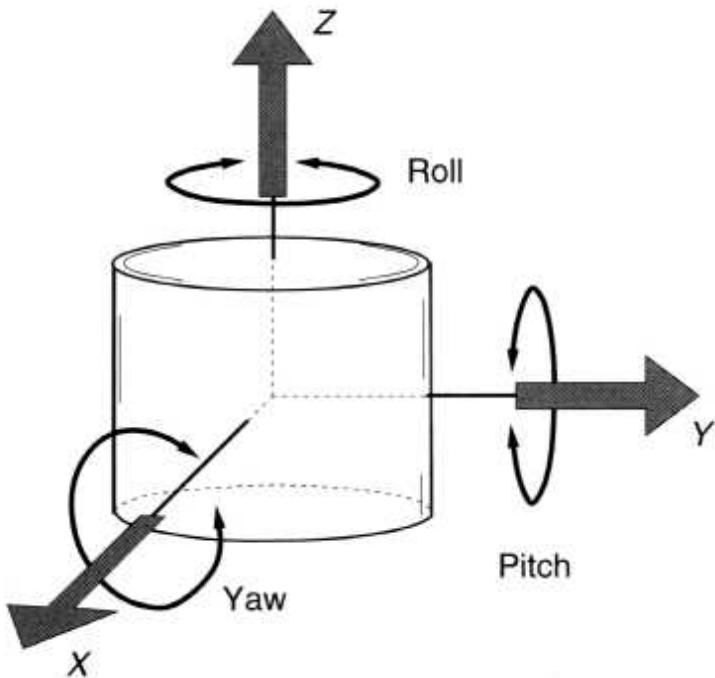


Fig. 2.1 System of coordinates of a moving 3D object. From Burdea and Coiffet [1993]. © Editions Hermès. Reprinted by permission.

Another VR sensorial modality that uses tracker information is 3D sound, which in Figure 2.2 is presented through headphones. Tracker data allow the computer to collocate sound sources with virtual objects the user sees in the simulation. This helps increase the simulation realism and the user's feeling of immersion in the synthetic world. The measurement accuracy requirements for the 3D sound application are much less stringent than those needed by the graphics feedback. As noted by Foxlin [2002], the visual acuity is higher than the auditory localization acuity, and auditory depth perception is even weaker in humans. Several competing tracking technologies are available, such as mechanical, magnetic, optical, ultrasonic, and hybrid. It is therefore necessary to look at tracker performance parameters in order to match their measurement capabilities to different sensorial channel requirements and available budgets.

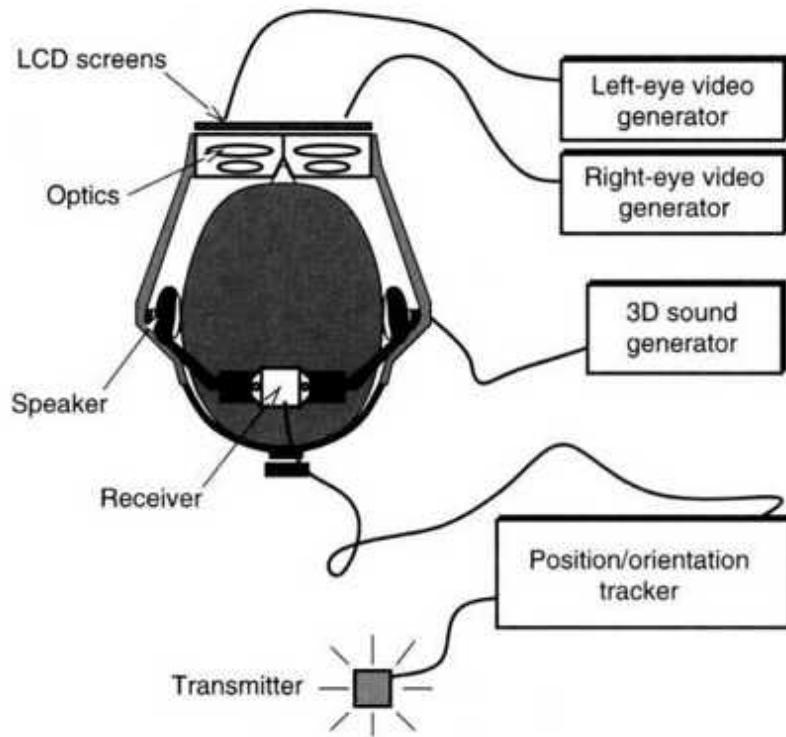


Fig. 2.2 A 3D magnetic sensor in a HMD. Adapted from Pimentel and Teixeira [1993]. Reprinted by permission.

2.1.1 Tracker Performance Parameters

All 3D trackers, regardless of the technology they use, share a number of very important performance parameters, such as accuracy, jitter, drift, and latency. These are illustrated in Figure 2.3.

Definition Tracker accuracy represents the difference between the object's actual 3D position and that reported by tracker measurements.

The more accurate a tracker, the smaller this difference is and the better the simulation follows the user's real actions. Accuracy is given separately for tracking translation (fraction of a millimeter) and rotation (fraction of a degree). Accuracy is typically not constant and is degraded with distance from the origin of the reference system of coordinates. The distance at which accuracy is acceptable defines the tracker operating range or work envelope. Accuracy should not be confused with resolution, which is the granularity or the minimum change in tracked object 3D position that the

sensor can detect. At the tip of the tracker accuracy vector shown in Figure 2.3a is the sphere of tracker repeatability. This encloses repeated measurements of a real object stationary position. Repeatability depends on tracker jitter.

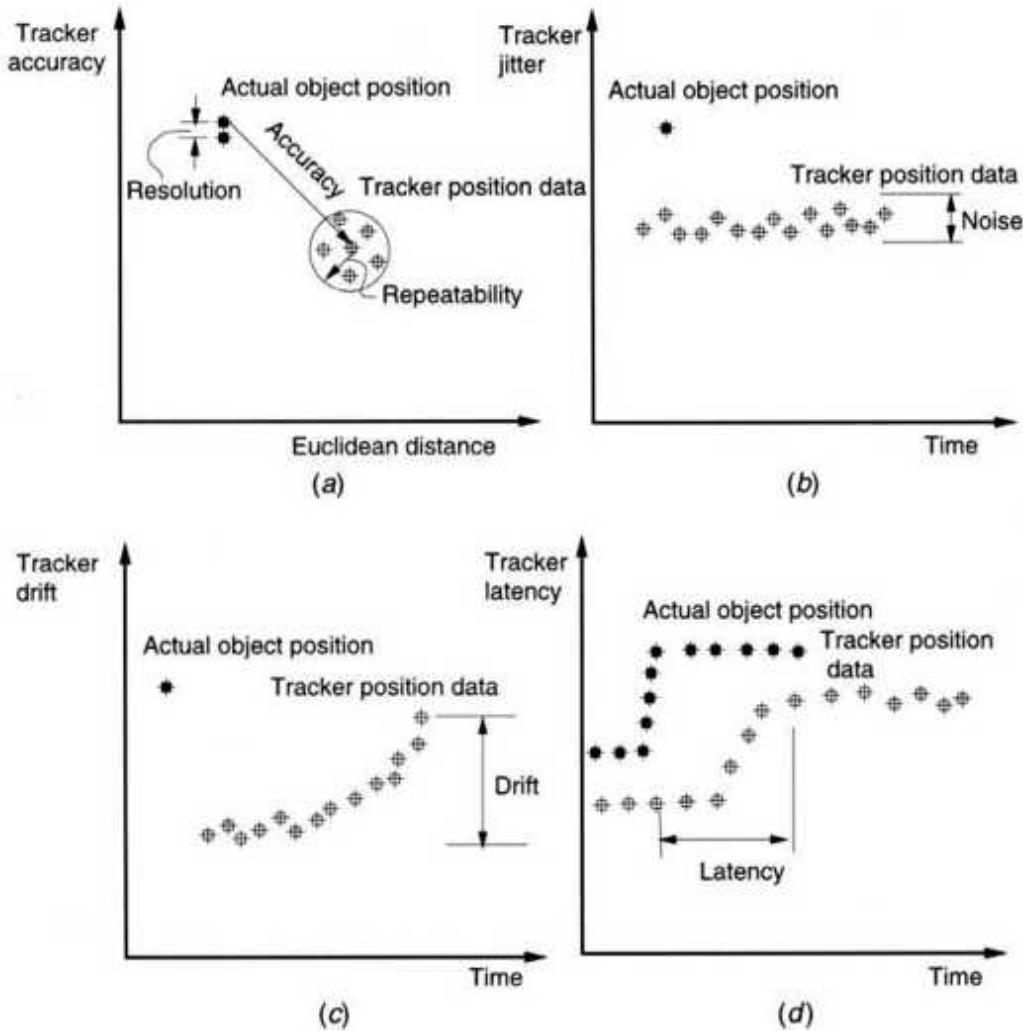


Fig. 2.3 Tracker performance parameters: (a) accuracy; (b) jitter; (c) drift; (d) latency.

Definition Tracker jitter represents the change in tracker output when the tracked object is stationary.

A tracker with no jitter (and no drift) would report a constant value if the tracked object is stationary in time. Sometimes called sensor noise, jitter makes the tracker data change randomly about an average value. Tracker

jitter needs to be minimized, since otherwise it leads to unwanted effects in terms of graphics quality (tremor, jumpy virtual objects, etc.). A noisy tracker makes accurate measurements difficult. Just like accuracy, jitter is not constant over the tracker work envelope, and is influenced by environmental conditions in the tracker's vicinity.

Definition Tracker drift is the steady increase in tracker error with time.

The output of a tracker with drift that measures the position of a stationary object is shown in Figure 2.3c. As time passes, the tracker inaccuracy grows, which makes its data useless. Drift needs to be controlled by periodically zeroing it using a secondary tracker of a type that does not have drift. This is typical of hybrid trackers discussed later in this chapter.

Definition Latency is the time delay between action and result. In the case of the 3D tracker, latency is the time between the change in object position/orientation and the time the sensor detects this change.

Minimal latency is desired, since large latencies have serious negative effects on the simulation. A tracker with large latency will contribute to a large time delay between, for example, the user's HMD motion and the corresponding motion of the virtual scene. This can induce "simulation sickness," including, for example, headaches, nausea, or vertigo. The effect that the user feels is that of the total, or system, latency. This sums up the time it takes the tracker to measure the object change in position, the communication time delay between the tracker electronics and the host computer, and the time it takes the computer to render and display the scene.

Typically the tracker measurement, communication, rendering, and display loops are asynchronous, each operating at a different speed. An efficient way to reduce system latency is to synchronize the tracker and communication loops with the display loop in what is called generation lock, or genlock. With genlock the computer receives tracker data just in time and overall system latency is reduced (but not eliminated). Whether genlock is used or not, a way to reduce system latency is to use fast communication lines. If the sensor data are sent to the host computer continuously, then the tracker operates in streaming mode. This is most

appropriate for fast-moving objects or when the application requires a quick response to a change in the moving object's position. This, however, taxes the communication lines, as a large amount of data needs to be sent. If the VR world needs to be rendered at a rate of 30 images (or frames) every second, then each frame needs to be drawn every 33 msec. A slow 9600-baud serial line will take approximately 12 msec to transmit a dataset of six numbers (each consisting of 16-bit words). Therefore, communication alone takes at least 36% of the time available. If, however, a much faster 115-kbaud serial line is used, then transmitting a tracker data packet takes only 1 msec (3% of available time). Another way to reduce system latency is to use a tracker with a high update (or sampling) rate.

Definition The tracker update rate represents the number of measurements (datasets) that the tracker reports every second.

The larger the tracker update rate, the better is the dynamic response of the simulation. Depending on the tracker technology, update rates vary between 30 and 144 datasets/sec or more. If the same tracker electronics is used to measure several moving objects, then the sampling rate suffers due to the multiplexing effect. Figure 2.4 illustrates the update rate of two types of trackers. Tracker A uses dedicated electronics for each tracked object (sensor), such that its update rate is independent of the number of tracked objects. By contrast, tracker B has only one electronic interface, which cycles between multiple tracked objects. Its update rate is reduced by 50% when two objects are tracked and by 75% when the electronics cycles among four measuring points (3D objects). Choosing between the two alternatives depends on the requirements of the VR application.

2.1.2 Mechanical Trackers

The first tracker used in a VR simulation was the mechanical arm that supported Sutherland's CRT-based HMD. The motion of the user's head was tracked with regard to the ceiling arm attachment.

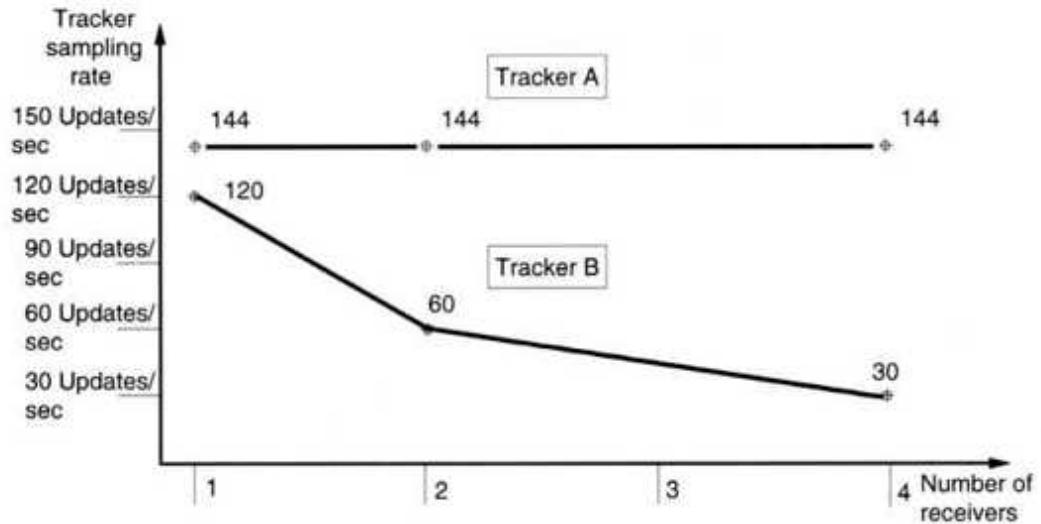


Fig. 2.4 Tracker update rate versus number of tracked objects.

Definition A mechanical tracker consists of a serial or parallel kinematic structure composed of links interconnected using sensorized joints.

The dimensions of each link segment are known a priori and used by the direct kinematics computational model stored in the computer. This model allows the determination of the position and orientation of one end of the mechanical tracker relative to the other, based on the real-time reading of the tracker joint sensors. By attaching one end of the tracker to the desk or floor and the other to an object, the computer can track the object's 3D position relative to the fixed end of the arm.

Mechanical trackers have certain advantages when compared with other tracker technologies. They are simpler and easier to use. Their accuracy is fairly constant over the tracker work envelope, and depends essentially on the resolution of the joint sensors used. Unlike electromagnetic trackers, mechanical ones are immune to interference from metallic structures or magnetic fields that may exist in their vicinity. Furthermore, mechanical trackers have very low jitter and the lowest latency of all tracking types. Unlike optical trackers, mechanical trackers have no problem with visual occlusion of the tracked object.

A mechanical tracker is used as part of the Push display [Fakespace Labs Inc., 1998; Mead et al., 2000], illustrated in Figure 2.5. This desk-top interface allows the user to navigate in virtual worlds displayed on a high-resolution stereo display. The weight of the CRT displays is supported by three compliant telescopic pistons and a bottom plate. Two of the pistons serve only for support, and their joints do not allow rotation. The third piston is connected with a three-degree-of-freedom gimbal mechanism, which allows the user to push the display in various orientations. The gimbal rotational encoders measure the degree of platform displacement front-back, left-right, and twist. The gimbal sensor data are combined with input from three buttons on the Push display handles and sent to a host computer over an RS232 serial line. The host computer uses a kinematic model to change the view to the simulation in 3D in response to the user's actions. The tracker accuracy is 4 mm, its update rate is 70 datasets/sec, and its latency is extremely low (0.2 μ sec).

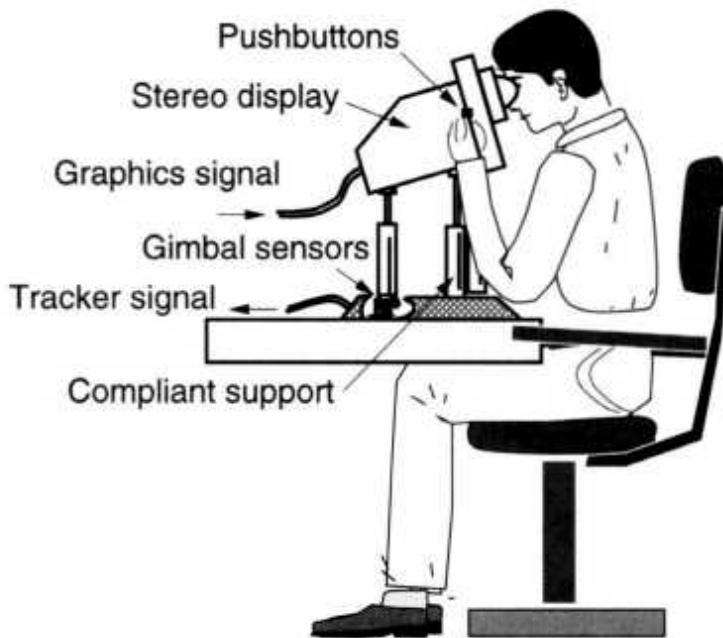


Fig. 2.5 Mechanical tracker used in the Push 1280 stereo display. Adapted from Mead et al. [2000]. Reprinted by permission of Fakespace Labs, Inc.



Fig. 2.6 Mechanical tracker used in the Gypsy 2 motion capture suit. Courtesy of Animazoo UK Ltd. Photo by Ali Kordestan.

A more complex mechanical tracker is the Gypsy motion capture suit illustrated in Figure 2.6 [ID8 Media, 1999]. It consists of a sensorized exoskeleton worn on a Lycra suit. The user's 15 joint positions are measured

by 42 single-turn conductive plastic precision potentiometers with an accuracy of 0.08°. Wires from each sensor are routed through hollow aluminum housings at the shoulders and hips. The wires are then connected to three cables that go to the host computer. Since the sensors produce an analog signal, the length of the three connection cables is limited (60 ft) to minimize resistive losses. An analog-to-digital converter card is needed by the host PC in order to sample the tracking suit more than 30 times every second. The structure of the mechanical tracker is formed of bendable aluminum plates, swiveling clips, and a rubber harness and weighs between 5 and 7 kg (12-17 lb), depending on the particular model. The mechanical tracking suit, like any other tracking suit, measures body positions relative to a body-attached reference. In order to simulate walking, running, and any other body position change relative to an external system of coordinates, the Gypsy tracking suit includes footstep microswitches [Fischer, 1998]. This is an approximate method, which works well as long as at least one foot is on the ground. The Gypsy 2.5 tracking suit includes a gyroscope, which improves the accuracy of footstep positional information by providing body orientation data independently of the mechanical tracker information.

For all their advantages, mechanical trackers have a number of drawbacks, the most obvious being their limited range, or work envelope, due to the dimension of the mechanical arm. If the links are longer, their weight and inertia increase, and their susceptibility to unwanted mechanical oscillations increases. Another drawback is the reduction in the user's freedom of motion due to the motion interference from the tracker arm itself. This is compounded if several mechanical trackers need to be used simultaneously, such as for HMD and sensing gloves. Finally, there are clear ergonomic drawbacks related to mechanical tracker weight. This poses a problem when the tracker structure has to be supported by the user, as in the case of tracker suits. It can lead to fatigue as well as a diminishing sense of immersion into virtual environments.

2.1.3 Magnetic Trackers

Whatever 3D measurement technique is used, it should not be intrusive, and should not hinder the user's freedom of motion in the process of tracking him

or her. In view of this requirement, noncontact 3D measurement techniques have largely replaced the earlier mechanical ones. The 3D noncontact trackers available today use either magnetic fields, ultrasound, infrared cameras and LEDs, or accelerometers and gyroscopes. We start our discussion with magnetic trackers, since they are the most prevalent today.

Definition A magnetic tracker is a noncontact position measurement device that uses a magnetic field produced by a stationary transmitter to determine the realtime position of a moving receiver element.

The transmitter consists of three antennas formed of three mutually orthogonal coils wound on a ferromagnetic cube. These antennas are excited sequentially to produce three orthogonal magnetic fields. These are either alternating fields of 7-14 kHz (for AC magnetic trackers) or pulsed fields (for DC magnetic trackers). The fields penetrate the receiver producing a signal that consists of nine voltages (three for each of the orthogonal transmitter fields). DC magnetic trackers add another three voltages obtained when the transmitter is turned off. These voltages correspond to the local value of Earth's DC magnetic field. The receiver consists of three small orthogonal coils when AC magnetic fields are used and three magnetometers (or alternatively Hall effect sensors) when DC magnetic fields are used [Foxlin, 2002]. The receiver voltages are sampled by an electronic unit which uses a calibration algorithm to determine the position/orientation of the receiver in relation to the transmitter. These data packets (three positions and three rotation angles) are subsequently transmitted to a host computer via communication lines. If the receiver is attached to a remote moving object, then the computer can indirectly track the motion of that object relative to the fixed transmitter.

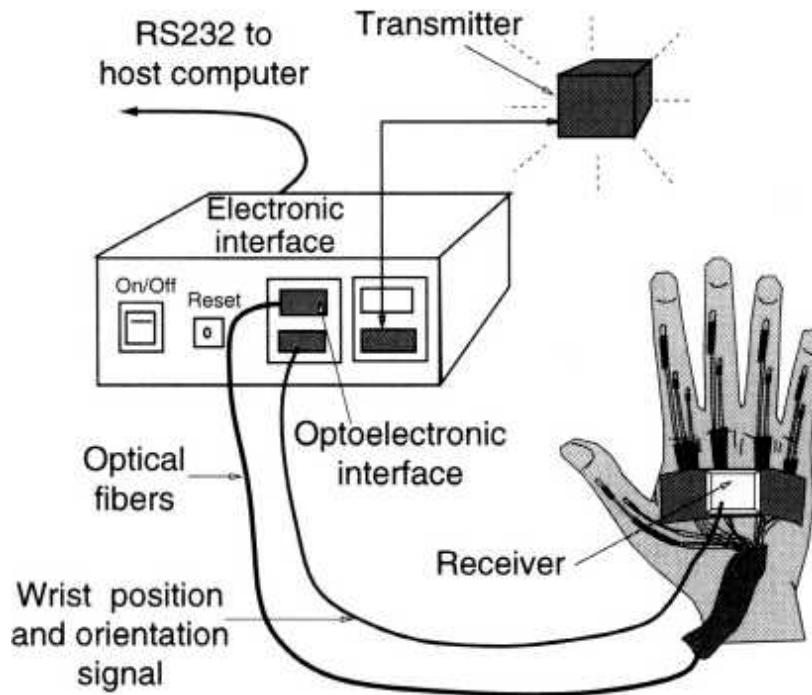


Fig. 2.7 The 3D magnetic tracker used by the VPL DataGlove. From Burdea [1993]. Reprinted by permission.

2.1.3.1 AC Magnetic Trackers. The first trackers used in VR simulations were Polhemus Isotrack AC magnetic trackers integrated with the VPL DataGlove. As illustrated in Figure 2.7 [Burdea, 1993], the tracker receiver was attached to a rubber/foam bridge over the glove optical sensors. Its electronics was integrated in the same box that housed the electrooptical circuitry necessary to read the glove bending sensors. The Isotrack latency was large (about 30 msec) owing to limitations in the settling time of the tracker analog signal. Furthermore, the Isotrack had large jitter noise (almost 1° rotation), which became even larger at the limits of the sensor range [Burdea et al., 1991]. When sampled by the host computer this noise resulted in a tremor of the virtual hand and blurred graphics.

These problems were solved by a redesigned Fastrack [Krieg, 1993], which uses a digital signal processing (DSP) architecture. As shown in Figure 2.8, a digital-to-analog (D/A) converter and amplifier allow the excitation of the three transmitter antennas with sinusoidal currents with a carrier frequency that is set to 8, 10, 12, or 14 kHz. These AC magnetic fields induce voltages in the receiver coils, which are then sent to coil-

specific, low-noise differential amplifiers. The output from these amplifiers is multiplexed with calibration signals and sent to three parallel analog-to-digital (A/D) converters. Here an oversampling technique is used by reading the receiver channels at a rate much higher than the carrier frequency. This results in an improved signal-to-noise ratio and noise shaping. In addition to reducing the jitter, the sampling rate is effectively double that of the older Isotrack. When a single receiver is used the Fastrack can measure up to 120 datasets/sec. This sampling rate drops to 60 sets/sec when two receivers are used to measure the real-time position of two moving objects with one electronic unit. With four receivers the Fastrack delivers only 30 datasets/sec due to multiplexing. The normal operating range of the Fastrack is a 75 cm (30 in.) distance between transmitter and receiver. This range can be tripled if the Long Ranger option transmitter antenna is used. This transmitter is a hollow plastic sphere 45 cm (18 in.) in diameter with three orthogonal metal coils on its outside surface. It can be attached to the ceiling or on a wooden tripod. The vendor recommends that the area around the tracker be free of large metallic objects, since these can degrade the tracker accuracy.

The problem with AC magnetic trackers is that the transmitter (time-varying) magnetic field induces eddy currents in surrounding metal. These currents in turn produce their own smaller magnetic fields. Thus the AC magnetic tracker receiver gets a distorted magnetic field and accuracy is degraded accordingly. Figure 2.9 illustrates the dependence of the Fastrack Long Ranger output on its distance from the floor. The Long Ranger was placed on its wooden tripod and raised from 1.37 m (54 in.) to 2.03 m (80 in.) from the floor in increments of 0.152 m (6 in.) [Trefftz and Burdea, 2000]. At each elevation a Fastrack receiver was moved away from the transmitter and accuracy measured. Accuracy was best when the Long Ranger was at 66 in. from the laboratory floor. Metal beams used in the construction of the building degraded accuracy when the Long Ranger was closest to the floor (54 in.) and to the ceiling (80 in.).

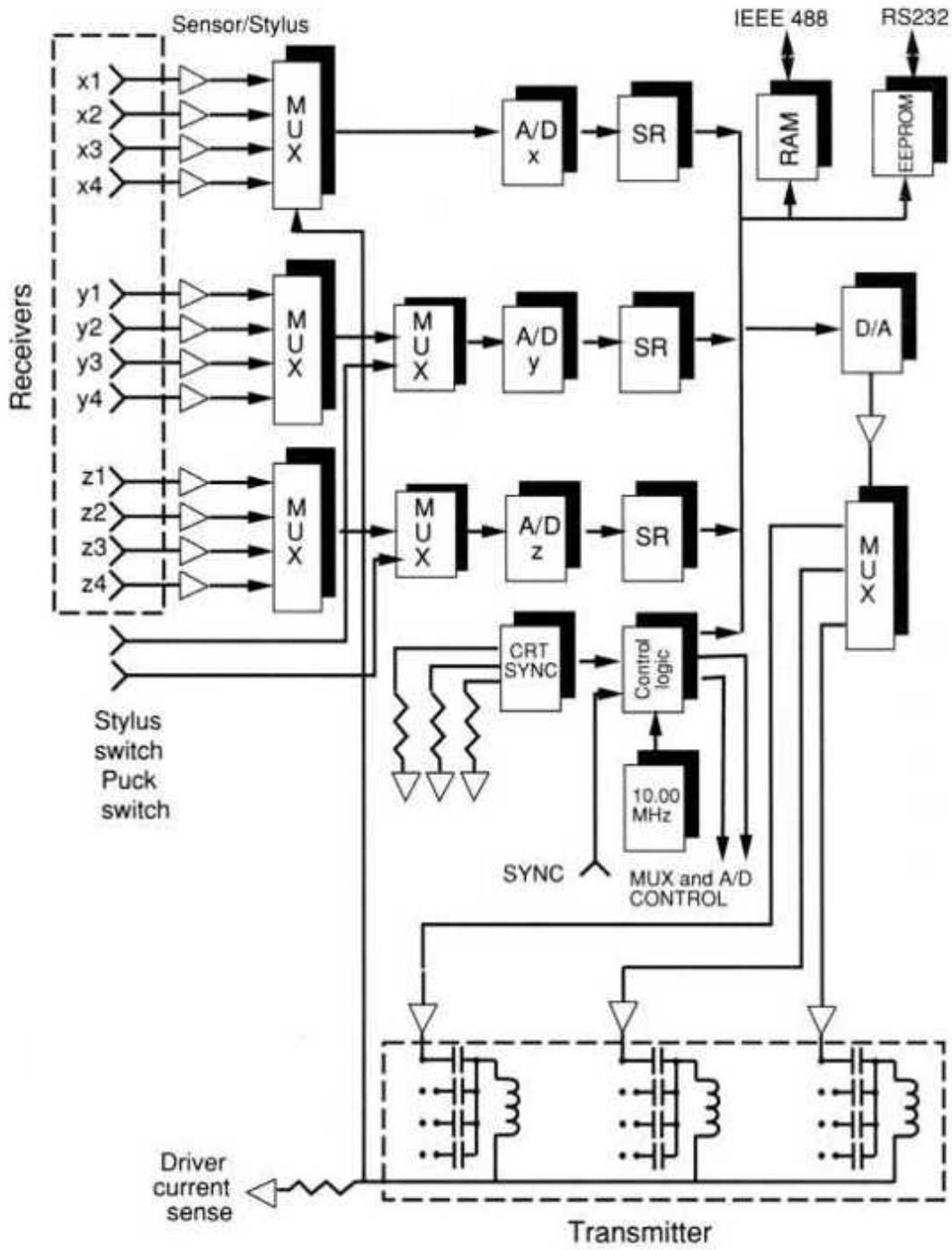


Fig. 2.8 Fastrack® block diagram. Adapted from Krieg [1993]. Reprinted by permission.

2.1.3.2 DC Magnetic Trackers. In order to alleviate the eddy current problem Blood [1989] proposed replacing the transmitter AC magnetic field with a DC one. His idea was to induce a short time delay between the

excitation of the transmitter and the sampling of the receiver to allow for the disappearance of eddy currents. Figure 2.10 illustrates the functioning of a typical DC magnetic tracker. It contains an embedded microprocessor that controls the amplitude of the transmitter DC magnetic pulses. This is realized by sending analog voltages to current sources connected to the transmitter coils. The higher the control voltage, the higher is the current produced by the current source, and thus the amplitude of the magnetic pulse. A multiplexer allows the microprocessor to cycle the current sources, such that three orthogonal DC magnetic fields are produced, one at a time. During time period marked TO-T 1 all current sources are off, from T_i to T₂ the X field is on, from T₂ to T₃ the Y field is on, and finally from T₃ to T₄ the Z field is on. Subsequently, the cycle is repeated. The transmitter fields induce voltages in three orthogonal antennas in the receiver. These induced voltages are sampled in sequence by the microprocessor, which controls a multiplexer on the receiver output. The receiver multiplexer signal is first sent to a differential amplifier, which subtracts a voltage that is provided by the microprocessor. This voltage corresponds to the local value of Earth's DC magnetic field and is stored by the microprocessor during the time period TO-T 1 when the transmitter is off. Thus the output from the differential amplifier corresponds to the voltages induced only by the transmitter antennas. Its output is subsequently filtered (to reduce noise) and digitized. The microprocessor then uses a calibration algorithm to determine the position/orientation of the receiver relative to the fixed transmitter.

A magnetic tracker that uses DC magnetic fields is the Flock of Birds produced by the Ascension Technology Co. [Ascension, 1998], illustrated in Figure 2.11. The Flock of Birds uses a distributed computing architecture to reduce computation time and maintain high tracker update rates. There are three electronic units, named Bird 1, 2, and 3, interconnected with a fast serial bus, called the Fast Bird Bus (RS-485, with rates up to 500 kbaud). The first microprocessor unit (designated the master) has both a transmitter, and a receiver. The other two are slave microprocessors that serve only to sample their receivers when prompted by the master microprocessor (Bird 1). Since each receiver's data are processed in parallel, update rates are not degraded with added receivers. The Flock of Birds maintains a global update rate of up to 144 datasets/sec, even with 30 receivers connected to

one transmitter. The operating range of the tracker is normally 1.2 m (48 in.) and can be extended to 3 m (120 in.) if an extended range transmitter (ERT) is used. The ERT has its own extended range controller, which becomes the master microprocessor for synchronization purposes.

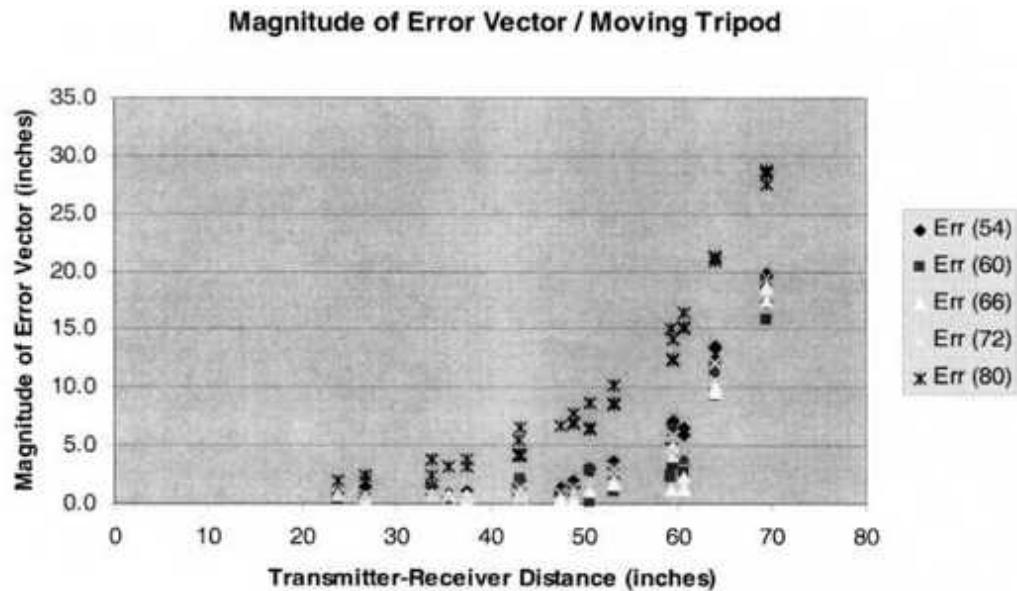


Fig. 2.9 Fastrack Long Ranger accuracy degradation due to surrounding metal in the floor and ceiling of the room.

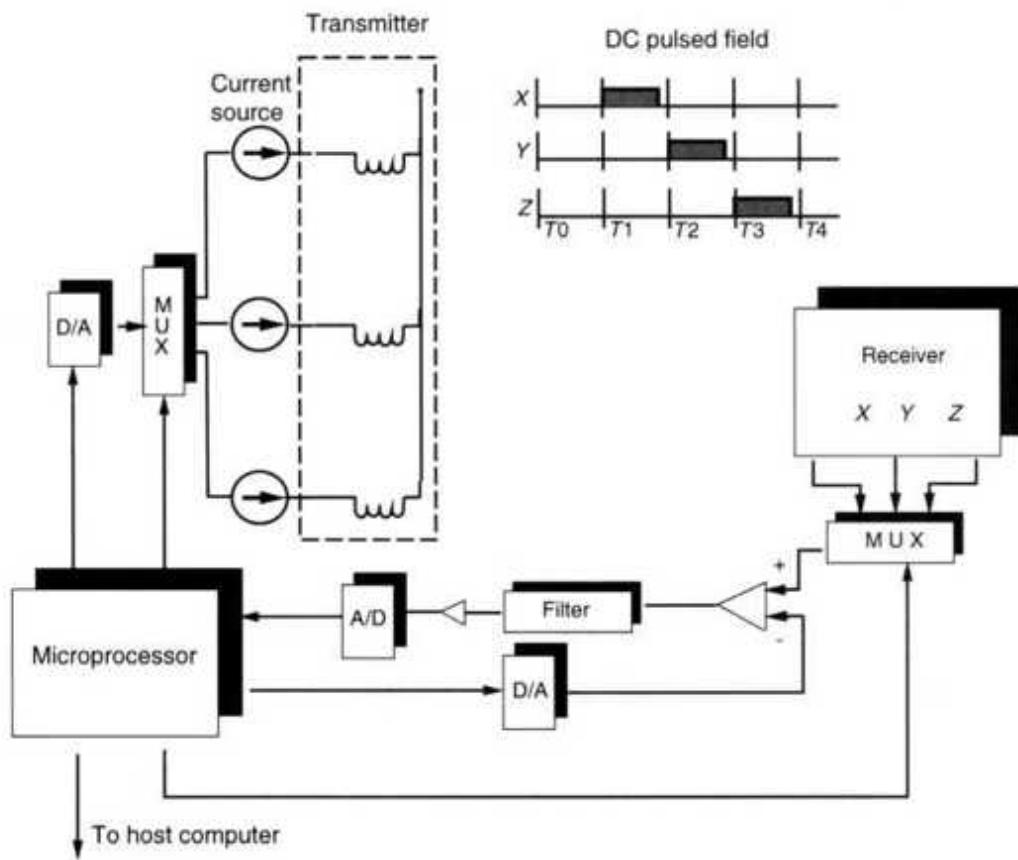


Fig. 2.10 DC magnetic tracker block diagram. Adapted from Blood [1989]. © Ascension Technology Co. Reprinted by permission.

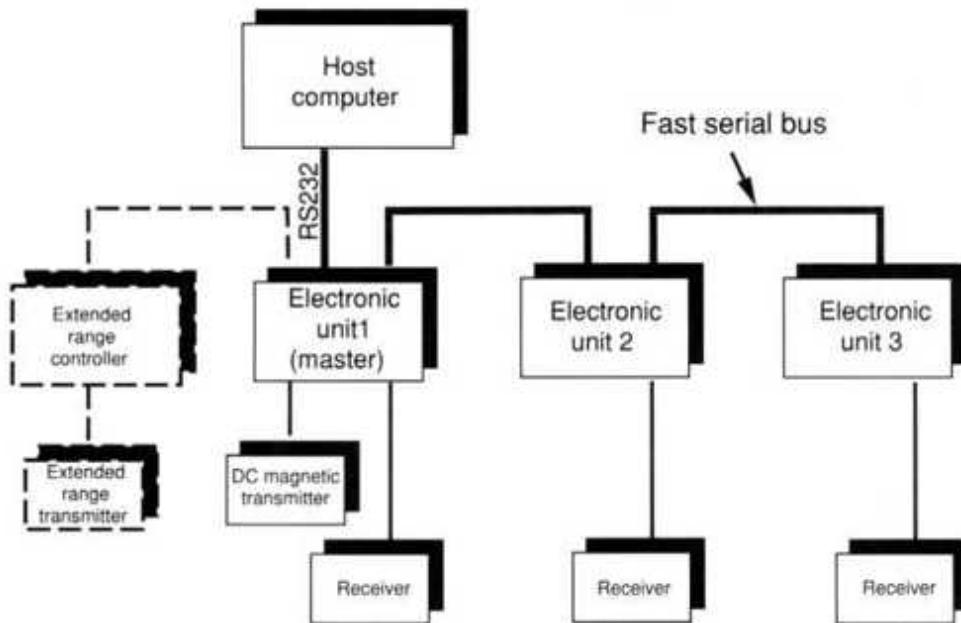


Fig. 2.11 Flock of Birds block diagram with optional extended range controller. Adapted from Ascension [1998]. © Ascension Technology, Co. Reprinted by permission.

There are applications where it is desirable to have a large number of receivers without a significant drop in the update rate. Such an application is avatar control for character animation and motion capture, where full-body motion needs to be measured over larger areas. Furthermore, several users (actors) may need to be tracked at the same time, and their motion should not be hindered by a myriad of cables running from their bodies all the way back to the sensor-processing electronics. Figure 2.12a illustrates the block diagram of the MotionStar wireless suit developed by Ascension Technology Co. to accommodate such a need. An extended range controller drives two extended range transmitters in a time-sequential way. First ERT1 sends a sequence of DC magnetic pulses, then ERT2, then ERT1, and so on. Depending on whether the two ERTs face each other (X axis of ERT1 aligned with -X of ERT2) or are placed side by side (the two transmitter systems of coordinates are aligned), different tracking surfaces are created. In the first case the tracking surface for optimal accuracy is 3.6 m x 3.6 m (12 ft x 12 ft), and in the second arrangement optimal tracking is over a 1.8 m x 4.2 m surface (6 ft x 14 ft). A total of 11-20 DC magnetic receivers is placed on the user's body on the hands, elbows, knees, heels, torso, head, toes, and spine (Fig. 2.12b, c). All receivers are wired to a backpack unit. The backpack contains processing electronics, a battery and an 11-Mbits/sec wireless modem. The total weight (receivers, wires, backpack, and battery) is about 1.7 kg (3.8 lb), which is much less than the weight of the Gypsy mechanical tracker suit discussed earlier. Up to 100 datasets/sec are transmitted by the backpack electronics to a base station that contains additional processing electronics. The base station processor knows which readings result from the field created by ERT1 and which correspond to ERT2. The base station calibration algorithm combines these sets by weighting more the data corresponding to the closer ERT (ERT2 in this case). This improves accuracy because the user is outside of the recommended 3-m range (10 ft) from ERT1. Several backpack-base station pairs can be used in order to track up to five users (each at 100 datasets/sec). In this case each wireless modem frequency needs to be

different to minimize interference. The base stations for each user are then daisy-chain-connected to the master base station, and data are sent to the host computer.

2.1.3.3 Comparison of AC and DC Magnetic Trackers. Both AC and DC magnetic trackers measure the distance $d_{transmitter_Leceiver}$ based on the local value of the transmitter magnetic field v . This intensity falls with the cube of the distance from the transmitter, and the relative importance of unwanted disturbances A_v grows accordingly. Nixon and his colleagues at Applied Research Associates (Christchurch, New Zealand) [Nixon et al., 1998] determined that the position measurement error due to ambient noise $e_{ambient}$ is given by

$$e_{ambient} = K_a(d_{transmitter-receiver})^4 \quad (2.1)$$

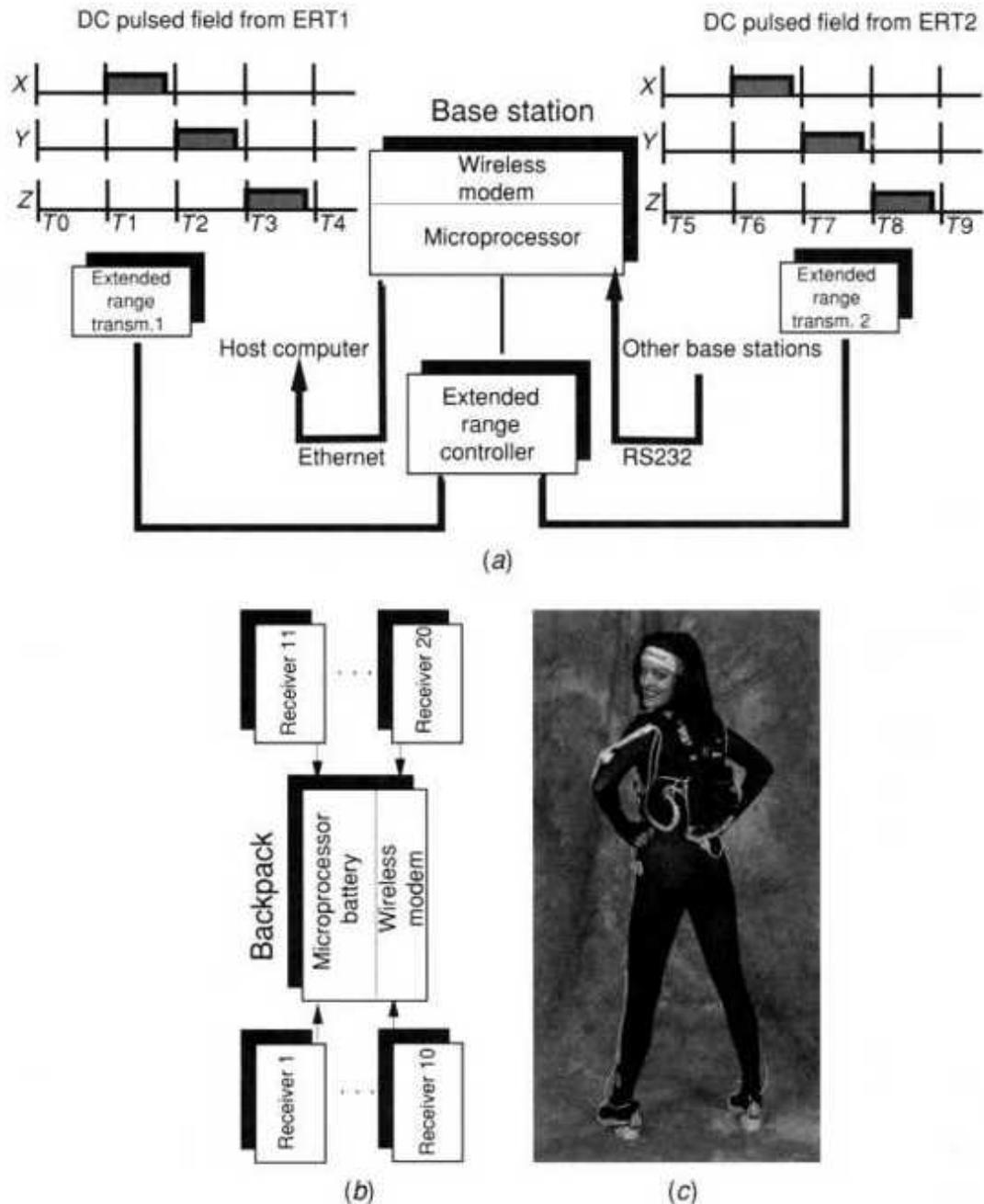


Fig. 2.12 MotionStar© wireless system: (a) block diagram; (b) back pack unit. Adapted from Ascension [2001b]; (c) User wearing the tracking suit. © Ascension Technology Co. Reprinted by permission.

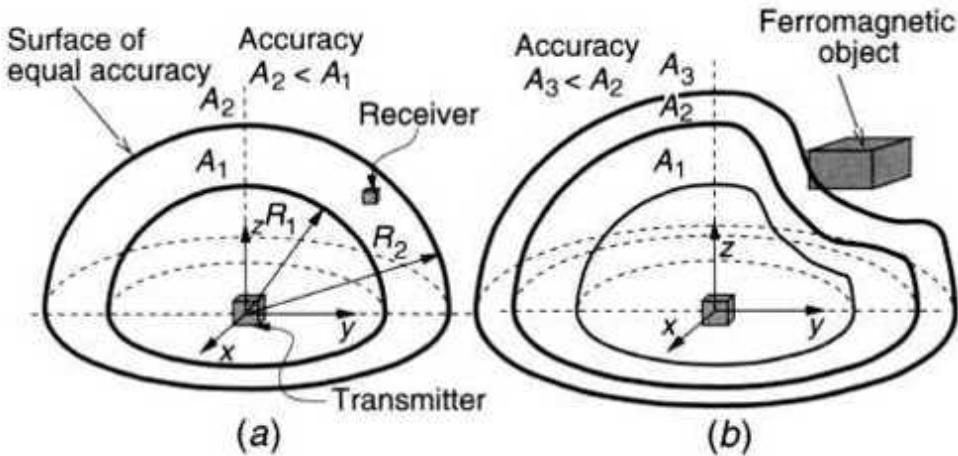


Fig. 2.13 Magnetic tracker accuracy degradation: (a) with distance; (b) due to metallic objects. From Burdea [1993]. Reprinted by permission.

where K_a is a proportionality constant. Surfaces of equal accuracy are thus hemispheres centered on the transmitter. As illustrated in Figure 2.13a, accuracy A_2 is less than A_1 , as radius R_2 is larger than R_1 .

Sources of magnetic interference vary depending upon whether the tracker uses AC or DC fields. The AC fields produced by a building's electrical wiring have frequencies of 60 Hz in North America and 50 Hz in Europe. Such frequencies are close to the operating frequency of DC trackers and require corrective measures. Nixon recommended a synchronization of the tracker with the mains supply and a sampling rate at twice the mains frequency. The Flock of Birds needs an added two-tap filter (averaging two adjacent signals) to cancel this interference. This added filtering has a negative effect on the dynamic response of the tracker. In a one transmitter, one receiver configuration the Flock of Birds latency without filtering (7.5 msec) was found to be comparable with that of the Fastrack (8.5 msec) [Adelstein et al., 1996]. However, the added filtering needed by the Flock of Birds to compensate for mains interference increases its latency by 50% to 11.5 msec. Further interference comes from CRT monitors that may be used to display graphics. Nixon [1998] recommended at least 0.8 m separation distance between CRT monitors (with a vertical refresh rate of about 70 Hz) and magnetic tracker receivers. This distance is conservative, since stereo monitors (discussed in a subsequent chapter) have refresh rates of 120 Hz.

Another major source of ambient interference is metal in the vicinity of either AC or DC trackers. The error ϵ_{metal} due to the presence of metal within a magnetic tracker work envelope is given by Nixon [1998] as

$$K_r (d_{\text{transmitter-receiver}})^4 \epsilon_{\text{metal}} = 3 \frac{1}{d_{\text{transmitter-metal}}} X (d_{\text{metal-receiver}}) \quad (2.2)$$

where K_r is a proportionality constant. Metals contribute in two ways to the degradation of magnetic tracker accuracy. The first is through eddy currents produced by a time-variant magnetic field (AC fields, as well as the rising and falling edges of DC magnetic fields). DC trackers are essentially immune to these, and the Flock of Birds is unaffected by the presence of nonferromagnetic metals such as brass, aluminum, and stainless steel. Both AC and DC trackers are affected by the presence of ferromagnetic metals such as mild steel and ferrite. The high magnetic permeability of such metals allows induced magnetic fields to exist for AC or DC trackers. Since magnetic permeability is higher for lower frequency fields, the Flock of Birds is affected more than the Fastrack by the presence of mild steel [Nixon et al., 1998]. Both trackers are affected by copper, even though it is nonferromagnetic. Its high conductivity allows eddy currents to exist longer than the time delay used in DC trackers, essentially canceling their advantage. In conclusion, magnetic tracker accuracy is poorer closer to metallic objects than it would otherwise be, as illustrated in Figure 2.13b. Large metallic objects need to be removed from the area close to the transmitter or receiver, otherwise very complicated calibration and compensation measures are needed.

Table 2.1 shows a performance comparison between the Fastrack and the Flock of Birds.

2.1.4 Ultrasonic Trackers

An alternative tracking solution that does not suffer from metal interference uses 3D ultrasound trackers.

Definition A ultrasound tracker is a noncontact position measurement device that uses an ultrasonic signal produced by a stationary transmitter to

determine the real-time position of a moving receiver element.

Ultrasound trackers have three components, a transmitter, a receiver, and an electronic unit, similar to their magnetic counterparts. The difference is that the transmitter is a set of three ultrasonic speakers mounted about 30 cm from each other on a rigid and fixed triangular frame. Similarly, the receiver is a set of three microphones mounted on a smaller rigid triangular frame. This triangular frame is placed at the top of the headmounted display, as illustrated in Figure 2.14. Alternatively the microphones may be part of 3D mice, stereo glasses (discussed in the next chapter), or other interface devices. Due to their simplicity, ultrasound trackers represent a cheaper alternative to the magnetic ones.

TABLE 2.1. Performance Comparison: Fastrack Versus Flock of Birds

Specification	Fastrack	Flock of Birds
Operation radius		
Normal	0.75 m (30 in.)	1.2 m (48 in.)
Extended	2.25 m (90 in.)	3 m (120 in.)
Angular range	All attitudes	±180° azimuth and roll, ±90° elevation
Translation accuracy	0.03 in. RMS	0.1 in. RMS
Translation resolution	0.0002 in./in. range	0.03 in. RMS
Angular accuracy	0.15° RMS	0.5° RMS
Angular resolution	0.025° RMS	0.1° RMS at 12 in.
Update rate (measurements/sec)	120 (1 receiver) 60 (2 receivers) 30 (4 receivers)	144 (\leq 30 receivers)
Latency (single receiver)	8.5 msec (no filtering)	7.5 msec (no filtering)
Metal interference	Ferrite, mild steel, copper, stainless steel, brass, aluminum	Ferrite, mild steel, copper
Interface	RS232 (selected baud rates to 115,200) or IEEE-488 (up to 100 kbaud/sec)	RS232 (selected baud rates to 115,200) or RS422/485 (selected baud rates to 500,000)
Data format	ASCII or binary	Binary
Modes	Point or stream	Point or stream

The speed of sound in air changes for room temperature based on the law [Sheingold, 1981]

$$c = (167.6 + 0.6Tk) \quad (2.3)$$

where c is the speed of sound in m/sec and Tk is the air temperature in degrees Kelvin. For a given temperature the speed of sound is known and can be used to measure distances based on time of flight. The ultrasound tracker measurements are based on triangulation. Each speaker is activated in cycle and the three distances from it to the three microphones in the receiver are calculated. A total of nine distances is measured in order to determine the position and orientation of the plane that contains the three microphones. The control unit CPU samples the microphones, converts their readings into position and orientation based on calibration constants, then transmits the data to the host computer for graphics scene rendering. The update rate of ultrasound trackers is about 50 datasets/sec, which is less than half that of modern magnetic trackers. The reason update rates are low is the need to wait 5-100 msec to allow echoes from a previous measurement to die out before a new measurement is initiated [Foxlin, 2002]. When several parts of the body (such as head and hands) have to be tracked, it is possible to use time multiplexing (similar to magnetic trackers) of up to four receivers with one transmitter. The drawback in this case is a further increase in simulation latency, since time multiplexing reduces the tracker update rate even more. When four receivers are tracked simultaneously, the update rate drops to (only) 12 datasets/sec! Using a single transmitter with multiple receivers has the additional drawback of limiting the total volume where the user's head and hands have to be. The operating range of ultrasound trackers is dependent on the attenuation of the transmitter signal due to molecular air absorption. A typical range might be 1.52 m (5 ft) from the ultrasound transmitter, but this is reduced significantly by the relative humidity of the air.

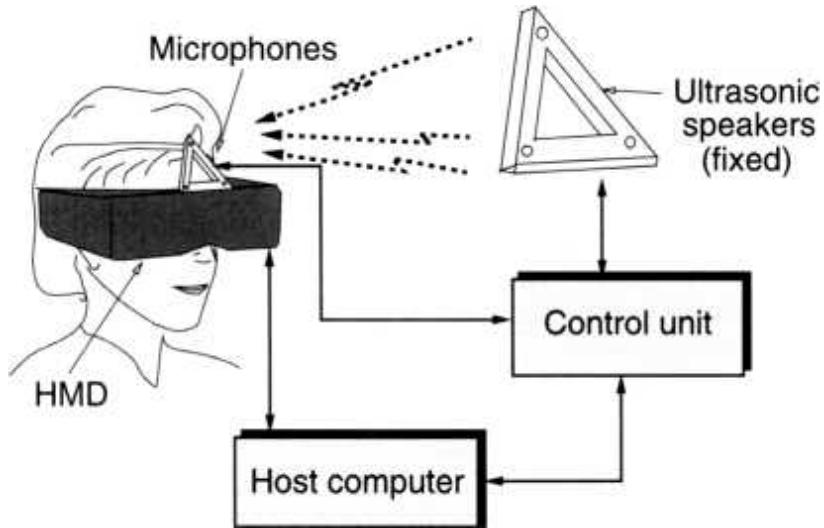


Fig. 2.14 Logitech ultrasound head tracker. From Burdea and Coiffet [1993].
© Editions Hermes. Reprinted by permission.

A direct line of sight is required between the transmitter and the receiver of an ultrasound tracker. This is another significant drawback compared to magnetic trackers, which do not require direct line of sight. If some object obstructs the line of sight between an ultrasound transmitter and receiver or the user's head is turned away, the tracker signal is lost. The signal is also corrupted by background noise in the room as well as other ultrasound sources (such as the ones used in building security).

Certain applications may require a larger user motion volume than is allowed by a single transmitter. The solution is to spatially multiplex several transmitters with a single receiver [Sowizral and Barnes, 1993]. The transmitters have to be placed such that their conic tracking volumes overlap (as illustrated in Fig. 2.15). In order to avoid mutual interference, it is necessary to turn on only one transmitter at a time. Thus the host computer has to determine where the receiver is and keep the corresponding transmitter on until the neighboring transmitter takes over, and so on. Rather than have several control boxes (one for each transmitter), it is more economical to interpose a switch box controlled by the computer. The analog switches in the box are normally closed on transmitter 1 and commute to transmitter 2 only when energized by the computer. This scheme can be extended to multiple transmitters. The computer activates the box based on predictive control, that is, determining the receiver position,

velocity, and acceleration. Hysteresis is provided in order to avoid cycling the switches in the overlap zone.

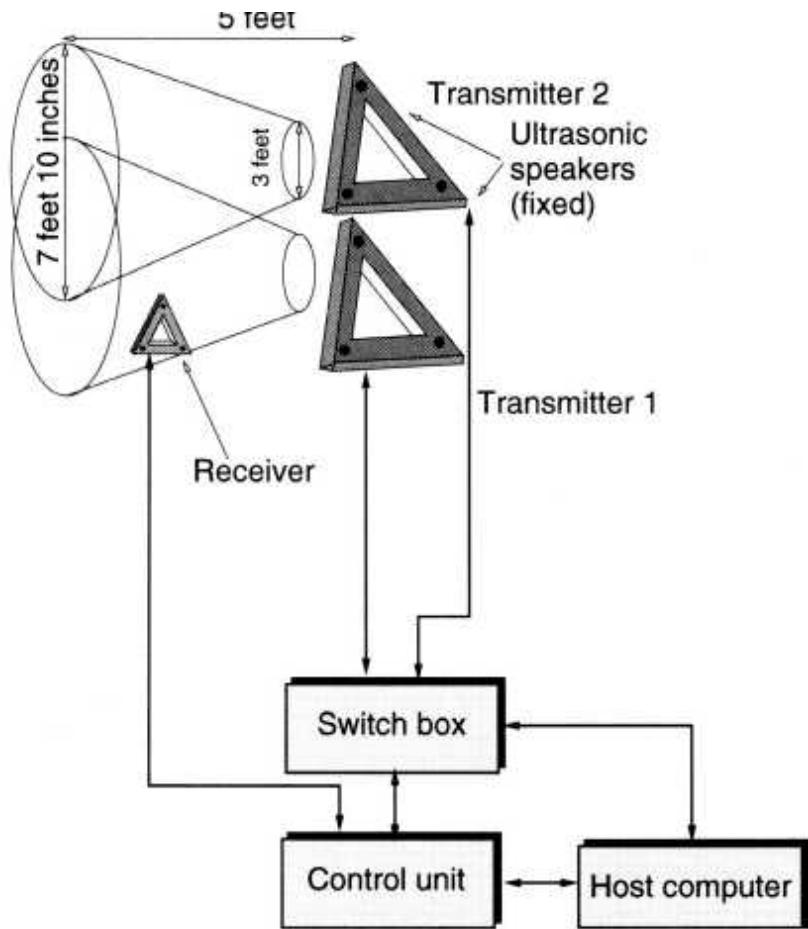


Fig. 2.15 Ultrasound tracking over large distances. Adapted from Sowizral and Barnes [1993]. © 1993 IEEE. Reprinted by permission.

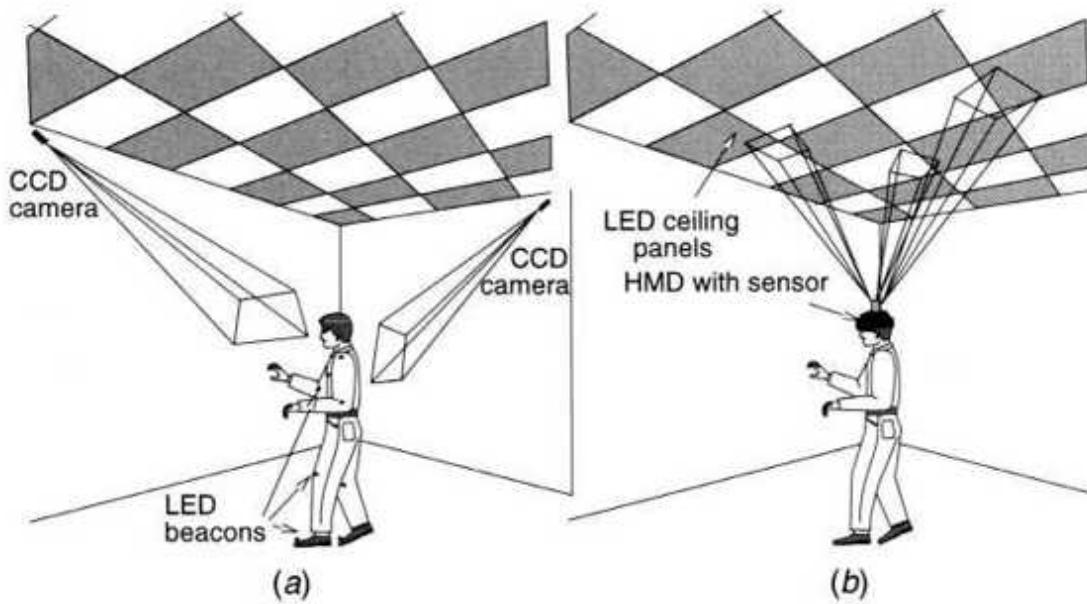


Fig. 2.16 Optical tracker arrangement: (a) outside-looking-in; (b) inside-looking-out. Adapted from Welch et al. [2001]. © 2001 Massachusetts Institute of Technology. Reprinted by permission.

2.1.5 Optical Trackers

Definition An optical tracker is a noncontact position measurement device that uses optical sensing to determine the real-time position/orientation of an object.

Similar to ultrasonic trackers, optical trackers work through triangulation, require direct line of sight, and are immune to metal interference. Optical trackers, however, offer significant advantages over their ultrasonic counterparts. Their update rates are much higher and their latency smaller than those of ultrasonic trackers because light (whether visible or infrared) travels much faster than sound. They are also capable of (much) larger work envelopes, which is increasingly important in modern VR systems.

If the tracker sensing component (charge-coupled device [CCD] camera, photodiode, or other photo sensor) is fixed and some light beacons are placed on the user, the tracker is said to be outside-looking-in, as illustrated in Figure 2.16a [Welch et al., 2001]. Position measurements are done directly, and orientation is inferred from the position data. Tracking

sensitivity is degraded as the distance decreases between the beacons on the user's body and the distance increases between the user and the camera. By contrast, an inside-looking-out optical tracker has the camera(s) attached to the tracked object or user, as shown in Figure 2.16b. Its sensitivity is maximized for changes in orientation (very useful for HMD tracking), and the work envelope can be scaled theoretically at infinity (very useful for wall or room-type graphics displays).

Traditionally, optical trackers have been outside-looking-in, and used primarily in motion capture for animation creation and biomechanics rather than for VR. The SIMI Motion Capture 3D [SIMI, 1999], for example, uses up to six fixed cameras to measure the position of up to 100 points (light-emitting diode [LED] "landmarks" worn by actors). Data are recorded in real time, and offline processing inserts character actions in video sequences, for games, TV shows, and other simulations.

An inside-looking-out optical tracker system designed for use primarily in VR is the laserBIRD produced by Ascension Technology Co. [Ascension, 2001a]. As illustrated in Figure 2.17, the tracker consists of a fixed laser scanner with dual rotary beams and a mobile triangular sensing element that can be attached to the tracked object. The laser scanners transmit two infrared (IR) modulated and offset light planes each, which intersect the sensing element. An imbedded microprocessor in the scanner controls the timing of the laser scanners such that the scanning angles of the light planes detected as intersecting any of the sensors S₁, S₂, S₃ are given by

$$a_1 = wL t_1 \quad (2.4)$$

$$a_2 = wA t_2 \quad (2.5)$$

where w is the scanner rotational speed.

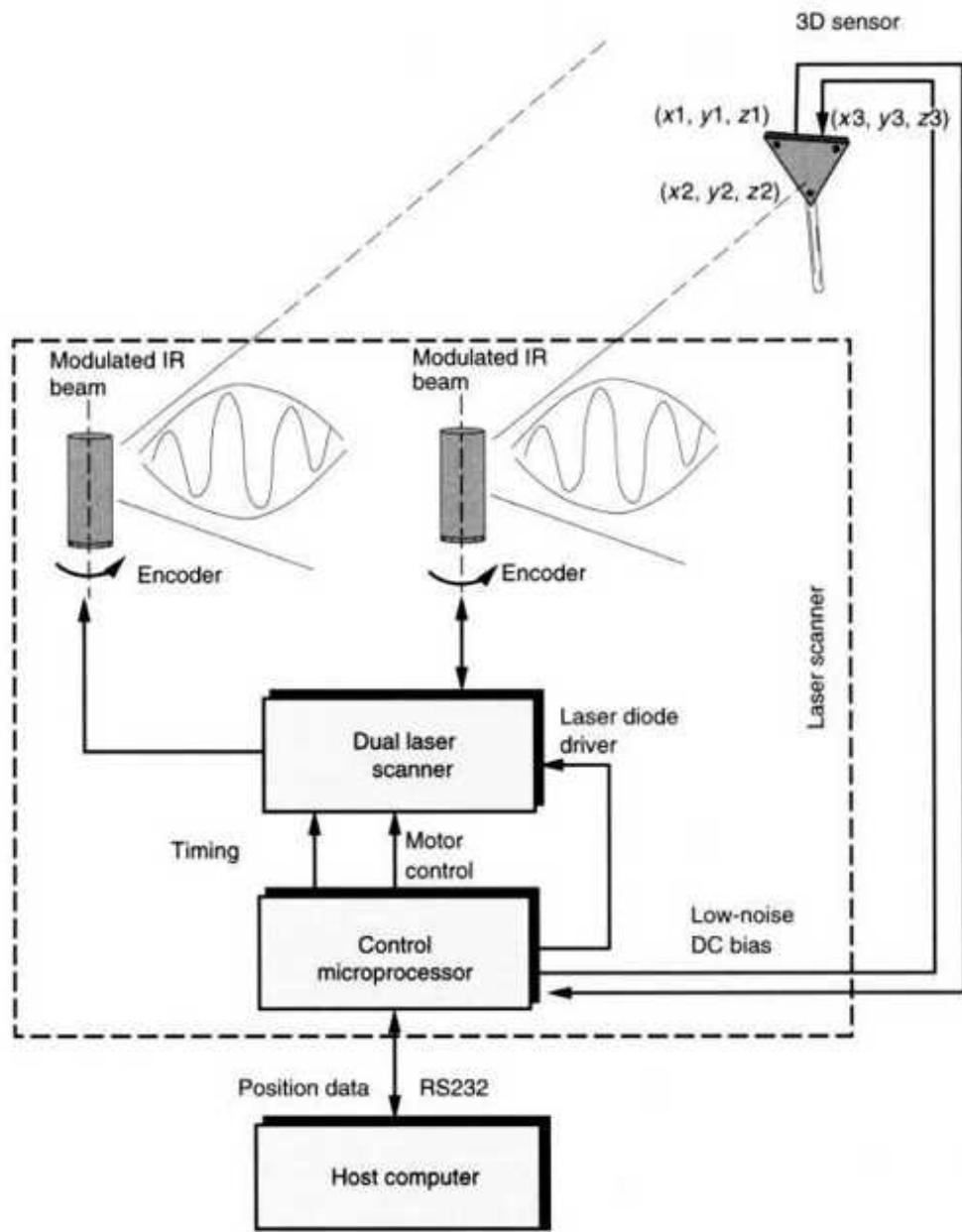


Fig. 2.17 Inside-looking-out laserBiRD optical tracker. Adapted from Hansen [1998]. © Ascension Technology Co. Reprinted by permission.

The positions of the three sensors with regard to the scanner are computed by the microprocessor based on the known geometry of their triangular arrangement as well as the known offset of the two laser scanning heads and the angular offsets of the IR beams they produce. Subsequently the position of the center of the triangular sensor is computed as the average of the triads (x_1, y_1, z_1) , (x_2, y_2, z_2) , and (x_3, y_3, z_3) [Hansen, 1998]. Complete 3D

datasets are then transmitted to the host computer at a rate of 240 updates/sec over an RS232 line (115 kbaud). The operational range of the laserBIRD tracker is up to 2 m from the fixed laser scanner, with a positional accuracy of 1.0 mm root mean square (RMS) and an angular accuracy of 0.5° RMS. The tracker accuracy is comparable to, or better than, that of magnetic trackers, with better latency (less than 7 msec) and twice the refresh rate of the Fastrack. Unlike other optical and hybrid trackers, this sensor is small (5.7 cm x 5.1 cm x 1.1 cm) and light (40 grams), making it comfortable to wear with HMDs.

Another example of inside-looking-out tracker is the HiBall 3000 produced by 3rdTech (Chapel Hill, NC) [Welch et al., 2001]. As illustrated in Figure 2.18, this optical tracking system consists of up to two HiBall units, LED beacon array(s), a ceiling-HiBall Interface, and a host computer. The HiBall sensing unit incorporates IR-filtering lenses, lateral-effect photodiodes (LEPDs), and miniaturized electronic circuits. There are six narrow-view (about 6°) lenses arranged in the six sectors of a hemisphere. Their view is projected onto six corresponding LEPDs, such that one LEPD can receive views from several lenses. The LEPD determines the x-y coordinates of a light spot that is the image of a pulsed LED on the beacon array. The analog signal is amplified, multiplexed, and converted to a digital value, which is then sent to the ceiling interface board (CIB) using an on-board modem and wires.

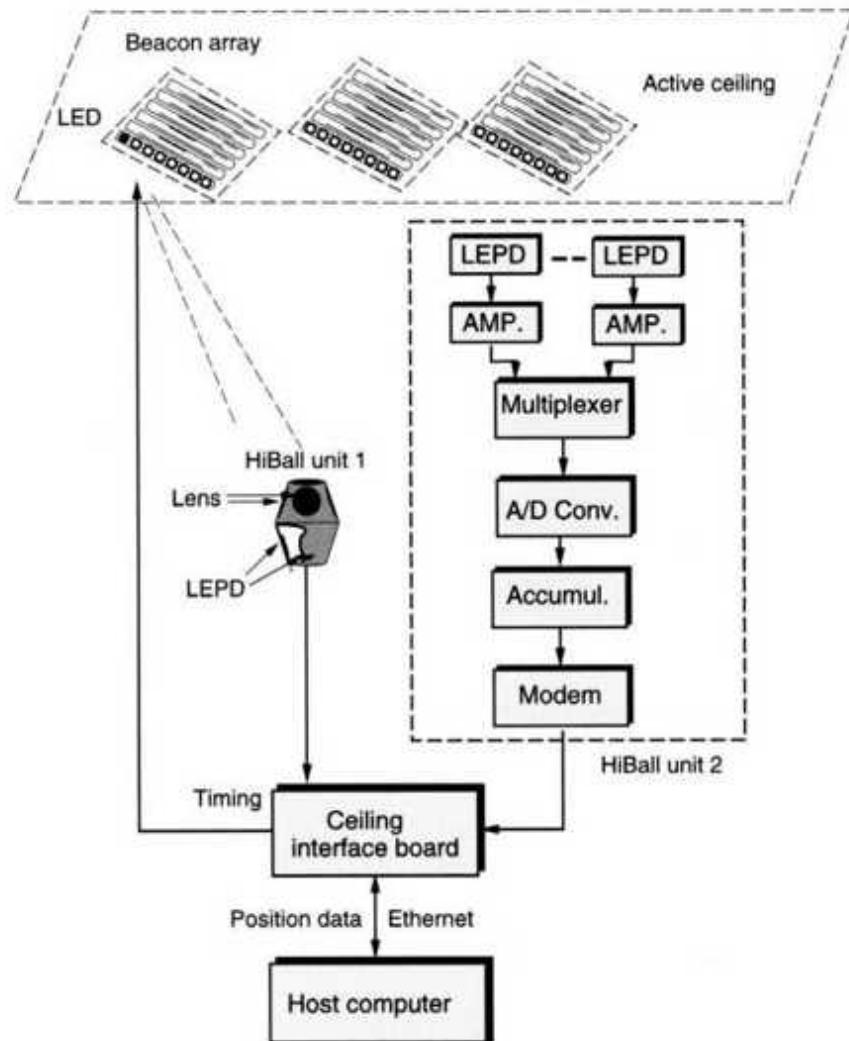


Fig. 2.18 The HiBall 3000 optical tracker. Adapted from Welch et al. [2001]. © Massachusetts Institute of Technology. Reprinted by permission.

The beacon arrays consist of 48 LEDs arranged in six parallel strips over an 8-ft² surface modular with American false ceiling panels. A number of beacon arrays can be added to increase the tracking envelope of the HiBall from 8 x 8 ft to 40 x 40 ft (12.2 x 12.2 m). Each beacon array is connected to the CIB such that only one LED is pulsed at a given instant. When that LED is imaged on several of the LEPDs it is possible to determine distance through triangulation because the geometries of the HiBall and of the beacon array being pulsed are known.

The CIB improves processing speed and tracker accuracy by implementing a single-constraint-at-a-time algorithm, which uses Kalman filtering to compute the distance to an LED without attempting to do a complete 3D position measurement of the HiBall. These partial data are immediately combined with previous measurements to get the position and orientation of the HiBall, and the more the algorithm runs, the more accuracy and tolerance to ceiling imperfection is obtained. In the current configuration a single HiBall update rate is 2000 datasets/sec, which is an order of magnitude more than any other tracker discussed so far. When two HiBall sensing elements are daisy-chained, each delivers 1000 such 3D data sets every second, with an extremely small latency of less than 1 msec. The accuracy of this optical tracker is 0.5 mm RMS for position and 0.030 for orientation, which is maintained throughout the tracking surface. The weight of the optical sensor (about 300 grams or 11 oz) does raise an ergonomic concern, as do long wires that are still needed, since this version is not wireless. The research group at the University of North Carolina at Chapel Hill, which did the work that resulted in the HiBall, plans to convert the sensor to wireless operation and replace the six narrow-view lenses with fewer wider view ones. This should reduce the dimensions (and weight) of the sensing unit.

2.1.6 Hybrid Inertial Trackers

Definition Inertial trackers are self-contained sensors that measure the rate of change in an object orientation. They may also measure the rate of change of an object translation velocity.

Modern inertial trackers are solid-state structures that use microelectromechanical systems (MEMS) technology. The rate of change in object orientation, or angular velocity, is measured by Coriolis-type gyroscopes. Three such gyroscopes are machined on mutually orthogonal axes, measuring yaw, pitch, and roll angular velocities ω . The orientation angle about the three orthogonal axes is then determined through integration over time. Inertial trackers measure the rate of change in translation velocity, or acceleration, using solid-state accelerometers. Three accelerometers machined coaxially with the three gyroscopes are needed to measure body-

referenced accelerations. Knowing the tracked object orientation (from gyroscopic data) and subtracting gravitational acceleration allows the computation of accelerations in world coordinates. The tracked object position is finally obtained through double integration over time and knowledge of starting position (calibration). Inertial trackers offer the advantage of sourceless operation with theoretically unlimited range, no line-of-sight constraints, and very low jitter (sensor noise). Whatever jitter exists is further filtered through integration. Therefore no additional time-consuming filtering is needed, with beneficial effects on reducing latency.

Inertial trackers have a significant drawback, namely rapidly accumulating errors, or drift. Any gyroscope bias A_w leads to an orientation error that increases proportionally with time due to integration. Accelerometer bias A_a in turn induces an error that increases with the square of time. The problem is compounded by the use of gyroscope (drifted) data in computing position. Therefore Foxlin [2002] estimated that position drift can grow to as much as 40 mm in 2 sec for commercial-grade inertial sensors. The much more expensive strategic-grade inertial trackers have the same drift after 200 sec, a time interval that is still too short for use in VR. The answer to the drift problem is to use data from other types of trackers to periodically reset the output of inertial ones.

Definition A hybrid tracker is a system that utilizes two or more position measurement technologies to track objects better than any single technology would allow.

When only orientation data are needed, such as for low-cost HMDs, then one solution is to add solid-state magnetometers aligned with the three gyroscopes. Data from the three magnetometers can be used to determine the local magnetic North, which is unchanged regardless of the user's head orientation. This compensates for drift in the yaw (or azimuth) measurement, and is used by the Ascension Technology 3D Bird [Hansen and Kogan, 1999] and the InterSense InterTrax2 [InterSense, 2000a] trackers. The 3D Bird has an accuracy of 4° , an update rate of 160 datasets/sec, and a latency of 15 msec. The InterTrax2 has an accuracy of 5° , an update rate of 256 datasets/sec, and a latency of 4 msec. Both trackers do processing on-board,

such that a single RS232 line is needed to connect to the host computer. Any disturbance of Earth's local DC magnetic field (such as a large metal) will impact the performance because it affects the reading on the magnetometers used to compensate drift.

Inertial trackers that provide both orientation and position data need additional means to compensate for drift. An example is the InterSense IS-900 ultrasonic-inertial tracker [Foxlin et al., 1998], illustrated in Figure 2.19.

In order to limit drift due to misalignment, all gyroscopes and accelerometers are machined out of a single cube, called the InertiaCube. Other monolithic shapes are used for integration into a stylus (discussed later in Chapter 5). The InertiaCube is combined with a number of ultrasonic rangefinder modules (URMs) to form a tracking station. The URM contains an infrared LED, omnidirectional microphone, and related electronics (time-of-arrival detector and counter). The counter is started when the IR LED sends a signal to an array of ceiling-mounted ultrasonic speakers, or sonic disks. Each sonic disk is self-contained (battery-operated) and has an IR sensor and electronics to decode the IR LED signal. If a particular sonic disk-IR code is detected, then that sonic disk emits a 40-kHz ultrasound. The measured URM counter data are sent to an interface electronic unit, where they are used to determine range. The interface CPU merges (fuses) range with data coming from the InertiaCube to obtain reliable 3D position/orientation.

The way the ultrasonic range data are fused with the inertial gyroscopic and accelerometer data is illustrated in Figure 2.20 [Foxlin et al., 1998]. The tracking algorithm first uses integration, and in the case of accelerometers, double integration, to get orientation and position data to the host computer. This direct output insures overall low latency of the hybrid tracker. The output data are fed back into an estimation Kalman filter (EKF), where they are compared with data from the ultrasonic range measurement, in the same single-constraint-at-a-time (SCAAT) fashion as in the HiBall. The EKF then estimates the amount of drift and resets the integration processes. Range data are rejected if background noise corrupts the URM

data (premature counter stop). Using this hybrid tracking approach, the IS-900 Large Area Tracker (LAT) model can track over a surface from 6 x 6 m² to 900 m² with an accuracy of 4 mm and 0.2°. Its update rate depends on how many trackers are used simultaneously, dropping from 180 datasets/sec for two tracking stations to 90 datasets/sec when four trackers are used simultaneously. This drop in update rates is due to the waiting time imposed by ultrasonic range tracking used in drift correction and tracker initialization. Data are sent to the host computer over an RS232 line with 4-10 msec latency [InterSense, 2000b].

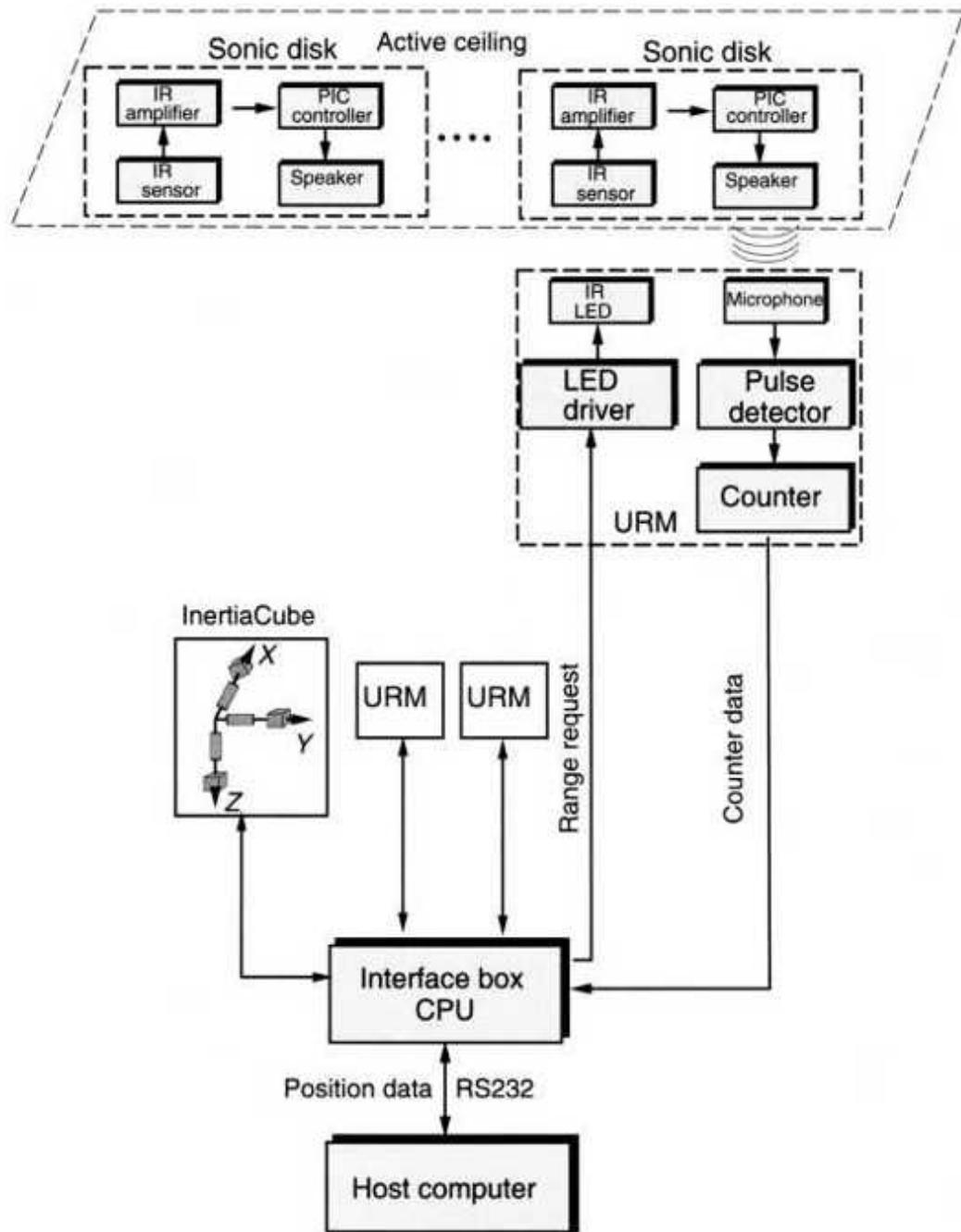
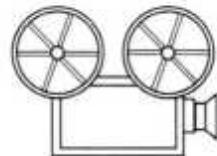


Fig. 2.19 The InterSense Inc. IS-900 block diagram. Adapted from Foxlin et al. [1998]. © 1998 ACM Inc. Reprinted by permission.



VC 2.1

Table 2.2 shows a performance comparison of the various trackers discussed in this chapter, based on their accuracy, range, latency, and update rate. They are ordered with the best performance in each category at the top and the worst at the bottom. Update rates are given for the best case, which usually is for a single sensing element.

2.2 NAVIGATION AND MANIPULATION INTERFACES

Definition A navigation/manipulation interface is a device that allows the interactive change of the view to the virtual environment and exploration through the selection and manipulation of a virtual object of interest.

The navigation/manipulation can be done in either absolute coordinates or relative coordinates. The trackers described so far are absolute, as they return the position and orientation of a moving object with respect to a fixed system of coordinates. The position of the VR object controlled on the screen is directly mapped to the absolute position of the receiver in the world (transmitter)-fixed system of coordinates.

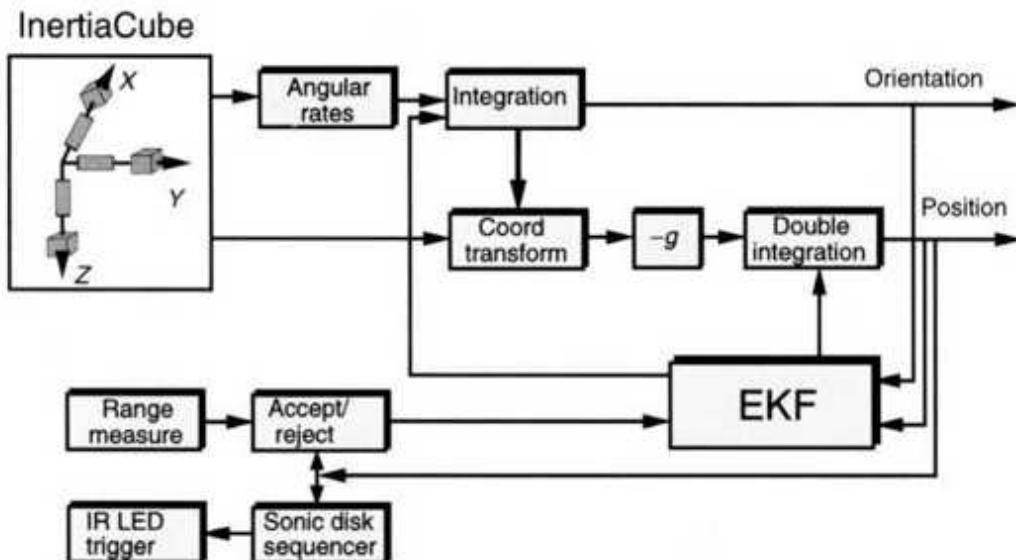


Fig. 2.20 The InterSense Inc. IS-900 software block diagram. Adapted from Foxlin et al. [1998]. © 1998 ACM Inc. Reprinted by permission.

TABLE 2.2. Performance Comparison of Various Trackers^a

Accuracy (mm/deg)	Range (m)	Latency (sec × 10 ⁻³)	Update Rate ^b (datasets/sec)
0.5/0.03	30 × 30	0.0002	2000
HiBall	IS-900	Push	HiBall
0.8/0.15	12.2 × 12.2	1	256
Fastrack	HiBall	HiBall	InterTrax2
1/0.5	2	4	240
laserBIRD	laserBIRD	InterTrax2	laserBIRD
2/0.5	1.52	7	180
Flock of Birds	Logitech	laserBIRD	IS-900
4/0.2	1.2	7.5	160
IS-900	Flock of Birds	Flock of Birds	3-D BIRD
4/NA	0.75	8.5	144
Push	Fastrack	Fastrack	Flock of Birds
NA/4	NA	10	120
3D BIRD	3D BIRD	IS-900	Fastrack
NA/5	NA	15	70
InterTrax2	InterTrax2	3D BIRD	Push
30	NA	30	50
Logitech	Push	Logitech	Logitech

^aFrom top to bottom, best to worst performance. NA, Not available.

^bFor a single sensing element.

Another way to control a VR object's position is through relative sensors. Whereas absolute position data are never a zero set (even if the receiver is at rest), a relative position sensor will always return zeros if not acted upon. Navigation/manipulation in relative coordinates allows for incremental position control relative to the object's previous 3D position. The position of the VR object is incremented by six signed quantities at every simulation cycle. Velocity is indirectly controlled, as a larger translation/rotation increment will result in a larger velocity of the VR object in the simulation.

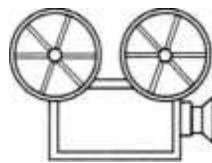
2.2.1 Tracker-Based Navigation/Manipulation Interfaces

Trackers offer more functionality to VR simulations than simply measuring the realtime position/orientation of the user's hand and head. Integrated

within a structure that houses user-programmable pushbuttons, trackers become navigation and manipulation interfaces. Examples are the Polhemus 3Ball and the Ascension Technology 3D Mouse [Anon, 1998]. The 3Ball is a hollow billiard ball that houses a tracker receiver inside and has a pushbutton on the surface. An example of its use is to move a virtual camera that travels along the 3D vector controlled by the ball as long as the pushbutton is pressed. Another example is a wand, which projects a virtual ray along the tracker-controlled vector. Any object intersected by this ray can be selected by pushing the button on the billiard ball, and then repositioned through wrist motion. The 3D Mouse offers more functionality, as it can work both as a mouse (in 2D) and as a navigation/manipulation interface in 3D, similar to the 3Ball. Further functionality is obtained through the programming of the three buttons incorporated on the 3D Mouse.

One drawback of absolute position control of virtual objects is the user's limited arm reach. The larger the dimensions of the graphics display, the more this becomes a problem. One way to solve the disparity between the user's arm reach and the (larger) display dimensions is to multiply the tracker readings with a constant (gain) larger than unity. The drawback is that tracker noise is equally amplified, and selected objects appear more jittery than they would otherwise be. An alternative is to use indexed motions, in which virtual objects are selected and deselected repeatedly. When the object of interest is deselected, it stays in place, while the user moves his or her arm back and then reselects the object and moves it using the tracker, and so on.

An example of a manipulation interface that uses this indexed motion is the CubicMouse, illustrated in Figure 2.21 [Frohlich et al., 2000]. It consists of a plastic cubic structure of dimensions 9 cm x 9 cm x 9 cm (3 5/8 in. x 3 5/8 in. x 3 5/8 in.). This plastic cube houses a Polhemus Fastrack, three mutually perpendicular translating rods, and six application-programmable control buttons. The buttons are placed on the face that normally is oriented upward (the "up" face), while wires from the tracker, the sensors embedded in the translating rods, and the buttons exit through the opposite ("down") face.



VC 2.2

As opposed to wands that are designed for flyby navigation, the Cubic-Mouse is designed primarily for the manipulation of a single, large virtual model. Navigation in this context means rotating and translating the virtual model through the tracker and dissecting/editing it using the three rods. Each of the rods controls a cutting plane, and buttons at the end of the rods allow for selecting of the orientation of the cutting planes. Further functionality is brought by the control buttons, one being dedicated to indexed motion (the clutch button) and two to zooming in-out on the virtual model. As opposed to classical computer mice, the Cubic-Mouse is designed for dual-hand operation. Users hold the mouse with their nondominant hand and push the rods and buttons with their dominant one. Proprioception (the body's own sense of position/motion) and tactile cues allow operation without looking at the rods.

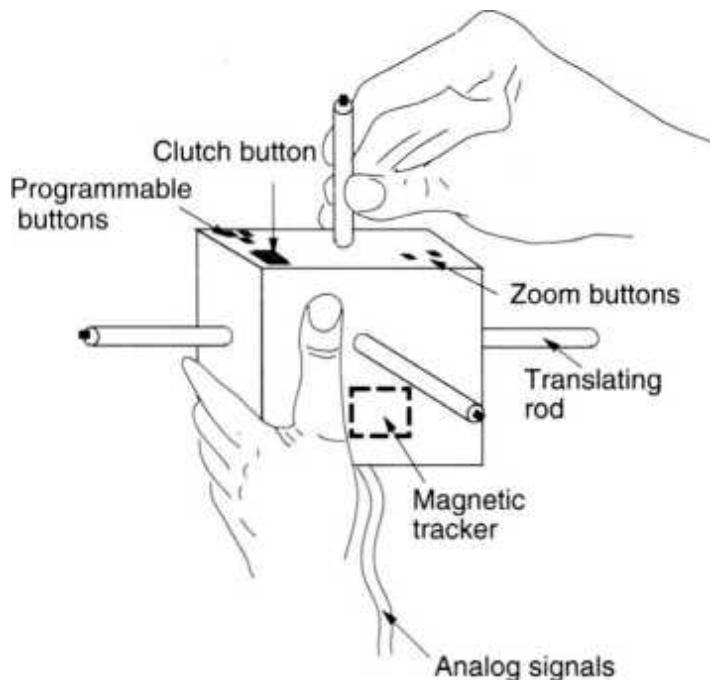


Fig. 2.21 The Cubic-Mouse. Adapted from Frohlich et al. [2000]. © 2000 IEEE. Reprinted by permission.

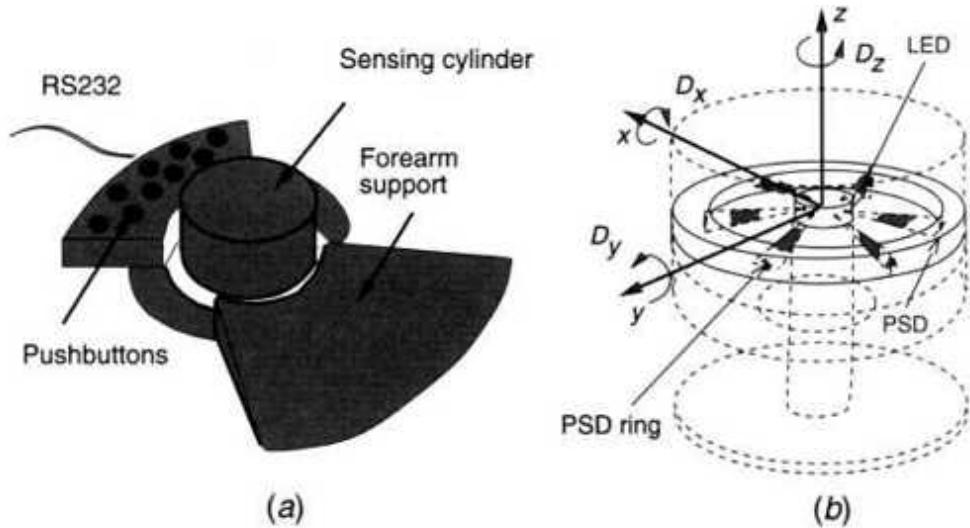


Fig. 2.22 The Magellan trackball: (a) outside configuration; (b) the sensing mechanism. Adapted from Hirzinger and Bejczy [1988]. Reprinted by permission.

2.2.2 Trackballs

A class of interfaces that allow navigation/manipulation in relative coordinates are trackballs, such as the Logitech Magellan shown in Figure 2.22a [Anon, 1998]. This is a sensorized cylinder that measures three forces and three torques applied by the user's hand on a compliant element. Forces and torques are measured indirectly based on the spring deformation law. The central part of the cylinder is fixed and has six light-emitting diodes, as shown in Figure 2.22b [Hirzinger and Bejczy, 1988]. Correspondingly, six photo sensors are placed on a moving external cylinder. When the user applies forces or torques on the moving shell, the photo sensor output is used to measure three forces and three torques. These forces and torques are then sent to a host computer over an RS232 serial line. Here they are multiplied by software gains to return a differential change in the controlled object position and orientation. Larger gains will result in larger speeds for the VR object the user controls, but its motion will not be smooth if the host cannot refresh the screen fast enough. An alternate way to control VR objects is through force control, where forces measured by the trackball are used to control forces applied by the VR object on the simulated environment. The trackball can also be used to fly by in the simulation. In

that case the sensor affects the velocity and orientation of a virtual camera looking at the simulated world.

Several pushbuttons are integrated with the trackball support, within reach of the user's fingers. These buttons are binary on/off and can be programmed according to the application. For example, one button can be used to increase the simulation gains (or VR object speed), and another button to couple and decouple the trackball with the VR object it controls. Finally, some buttons may be used to start and stop the simulation, or to reset the simulation to a default start location, if the user becomes disoriented.

Trackballs suffer from sensor coupling. Although the user may wish to have the VR object translate, but not rotate, the VR object may do both translations and rotations. This is due to nonzero sensed torques when pure forces are applied by the user's fingers on the cylinder. These unwanted motions can be suppressed with software filters that only read forces and not torques, or by hardware filters. These hardware filters are pushbuttons (available on some trackballs) that allow users to select translation-only, rotation-only, or dominant-motion input.

2.2.3 Three-Dimensional Probes

Users felt a need for an I/O device that would be intuitive to use, inexpensive, and allow either absolute or relative position control of the simulation. One such device is the Immersion Probe produced by Immersion Co. in the early 1990s. This was later renamed the MicroScribe 3D and its use extended to include digitizing objects [Rosenberg et al., 2000]. It consists of a small, sensorized mechanical arm that sits on a support base, with a small 6 in. x 6 in. footprint. The probe has six joints (joints 0-5), as illustrated in Figure 2.23.

Each rotary joint represents one degree of freedom, and thus the probe has six degrees of freedom, allowing simultaneous positioning and orienting of its tip. A counterbalance is placed close to the base to minimize the user's fatigue. The tip position relative to the base is obtained through direct kinematics calculations, based on sensor values and the length of the links. Software on the host computer reads the joint sensors on an RS232 line, then

uses its kinematic model to determine where the tip is. A binary switch on a foot pedal is used to select/deselect (release) virtual objects, navigate (start/stop), or mark a point on the real object surface for digitization purposes. Alternatively, the foot pedal may also be used to control the flying speed in the virtual scene. The direction of the velocity vector is then given by the probe tip (last link) orientation.

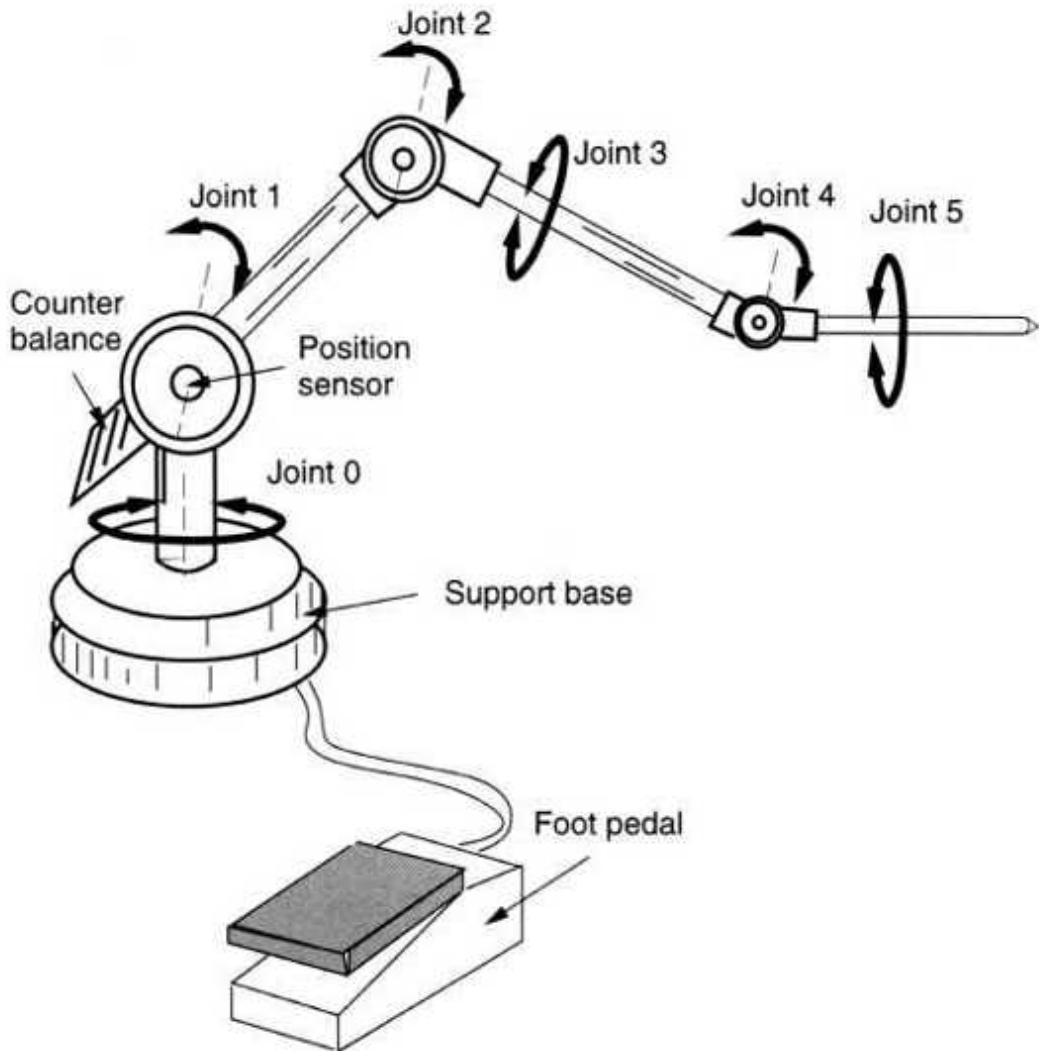


Fig. 2.23 The Immersion Co. Microscribe 3D. Adapted from Rosenberg et al. [2000]. © 2000 Immersion Co. Reprinted by permission.

The MicroScribe 3D workspace depends on its model, ranging from a 50-in.-radius hemisphere (models 3D and 3DX) to a 66-in.-radius hemisphere (models 3DL and 3DLX). These measurements are not affected by magnetic

fields, as the probe is a mechanical structure. It is thus possible to measure over 1000 positions/sec with minimal latency. The difference between the probe's actual and measured positions and orientations represents its accuracy. Errors will occur owing to the joint sensor resolution, temperature variations, or joint tolerances. Unfortunately, all open kinematic chains, including the probe, accumulate errors from the base toward the tip. The larger the length of the particular MicroScribe model, the less accurate it is. To minimize errors the MicroScribe needs calibration before use. Calibration is done by placing the probe tip in a housing close to the base and reading the rotary sensor data. These readings are then compared with stored values corresponding to the joint angles when the probe arm is in this known position. Model 3DX, which uses high-resolution joint sensors, has a tip accuracy of 0.23 mm (0.009 in.). The larger model 3DLX is slightly less accurate (0.30 mm, or 0.012 in.).

2.3 GESTURE INTERFACES

Trackballs and 3D probes have the advantage of simplicity, compactness, and quiet operation. By their nature they limit the freedom of motion of the user's hand to a small area close to the desk. Therefore the natural motion of the user's hand is sacrificed, and interaction with the virtual world is less intuitive. In order to have large gesture-based interactions with the VR simulation, it is necessary that the I/O devices maintain the hand freedom of motion within a certain volume. It is also desirable to allow additional degrees of freedom by sensing individual finger motions. The human fingers have degrees of freedom associated with flexion-extension and lateral abduction-adduction, as illustrated in Figure 2.24 [American Society for Surgery of the Hand, 1983]. Additionally, the thumb has an anteposition-retroposition motion, which brings it in opposition to the palm.

Definition Gesture interfaces are devices that measure the real-time position of the user's fingers (and sometimes wrist) in order to allow natural, gesture-recognitionbased interaction with the virtual environment.

Most gesture interfaces today are sensing gloves that have embedded sensors which measure the position of each finger versus the palm. Sensing

gloves differ in such factors as, for example, the type of sensors they use, the number of sensors for each finger (one or several), their sensor resolution, the glove sampling rate, and whether they are tethered or wireless.

Some of the available commercial sensing gloves are the Fakespace Pinch Glove [McDowall et al., 2000], the Fifth Dimension Technology 5DT Data Glove [Fifth Dimension, 2000], the Didjiglove [Anon, 2000], and the Immersion CyberGlove [Immersion Co., 2001]. They have sensors that measure some (or all) of the finger joint angles. Some have built-in trackers as well, in order to measure the user's wrist motion. The resulting sensing glove work envelope is much larger than that of trackballs or joysticks, as illustrated in Figure 2.25. As opposed to trackballs and 3D probes, which have single-point interaction with the virtual environment, sensing gloves allow dexterous, multipoint interaction at the fingertips or palm. This results in a more realistic simulation, especially for object manipulation tasks. Additionally, sensing gloves can become navigation interfaces, based on user-programmed gesture libraries [Laviola, 2000].

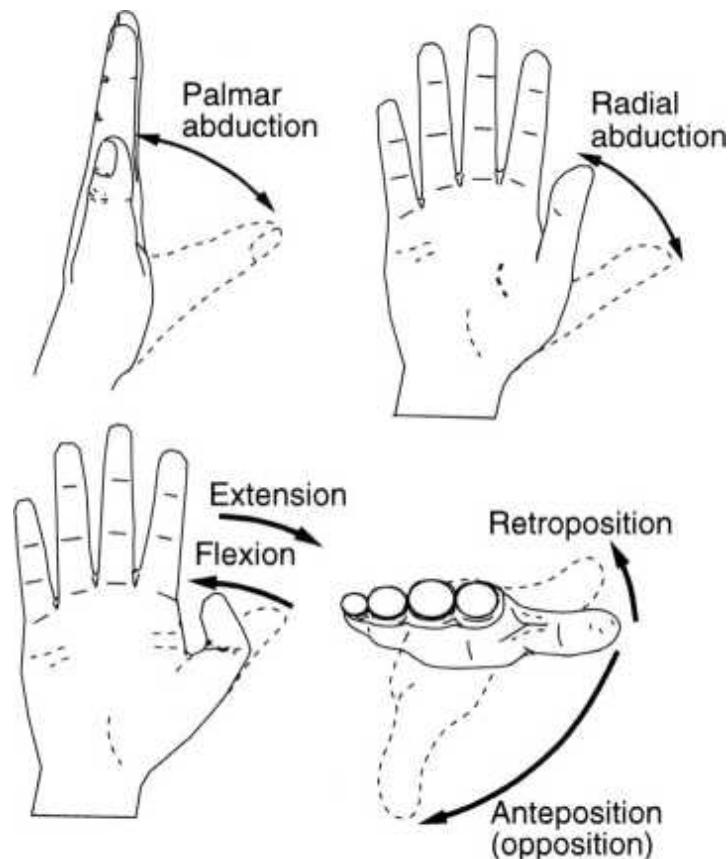


Fig. 2.24 Terminology of hand and finger motions. Adapted from American Society for Surgery of the Hand [1983]. ©1983 Elsevier Science. Reprinted by permission.

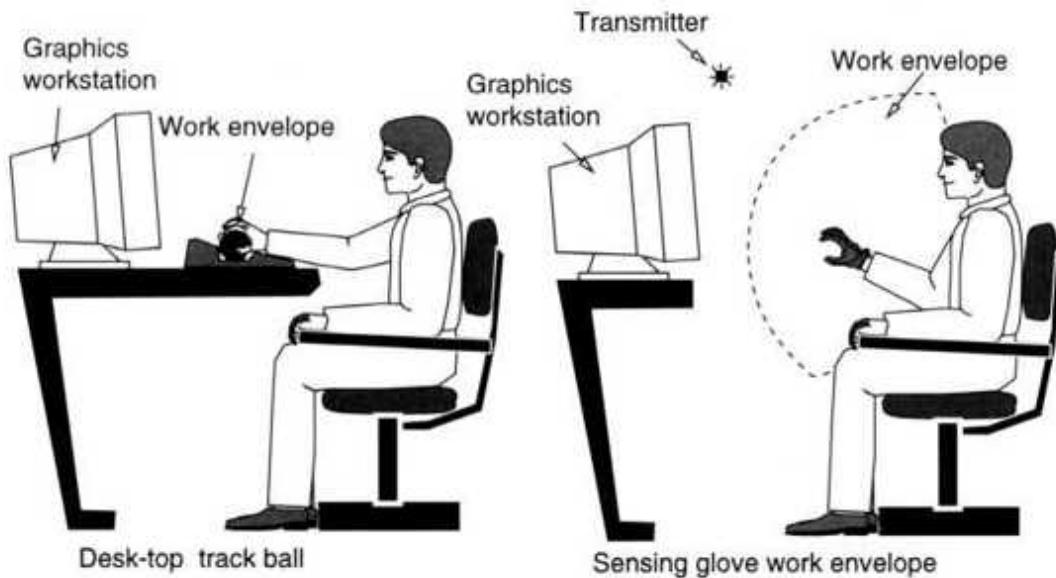


Fig. 2.25 Comparison of sensing glove work envelope and trackball work envelope. From Burdea [1993]. Reprinted by permission.

2.3.1 The Pinch Glove

The drawbacks that most sensing gloves have are need for user-specific calibration, complexity, and high cost. Each person has a different hand size, with women generally having smaller hand size than men. As a consequence, the glove-embedded sensors will overlap different finger locations for different users. In order to reduce inaccuracies, most sensing gloves need to be calibrated to the particular user wearing them. Users have to place their hands in predetermined gestures (such as a flat hand or a fist) and the sensor output measured. These raw values are then converted to finger joint angles based on glove-specific algorithms.

The only commercial sensing glove that makes calibration unnecessary is the Pinch Glove, which is illustrated in Figure 2.26 [McDowall et al., 2000]. The glove incorporates electrodes in the form of conductive fiber patches at the fingertips, on the back of fingers, or in the palm. Gestures are

positively detected as the establishing and breaking of electrical contacts between the fingers of one hand, fingers of one hand and fingers of the other hand, fingers and palm, etc. A multiplexing chip embedded in the glove reduces the number of wires that need to be connected to an electronics control box. The control box incorporates a microprocessor, low-current power supply, timing circuits, and RS232 serial port for communication with the host computer.

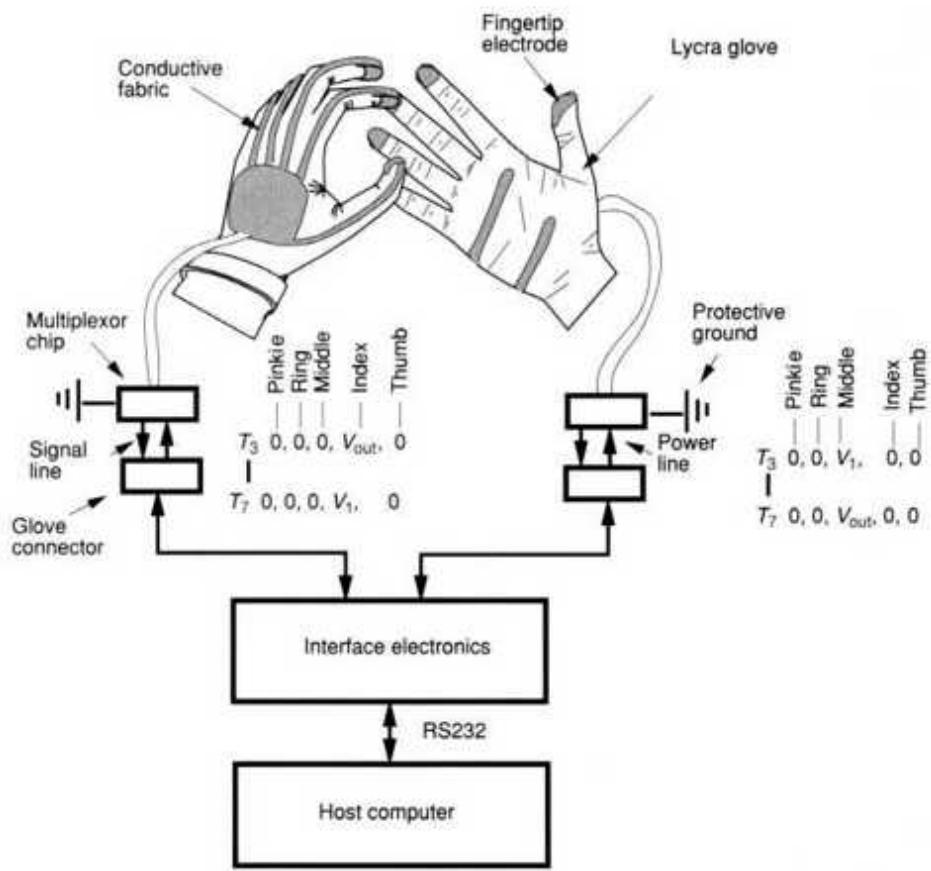


Fig. 2.26 The Pinch Glove block diagram. Adapted from McDowall et al. [2000]. Reprinted by permission of Fakespace Labs, Inc.

The Pinch Glove interface detects finger contact by applying a polling algorithm in which each finger in turn receives a voltage V_i and the interface looks for output voltages on the other fingers. At time T_1 the thumb of the right hand is energized. Since it makes no contact with any finger, all the other fingers of the right hand, as well as the fingers of the left hand, will

show 0 V on their output. At time T2 the index of the right hand is energized, and again no voltages are detected on the other fingers. At time T3 the middle finger of the right hand receives voltage V₁ and the interface detects a voltage V_{,t} on the index of the left hand, meaning the two fingers are in contact. Subsequently the right ring, right pinkie, and left thumb are energized and no contact detected. At time T7 the left index receives voltage V₁ and the corresponding V_{,t} is detected on the right middle finger, meaning the two fingers are still in contact. The voltage sequencing is done at high frequency, such that contacts are detected even if they only occur for short durations. The interface box keeps a record of a gesture duration, such that compound gesture sequences (such as double clicking) can be reported as a single gesture to the host computer. This alleviates the problem that some Unix hosts have in keeping real-time tags and makes the Pinch Glove usable across platforms.

The Pinch Glove has numerous advantages in terms of its simplicity, lack of a need for calibration, confirmation of gestures through hand haptic sensing, and possibility to use both hands for gesture interaction. However, the glove can only detect whether a contact is made or not, and cannot measure intermediary finger configurations. Virtual hands controlled by a Pinch Glove will not follow in real time the true position of the user's fingers, with a negative effect on simulation realism.

2.3.2 The 5DT Data Glove

In order to detect incremental changes in the user's finger configuration, sensing gloves need to measure the finger joint angles over their full range of motion. The number of sensors depends on whether each joint is measured separately or not. The simplest configuration, used by the 5DT Data Glove 5W sensing glove illustrated in Figure 2.27, is to have one sensor per finger and a tilt sensor to measure wrist orientation.

Each finger has a fiber loop routed through attachments which allow for small translations due to finger bending. Additional sensors for minor joints as well as abduction-adduction are available in the 5DT Data Glove 16 option. The advantage of fiber-optic sensors is their compactness and lightness, and users feel very comfortable wearing the glove. The optical

fibers are joined to an optoelectronic connector on the back of the hand. One end of each fiber loop is connected to an LED, while light returning from the other end is sensed by a phototransistor. When the fiber is straight, there is no attenuation in the transmitted light, as the index of refraction of the cylindrical walls is less than the refractive index of the core material. The fiber walls are treated to change their index of refraction such that the light will escape upon finger flexion. In this way the glove measures the finger bending indirectly based on the intensity of the returned light.

The sensors used by the 5DT Data Glove 5 are similar to those incorporated in the much earlier VPL DataGlove. That glove, sold in the late 1980s, had a separate control box holding the optoelectronics circuitry, as well as analog-to-digital converters, multiplexers, and RS232 communication ports. Advances in electronics and miniaturization techniques allowed the shrinking of the electronics interface to the size of a small box placed on the user's wrist. Another technological advance is wireless communication, which significantly increases the user's work envelope to a hemisphere of 30 m radius centered on the wireless receiver. The wireless receiver gets 100 samples every second (fingers and tilt sensor data), which are then sent to the host computer running the simulation. Different frequencies are used for right- and left-hand gloves, in order to allow wireless communication without mutual interference.

The glove uses an 8-bit A/D converter, such that it has a resolution of 256 intermediate positions between a flat and a fisted hand. The glove raw sensor data raw,,a] needs first to be normalized, in order to account for the effect of varying hand sizes. This normalization is carried out using the equation

$$\text{out} = \text{rawval} - \text{rawmin} \times 255 / (\text{rawmax} - \text{rawmin}) \quad (2.6)$$

where rawmax - rawm;n is the dynamic range for a given flex sensor and a given user. Calibration is done by having the user flex their hand a few times, while the system updates the value of the glove dynamic range. This needs to be repeated every time the 5 DT Data Glove is put on at the start of a new simulation session. Once the glove is calibrated the corresponding

finger posture is obtained based on a look-up table or based on the knowledge that finger joints are naturally coupled. Thus the flexion angle θ_3 of the finger distal joint (the one furthest from the palm) is coupled to θ_2 [Burdea et al., 1992] by the formula

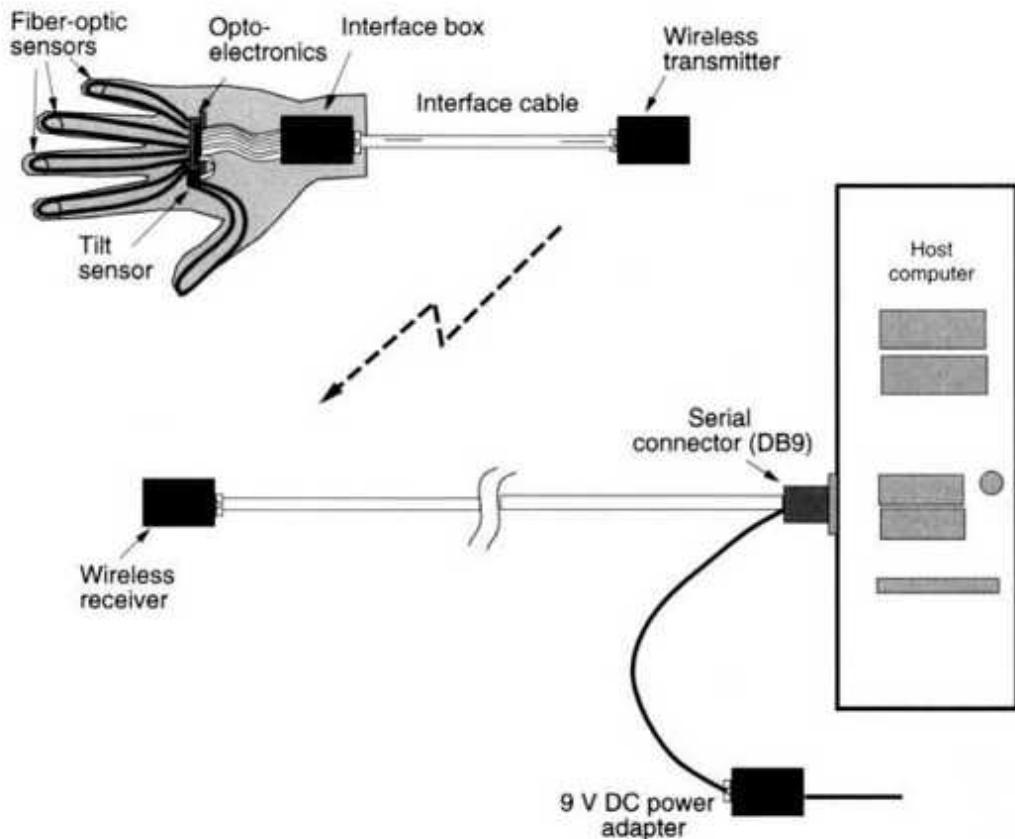


Fig. 2.27 The 5DT Data Glove 5W block diagram. Adapted from Fifth Dimension Technology Inc. [2000]. Reprinted by permission.

$$\theta_3 = a - b\theta_2 + c\theta_2^2 \quad (2.7)$$

where the parameters a , b , and c depend on the characteristics of each finger.

The 5DT Data Glove 5 software driver running on the host computer interprets the glove readings as gestures. The current gesture library uses binary open/close configurations for all fingers, except the thumb. This simplifies user learning to only 16(2^4) combinations, as illustrated in

Figure 2.28. Thus a finger is considered unflexed (opened) if its normalized flex sensor value is smaller than a predetermined threshold. Conversely, the finger is considered flexed (closed) if it is above an upper threshold. By keeping the lower and upper threshold values apart, the probability of erroneous gesture interpretation is reduced.

2.3.3 The Didjiglove

Another sensing glove is the Didjiglove, which uses 10 capacitive bend sensors to measure the position of the user's fingers [Anon, 2000]. The capacitive sensors consist of two layers of conductive polymer separated by a dielectric. Each conductive layer is arranged in a comblike fashion, such that the overlapping electrode surface is proportional to the amount of sensor bending [Neely and Restle, 1997]. Since capacitance is directly proportional to the overlapping surface of the two sensor electrodes, the bending angle can be measured electrically.



Fig. 2.28 The 5DT Data Glove 5 gesture library. From Fifth Dimension Technology Inc. [2000]. Reprinted by permission.

The Didjiglove interface is located on the user's cuff, similar to the 5DT Data Glove. It has an A/D converter, a multiplexer, a processor, and an RS232 line for communication with the host computer. The 10-bit A/D converter resolution is 1024 positions for the proximal joint (closest to the palm) and the interphalangeal joint (the intermediate joint of the finger). Calibration is done similar to the 5DT Data Glove, by reading the sensor values when the user keeps the fingers extended (value set to 0) and when the fingers are bent (value set to 1023). The Didjiglove was designed as an advance programming interface for computer animation, for user input to such toolkits as 3D Studio Max, Softimage, and Maya [Tehrani, 1999]. However, the small glove latency (10 msec) and its low cost make the Didjiglove useful for VR interactions as well.

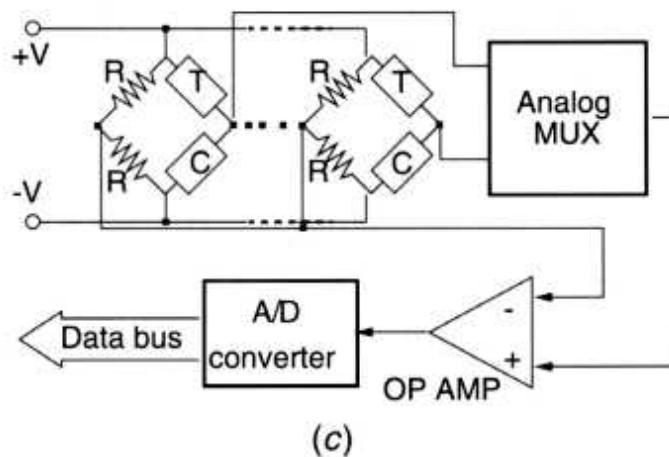
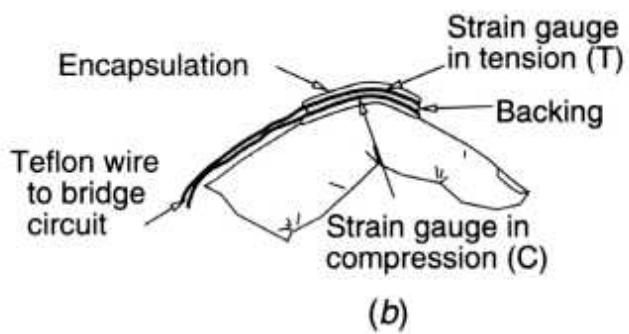
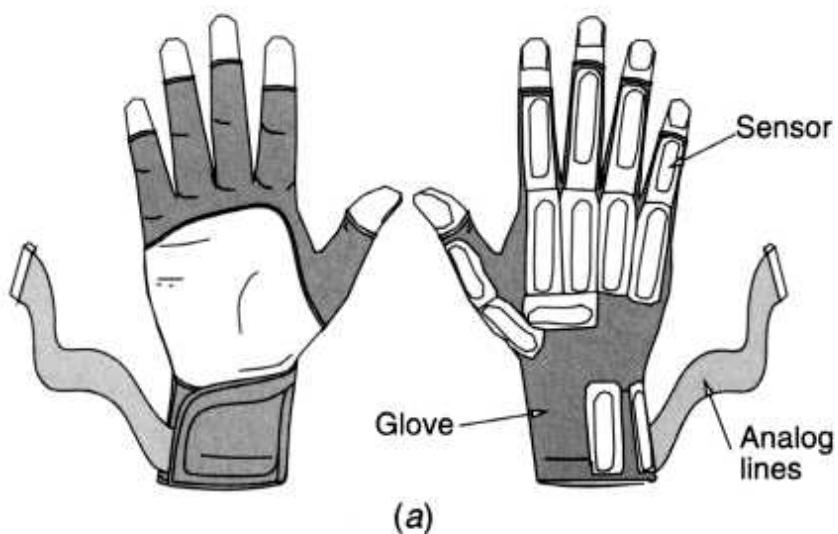


Fig. 2.29 The CyberGlove: (a) instrumented glove; (b) sensor detail; (c) interface electronics. Adapted from Kramer et al. [1991]. © Immersion Co. Reprinted by permission.

2.3.4 The CyberGlove

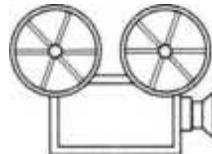
A more complex (and more expensive) sensing glove, which uses linear bend sensors, is the CyberGlove [Immersion Co., 2001]. This glove was invented by Jim Kramer as a gesture recognition interface in order to aid persons with speech impairments [Kramer and Leifer, 1989]. It subsequently became apparent that the same device could be successfully used as a VR interface.

The CyberGlove incorporates thin electrical strain gauges placed on an elastic nylon blend material, as shown in Figure 2.29 [Kramer et al., 1991]. The palm area (and the fingertips in some models) is removed for better ventilation and to allow normal activities such as typing, writing, etc. As a result the glove is light and easy to wear.

The glove sensors are either rectangular (for the flexion angles) or U-shaped (for adduction-abduction angles). There are between 18 and 22 sensors in the glove, used to measure finger flexing (two or three per finger), abduction (one per finger), plus thumb anteposition, palm arch, and wrist yaw and pitch. According to the manufacturer, sensor resolution is 0.5° and remains constant over the entire range of joint motion [Immersion Co., 2001]. It is further claimed that this glove has decoupled sensors so that outputs are independent of each other.

Joint angles are measured indirectly by a change of resistance in a pair of strain gauges. During finger motion one of the strain gauges is under compression (C), while the other is under tension (T). Their change in resistance produces a change in voltage on a Wheatstone bridge, as shown in Figure 2.29c. There are as many Wheatstone bridge circuits as sensors on the glove. These differential voltages are then demultiplexed, amplified, and subsequently digitized by an A/D converter. The glove data from its 18 sensors are then sent to the host computer over an RS232 serial line. The communication rate (currently 115.2 baud) allows a maximum of 150 datasets to be sent every second. The sampling rate drops to 112 datasets when filtering is used (to reduce sensor noise), and drops further for the 22-sensor CyberGlove model. Calibration is required to account for the user's hand size and to convert strain gauge voltages to joint angles.

Currently the CyberGlove is a de facto standard for high-performance hand measurements. This is due to its large number of sensors, its good programming support, and its extension into more complex haptic gloves (discussed in Chapter 3). Table 2.3 compares the sensing gloves described here.



VC 2.3

TABLE 2.3. Performance Comparison of Various Sensing Gloves

Specifications	Pinch Glove	5DT Data Glove	Didjiglove	CyberGlove
Number of sensors	7/glove (2 gloves)	5 or 14 /glove (1 glove)	10/glove (2 gloves)	18 or 22/glove (1 glove)
Sensor type	Electrical	Fiber-optic	Capacitive	Strain gauge
Records/sec	NA	100 (5DT 5W), 200 (5DT 5)	70	150 (unfiltered), 112 (filtered)
Sensor resolution	1 bit (2 points)	8 bit (256 points)	10 bit (1024 points)	0.5°
Communication rates	Wired (19.2 kb)	Wireless (9.600 kb), wired (19.2 kb)	Wired (19.2 kb)	Wired (115 kb)
Wrist sensors	None	Pitch (5DT 5 model)	None	Pitch and yaw

2.4 CONCLUSION

This chapter presented a number of input devices, ranging from trackers to trackballs, 3D probes, and sensing gloves. These devices are all aimed at capturing the user's input in real time and transmitting it to the host computer running the simulation. These interfaces are key to the interactivity component of the VR three I's continuum. The simulation response to the user's input is conveyed through output interfaces, which are the subject of the next chapter.

2.5 REVIEW QUESTIONS

1. What are trackers? Enumerate some important tracker characteristics (make drawings to illustrate your concepts).
2. How do AC magnetic trackers function? How is their accuracy affected by the presence of metal objects and distance from the source?
3. Why do DC magnetic trackers suffer less from metal interference than AC ones?
4. How does a wireless tracking suit work? Give examples.
5. What is the difference between an absolute and a relative position input device?
6. What are the advantages/disadvantages of each?
7. What are the advantages/disadvantages of ultrasonic trackers?
8. What is a HiBall tracker and in what ways is it superior to magnetic trackers?
9. What are hybrid trackers?
10. How do InterSense trackers differ from magnetic ones?
11. What are gesture input devices?
12. How does the Pinch Glove work?
13. What is the 5DT Data Glove and how does it function?
14. How does the CyberGlove work?

REFERENCES

Adelstein, B., E. Johnston, and S. Ellis, 1996, "Dynamic Response of Electromagnetic Spatial Displacement Trackers," *Presence*, Vol. 5(3), pp. 302-318.

American Society for Surgery of the Hand, 1983, *The Hand: Examination and Diagnosis*, Churchill Livingstone, London.

Anon, 1998, "3D Navigational and Gestural Devices," *VR News*, Vol. 7(1), pp. 26-29.

Anon, 2000, "3 Strings Pty. Ltd," *VR News*, Vol. 9(1), p. 6.

Ascension, 1998, "Flock of Birds Real-Time Motion Tracker," company brochure, Ascension Technology Co., Burlington, VT.

Ascension, 2001a, "laserBIRD," Ascension Technology Co., online at www.ascension-tech.com.

Ascension, 2001b, "MotionStar Wireless Installation and Operation Guide," company brochure, Ascension Technology Co., Burlington VT. Also online at www.ascension-tech.com.

Blood, E., 1989, "Device for Quantitatively Measuring the Relative Position and Orientation of Two Bodies in the Presence of Metals Utilizing Direct Current Magnetic Fields," U.S. Patent 4,849,692, July 18 1989.

Burdea, G., 1993, "Virtual Reality Systems and Applications" [Short Course], in *Electro'93 International Conference*, Edison, NJ.

Burdea, G., and P. Coiffet, 1993, *La Realite Virtuelle*, Hermes, Paris.

Burdea, G., S. Dunn, C. Immendorf, and M. Mallik, 1991, "Real-Time Sensing of Tooth Position for Dental Subtraction Radiography," *IEEE Transactions on Biomedical Engineering*, Vol. 38(4), pp. 366-378.

Burdea, G., J. Zhuang, E. Roskos, D. Silver, and N. Langrana, 1992, "A Portable Dextrous Master with Force Feedback," *Prcence-Teleoperators and Virtual Environments*, Vol. 1(1), pp. 18-27.

Fakespace, Labs Inc., 1998, "The PUSH Desktop Display," company brochure, Fakespace Labs Inc., Mountain View, CA. Also online at www.fakespace.com.

Fifth Dimension Technology Inc., 2000, 5DT Data Glove 5 User's Manual, Fifth Dimension Technology, Inc., Pretoria, South Africa.

Fischer, R., 1998, "Knightmare in East Anglia and the Gypsy Comes Home," VR News, Vol. 7(6), pp. 21-24.

Foxlin, E., 2002, "Motion Tracking Requirements and Technologies," in K. Stanney (Ed.), Handbook of Virtual Environments, Erlbaum, Mahwah, NJ, pp. 163-210.

Foxlin, E., M. Harrington, and G. Pfeifer, 1998, "Constellation: A Wide-Range Wireless Motion Tracking System for Augmented Reality and Virtual Set Applications," in Proceedings of SIGGRAPH '98, ACM, Orlando, FL, pp. 371-378.

Frohlich, B., J. Plate, J. Wind, G. Wesche, and M. Gobel, 2000, "Cubic-Mouse-Based Interaction in Virtual Environments," IEEE Computer Graphics and Applications, 2000 (July/August), pp. 12-15.

Hansen, P., 1998, "Optical 6D Measurement System with Two Fan Shaped Beams Rotating Around One Axis," U.S. Patent 5,742,394, April 21, 1998.

Hansen, P., and V. Kogan, 1999, "Sourceless Orientation Sensor," U.S. Patent 5,953,683, September 14, 1999.

Hightower, J., and G. Borriello, 2001, "Location Systems for Ubiquitous Computing," Computer, Vol. 34(8), pp. 57-66.

Hirzinger, G., and A. Bejczy, 1988, "Rotex-TRIIFEX: Proposal for a joint FRG-USA Telerobotic Flight Experiment," in Proceedings of NASA Conference on Space Telerobotics, Pasadena CA, pp. 111-123.

ID8 Media, 1999, "The Gypsy Motion Capture System," company brochure, ID8 Media, San Francisco. Also online at www.id8media.com.

Immersion Co., 1993, "Immersion Probe," Immersion Co., Palo Alto, CA.

Immersion Co., 2001, "CyberGlove," online at www.immersion.com/products/3d/.

InterSense, 2000a, "InterTrax2," online at www.isense.com.

InterSense, 2000b, "IS-900 Precision Motion Tracker," company brochure, InterSense Co., Burlington, MA.

Kramer, J., and L. Leifer, 1989, "The Talking Glove: A Speaking Aid for Nonvocal Deaf and Deaf-Blind Individuals," in Proceedings of the RESNA 12th Annual Conference, New Orleans, LA, pp. 471-472.

Kramer, J., P. Lindener, and W. George, 1991, "Communication System for Deaf, Deaf-Blind, or Non-vocal Individuals Using Instrumented Glove," U.S. Patent 5,047,952, September 10, 1991.

Krieg, J., 1993, "Motion Tracking: Polhemus Technology," Virtual Reality Systems, Vol. 1(1), pp. 32-36.

Laviola, Jr., J., 2000, "MSVT: A Virtual Reality-Based Multimodal Scientific Visualization Tool," in Proceedings of the Third LASTED International Conference on Computer Graphics and Imaging, pp. 1-7.

McDowall, I., M. Bolas, R. Mead, and C. Greuel, 2000, "Virtual Reality Glove System with Fabric Conductors," U.S. Patent 6,128,004, October 3, 2000.

Mead, Jr., R., M. Bolas, and I. McDowall, 2000, "Gimbal-Mounted Virtual Reality Display System," U.S. Patent 6,094,180, July 25, 2000.

Neely, J., and P. Restle, 1997, "Capacitive Bend Sensor," U.S. Patent 5,610,528, March 11, 1997.

Nixon, M., B. McCallum, R. Fright, and B. Price, 1998, "The Effect of Metals and Interfering Fields on Electromagnetic Trackers," Presence, Vol. 7(2), pp. 204-218.

Pimentel, K., and K. Teixeira, 1993, Virtual Reality: Through the New Looking Glass, Windcrest McGill, New York.

Rosenberg, L., B. Schena, S. Brave, and B. Jackson, 2000, "Method for Producing a Precision 3-D Measuring Apparatus," U.S. Patent 6,015,473, January 18, 2000. Also online at www.immersion.com.

Sheingold, D., 1981, Transducer Interfacing Handbook, Analog Devices, Norwood, MA.

SIMI, 1999, "SIMI Motion Capture 3D," company brochure, SIMI Reality Motion Systems GmbH, Unterschleissheim, Germany. Also online at www.simi.com.

Sowizral, H., and J. Barnes, 1993, "Tracking Position and Orientation in a Large Volume," in Proceedings of IEEE Virtual Reality Annual International Symposium, Seattle, WA, pp. 132-139.

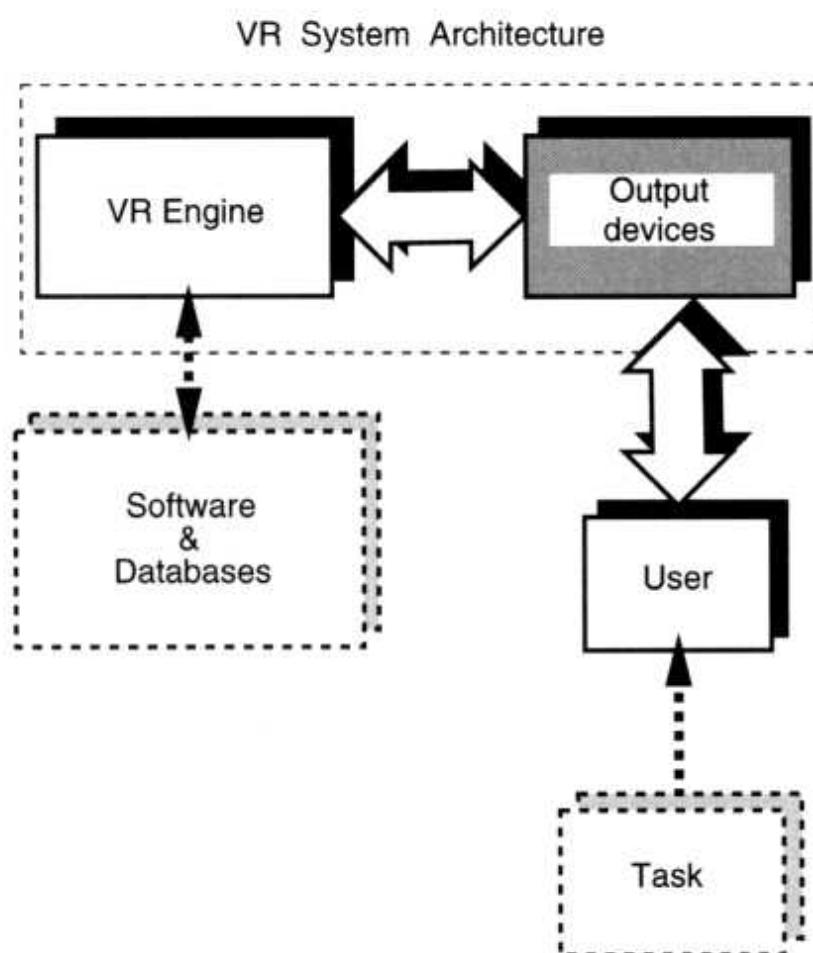
Tehrani, B., 1999, "SIGGRAPH '99 Top 20 Products," Animation Magazine, (October), p. 53.

Treffitz, H., and G. Burdea, 2000, "3-D Tracking Calibration of a Polhemus Long Ranger Used with the Baron Workbench," Rutgers University CAIP Center, Technical Report CAIP-TR- 243, Piscataway, NJ.

Welch, G., G. Bishop, L. Vicci, S., Brumback, K. Keller, and D. Coluci, 2001, "HighPerformance Wide-Area Optical Tracking," Presence, Vol. 10(1), pp. 1-21.

CHAPTER 3

OUTPUT DEVICES: GRAPHICS, THREE-DIMENSIONAL SOUND, AND HAPTIC DISPLAYS



The previous chapter described 3D trackers, trackballs, and sensing gloves, which are devices used to mediate the user's input into the VR simulation. Now we look at special hardware designed to provide feedback from the simulation in response to this input. The sensorial channels fed back by these

interfaces are sight (through graphics displays), sound (through 3D sound displays), and touch (through haptic displays).

For reasons of simplicity, this chapter treats each sensorial feedback modality separately. A further simplification here is the decoupling of sensorial feedback from the user's input. Modern VR systems are, however, multimodal, and feedback interfaces usually incorporate hardware to allow user input (such as trackers, pushbuttons, etc.). Combining several types of sensorial feedback in a simulation increases the user's immersion into VR. Furthermore, combining several sensorial modalities improves the simulation realism and therefore the usefulness of VR applications.

3.1 GRAPHICS DISPLAYS

Definition A graphics display is a computer interface that presents synthetic world images to one or several users interacting with the virtual world.

Other ways to characterize graphics displays are according to the type of image produced (stereoscopic or monoscopic), their image resolution (number of pixels in the scene), the field of view (portion of the eye's viewing volume they cover), display technology (LCD- or CRT-based), ergonomic factors (such as weight), and cost. The great variety of graphics displays is a result of the fact that vision is the most powerful human sensorial channel, with an extremely large processing bandwidth. Some VR systems may not incorporate 3D sound or haptic feedback interfaces, but all will have some type of graphics display.

3.1.1 The Human Visual System

Designing or selecting a graphics display cannot be done meaningfully without first understanding the human visual system. An effective graphics display needs to match its image characteristics to those of the user's ability to view the synthetic scene. Therefore, before attempting to describe any particular viewing hardware, it is necessary to describe the human visual perception mechanism.

The eye has over 126 million photoreceptors, with an uneven distribution over the retina. The central area of the retina (several degrees around the eye's viewing axis) is called the fovea. It represents a high-resolution, color perception area. This is surrounded by low-resolution, motion perception photoreceptors covering the rest of the viewing field. The portion of the displayed image that is projected on the fovea represents the focus area. A viewer's focus changes dynamically and unconsciously during a simulation and can be detected if the eye motion is tracked. Unfortunately, the eye tracking technology is at present too bulky to be meaningfully integrated with personal displays. Since we do not know what portion of the display is viewed by the fovea, the whole scene needs to be rendered at high resolution. This represents a waste of graphics pipeline processing (as discussed in detail in Chapter 4).

Another important characteristic of the human vision system is the field of view. This is approximately 150° horizontally and 120° vertically when one eye is used and grows to 180° horizontally and 120° vertically when both eyes are used [Kalawsky, 1993]. A central portion of this viewing volume represents the area of stereopsis, where both eyes register the same image. This binocular overlap is approximately 120° horizontally. The brain uses the horizontal shift in image position registered by the two eyes to measure depth, or the distance from the viewer to the virtual object presented in the scene [Julesz, 1971]. A physiological model of human stereoscopic vision adapted from Hatada [1992] is illustrated in Figure 3.1.

Within the field of view the eyes register the objects surrounding the viewer, such as object A, which is behind object B. The eyes concentrate on a feature of B, focusing on a fixation point F. The angle between the viewing axis and the line to the fixation point determines the convergence angle. This angle depends also on the distance between the pupils of the right and left eyes. This distance is called the interpupillary distance (IPD) and varies among male and female adults within a range of 53-73 mm [Robinett and Rolland, 1992]. The IPD is the baseline from which a person interprets distances to objects in the real world. The larger the IPD, the larger is the convergence angle. The point F will appear shifted horizontally between the right and left eyes because of its different position in relation to the two

eyes. This shift is called image parallax and needs to be replicated by the VR graphics and stereo viewing hardware in order to help the brain interpret depth in the simulated world. To do so stereographics displays need to output two slightly shifted images. When two displays are used (such as in head-mounted displays), each presents its image to the corresponding eye. When a single display is used, the two images can be time-sequenced (when using shutter glasses) or spatially sequenced (for autostereoscopic displays).

Stereopsis works well as a depth cue in the nearfield, where the image parallax is substantial. As objects get further away from the viewer, their horizontal shift in the viewing volume gets smaller, such that stereopsis is degraded substantially at approximately 10m from the user [Durlach and Mavor, 1995]. Depth perception then occurs based on cues inherent in the image, such as linear perspective, shadows, occlusions (distant objects being blocked from view by closer ones), surface texture, and object detail. Also important in monoscopic depth perception is motion parallax, so that closer objects seem to move further than distant ones when the user's head is moved. These depth cues are effective even if only one eye is used.

Depth perception, combined with a large viewing area, and high-resolution images are important factors determining the user's subjective feeling of immersion in the simulated world. Designing graphics displays that satisfy all these requirements, while at the same time being ergonomic and inexpensive, is a daunting technological task.

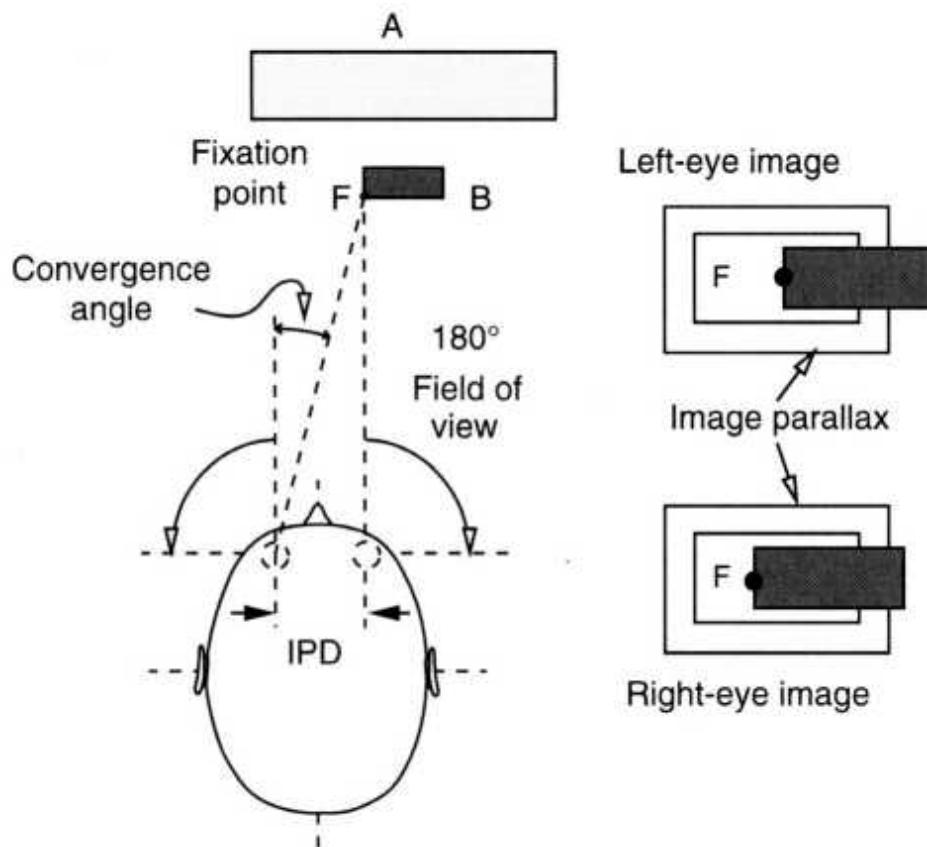


Fig. 3.1 Physiological model of stereoscopic vision. IPD, Interpupillary distance. Adapted from Hatada [1992]. Reprinted by permission.

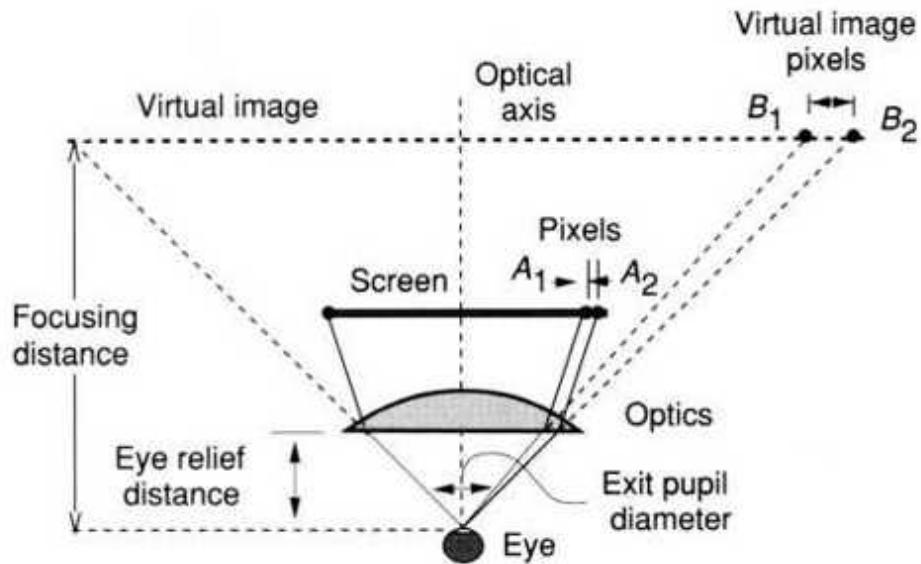


Fig. 3.2 Simplified optics model of an HMD. Adapted from Robinett and Rolland [1992]. ©1992 Massachusetts Institute of Technology. Reprinted by permission.

3.1.2 Personal Graphics Displays

Definition A graphics display that outputs a virtual scene destined to be viewed by a single user is called a personal graphics display. Such images may be monoscopic or stereoscopic, monocular (for a single eye), or binocular (displayed on both eyes).

Within the personal display category are grouped head-mounted displays (HMDs), hand-supported displays, floor-supported displays, and autostereoscopic monitors.

3.1.2.1 Head-Mounted Displays. These project an image floating some 1-5 m (3-15 ft) in front of the user [Anon, 2001], as illustrated in Figure 3.2. They use special optics placed between the HMD small image panels and the user's eyes in order to allow the eyes to focus at such short distances without tiring easily. Optics is also needed to magnify the small panel image to fill as much as possible of the eyes' field of view (FOV). As can be seen in Figure 3.2 [Robinett and Rolland, 1992], the unfortunate byproduct is that the distance between display pixels (At, A2) is amplified as well. Therefore the "granularity" of the HMD displays (expressed in arc-minutes/pixel) becomes apparent in the virtual image. The lower the HMD resolution and the higher the FOV, the greater is the number of arc-minutes of eye view corresponding to each pixel. Older HMDs had very low resolution (as discussed in Chapter 1) and incorporated a diffuser sheet overlaid on the input to their optics. These diffusers blurred the edges between pixels somewhat, and the image was perceived to be better. Modern HMDs have resolutions up to extended video graphics array (XVGA; 1024 x 768) or better. Even these resolutions may be unsatisfactory if the optics amplifies the pixel dimension too much in order to enlarge the apparent FOV. Furthermore, very large FOVs force large exit pupil diameters, which can produce shadows around the edges of the perceived image.

Another HMD characteristic is the display technology used. Consumer-grade HMDs use LCD displays, while more expensive professional-grade devices incorporate CRT-based displays, which tend to have higher resolution. Since consumer HMDs are designed primarily for private viewing of TV programs and for video games rather than for VR, they accept NTSC (in Europe, phase alternating line, PAL) monoscopic video input. When integrated within a VR system, they require the conversion of the graphics pipeline output signal red-green-blue (RGB) format to NTSC/PAL, as illustrated in Figure 3.3a. The HMD controller allows manual adjustment for brightness and forwards the same signal to both HMD displays. Professional-grade HMDs are designed specifically for VR interaction and accept RGB video input. As illustrated in Figure 3.3b, in the graphics pipeline two RGB signals are sent directly to the HMD control unit (for stereoscopic viewing). The control unit also receives stereo sound, which is then sent to the HMD built-in headphones. The user's head motion is tracked and position data sent back to the VR engine for use in the graphics computations. A ratchet on the back of the HMD head support allows comfort adjustment for various head sizes.

The HMD weight, comfort, and cost are additional criteria to be considered in comparing the various models on the market. Technology has made tremendous advances in miniaturizing the HMD displays and associated optics. The first LCD-based HMD (the VPL EyePhone), which was available in the early 1990s, had a resolution of 360 x 240 pixels, a FOV of 100° horizontally and 60° vertically, and a weight of 2.4 kg (2400 grams). This large weight induced user fatigue. Modern LCD-based HMDs, such as the Olympus Eye-Trek, shown in Figure 3.4a [Isdale, 2000], weighs only about 100 grams (24 times less!). This class of HMDs resembles eyeglasses, and so they are also called face-mounted displays (FMDs). Key to the compactness and lightness of the Olympus FMD is the placement of its small active matrix LCD (AMLCD) display panels eccentrically, directly above the optics (as shown in Figure 3.4b) [Olympus Co., 2001]. This obviates the need for a half-mirror as used in the earlier Sony Glastron FMD. Since there is less light loss (due to the absence of the half-mirror), the image seems brighter. In order to compensate for optics-induced image distortions (or aberrations), the Olympus team invented a variable-curvature

free-form lens. The result is a monoscopic FMD with a field of view of 30° horizontally and 22.7° vertically, a very low weight, a fixed IPD, and a cost of less than \$500 (in year 2001 prices). The virtual image seen by the user has a 1.3 m (62 in.) diagonal at a fixed focusing distance of 2 m in front. An interesting enhancement of the FMD 200, recently introduced in Europe, is the cordless TV. A pair of miniature TV transmitter and receiver units, operating in the 2.4-GHz audio/video frequency range, allows viewing up to 100 m from the TV. This greatly enhances the user's freedom of motion, at the expense of a larger system weight (the receiver is coupled with the FMD controller). Furthermore, interference from other home appliances using the same frequency range and from metal in the house construction can occur.

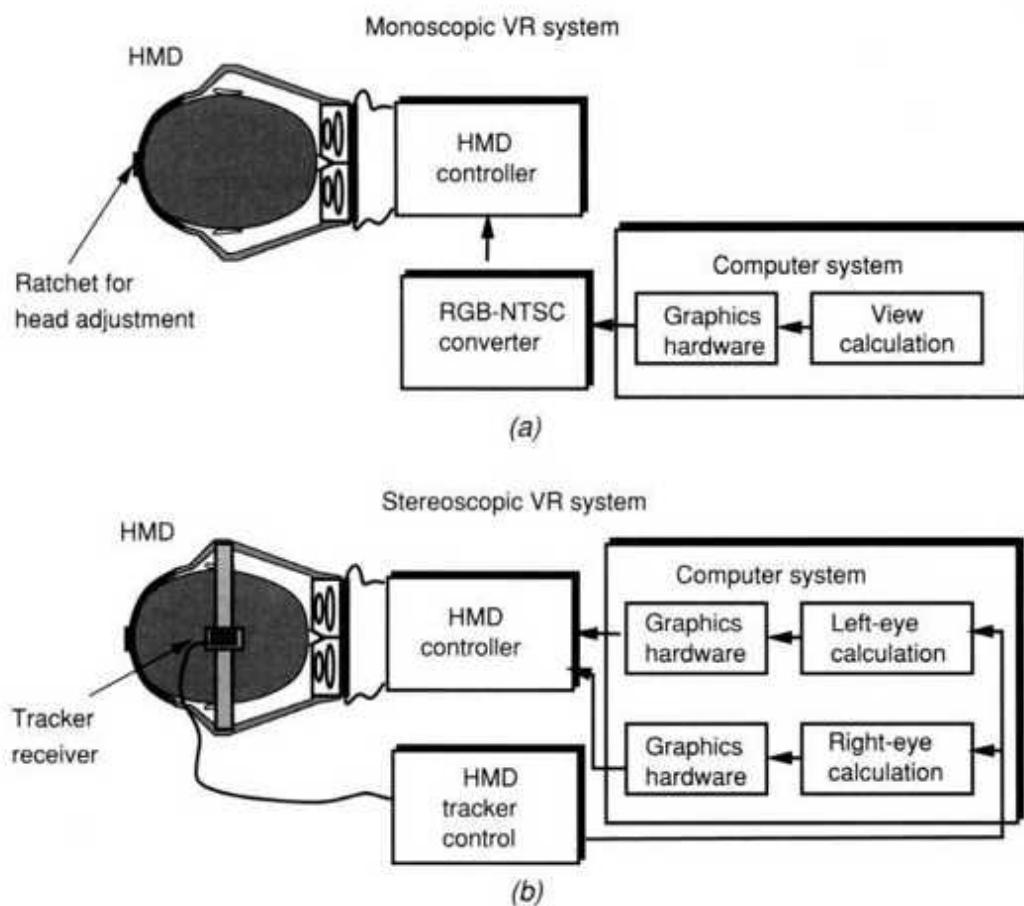


Fig. 3.3 Head-mounted display (HMD) integration in a VR system for: (a) consumer (monoscopic) HMD; (b) professional (stereoscopic) HMD. Adapted in part from Pimentel and Teixeira [1993]. Reprinted by permission.

For all its ergonomic advantages, the Olympus FMD 200 has a number of drawbacks. It cannot be easily retrofitted with a head tracker, for which there are no attachments provided. A tracker receiver placed on top of the FMD will make the whole assembly fall off due to the added weight and the lack of a restraint at the back of the head. Another problem is the FMD 200 resolution, which is only 267 x 225 (or 180,000 pixels/LCD panel). Recognizing this problem, Olympus developed the FMD 700, which has the same FOV and four times the resolution of the FMD 200 model. The FMD 700 resolution of 533 x 450 (720,000 pixels/LCD panel) is obtained by adding special polarization and double-refraction effects, a process which Olympus calls optical super resolution.

Another LCD-based consumer FMD is the Daeyang cy-visor DH-4400VP shown in Figure 3.5a. Although it looks somewhat similar to the Eye-Trek FMD, the cy-visor uses liquid-crystal-on-silicon (LCOS) display technology. Classic AMLCD panels are transmissive, meaning that light passes through the electrode/liquid crystal/electrode substrate and is then seen by the user. The great advantage of newer LCOS LCD displays is that the liquid crystal layer is mounted directly on a silicon chip, which is used as control circuitry. This results in higher electrode densities (i.e., display resolution), less external circuitry, reduced power consumption, reduced volume, and reduced manufacturing costs compared to AMLCD-based displays. However, since they do not have backlighting, LCOS reflective displays need external light to be able to display the image [Schott, 1999]. As seen in Figure 3.5b, the cy-visor incorporates a light-emitting diode (LED) light source and a combination of an optical prism and a 45° mirror/polarizer to illuminate the LCOS panel. This light is reflected by a substrate under the liquid crystal layer of the LCOS chip, is blocked by the activated pixels, and subsequently is seen by the eye through another polarizer and optics at the front end of the FMD. By using this new technology the cy-visor is able to display an SVGA-resolution image (800 x 600 pixels) with a 60° horizontal x 43° vertical FOV. The use of the LCOS chips is one reason why the cy-visor is light (160 grams) while having an image resolution superior to any of the Eye-Trek FMD models previously described.

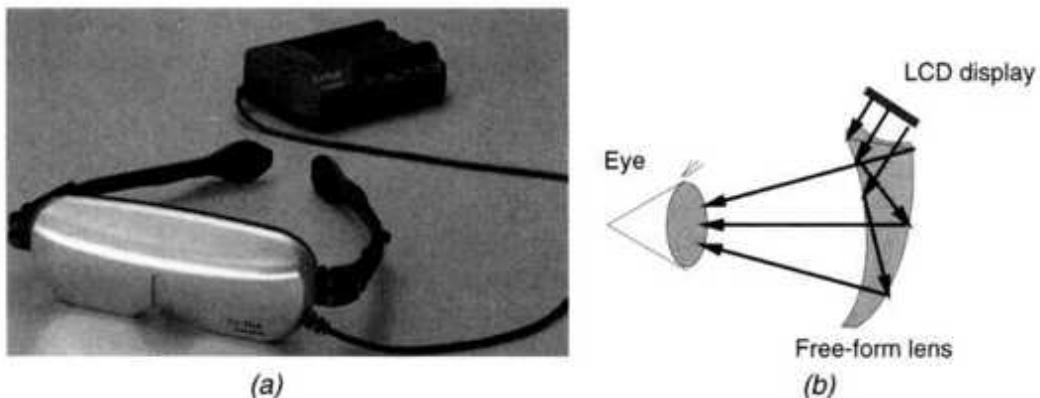


Fig. 3.4 The Olympus LCD-based Eye-Trek FMD: (a) the FMD 200; (b) optics arrangement. Adapted from Olympus Co. [2001]. Reprinted by permission.

Even with improved resolution, consumer FMDs cannot match the professional-grade LCD-based HMDs. An example of such devices is the Pro View XL35 illustrated in Figure 3.6a [Kaiser Electro-Optics, 2001]. It has high-resolution AMLCD displays located at the user forehead and attached to an ergonomically shaped adjustable head support. The support allows for the placement of a tracker close to the top of the head, while at the same time serving as a counterweight to reduce the user's fatigue. Each display has three separate LCD panels, with red, green, and blue backlights and an "X-cube" optical path for each eye. This color addition approach has fewer visual artifacts than single-panel based displays. This results in a 28° horizontally and 21° vertically stereo (100% overlap) FOV, comparable to that of the FMD 200. However, the ProView XL35 has a much higher XGA resolution ($1024 \times 768 \times 3$ pixels). The high resolution, combined with a relatively small FOV, results in a very low virtual image granularity (1.6 arc-minutes/color pixel). Further advantages are the adjustable IPD (55-75 mm) and an eye relief that allows eyeglasses to be worn together with the ProView HMD. The high image quality of the ProView XL35 comes at a price in terms of much higher cost (\$19,500) and weight (992 grams, or 35 oz) compared to those of the FMD 200.

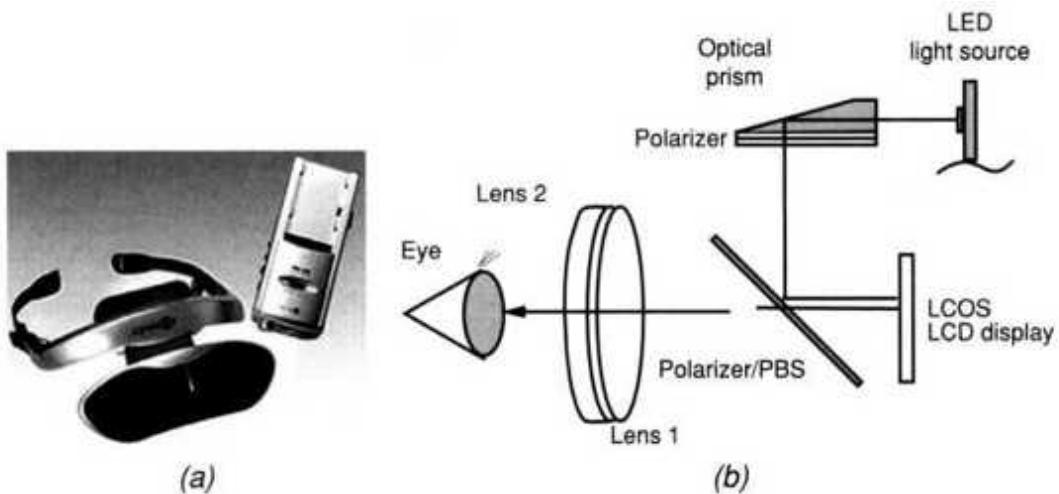


Fig. 3.5 The Daeyang cy-visor FMD: (a) the DH-4400VP; (b) optics arrangement. Adapted from Daeyang [2001]. Reprinted by permission.

The high resolution of the ProView XL35 HMD represents only 60% of that available on desk-top graphics monitors, which typically have 1280 x 1024 color pixels. To obtain such display resolutions, today's high-end HMDs use miniature (1-in. diameter) CRTs. An example is the Datavisor HiRes, illustrated in Figure 3.6b [n-vision Inc., 1998]. Its two monochrome (black/white) CRTs are placed laterally to the user's head and retrofitted with small Tektronix Nucolor liquid crystal shutters. The CRTs and shutter units are metal-shielded to eliminate electronic interference and to dissipate heat away from the user's head. Special care is required in the electrical circuitry because of the high voltage of the CRTs, which are close to the user's head.

By rapidly switching the filters, a single line in the Datavisor's image is first seen in red, then in green, and finally in blue. The human brain integrates the three images to obtain a color scene. This approach in turn requires the so-called field-sequential video input format, meaning that RGB color information is transmitted sequentially on a single line (rather than on three separate lines for the red, green, and blue components of an image). As a consequence the scan frequency needs to be three times the normal rate, and this in turn requires expensive high-speed electronics. The output of the LCD shutters is reflected off 45° mirrors and then through special optics directed to the user's eyes. These optics incorporate collimated windows,

which include an undistorted optical path to the eye. This allows the Datavisor to be configured for seethrough applications, such as augmented reality (AR), where graphics is overimposed on the visual image of the user's immediate surrounding area. The CRT and optics of the Datavisor are fully contained in a light plastic enclosure, which allows adjustments for head height and circumference. Further adjustments are for the IPD (58-73 mm) and focal distance (from 0.25 m to infinity). The Datavisor's resolution of 1280 x 1024 pixels is spread over a FOV of 78° horizontally and 39° vertically. The resulting image granularity is 1.9 arc-minutes/color pixel, which is about 20% worse than that of the Pro View XL35 LCD HMD. Other drawbacks of the Datavisor are large weight (1587 grams, or 56 oz) and unit cost (\$35,000). This is further complicated by safety concerns related to the CRT electromagnetic radiation close to the brain.

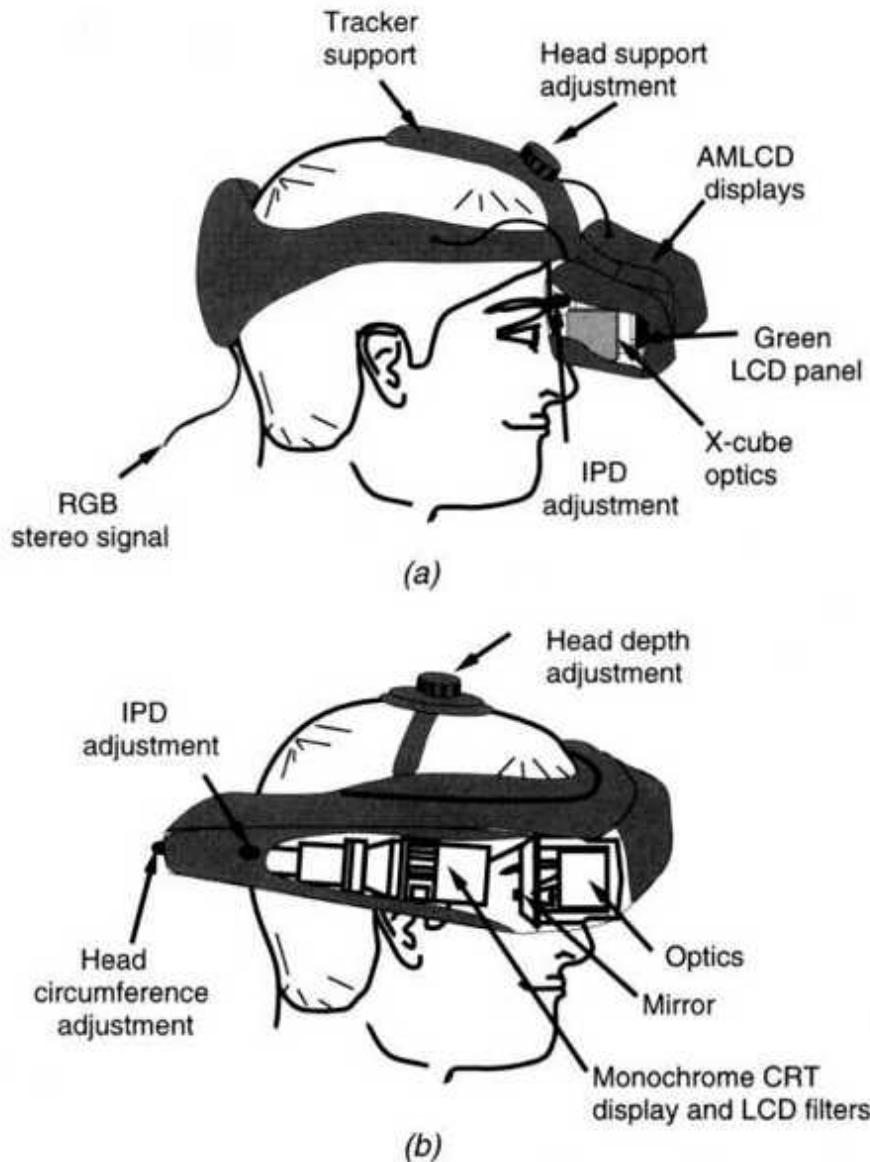


Fig. 3.6 Stereoscopic HMDs. (a) ProView XL35. Adapted from Kaiser Electro-Optics [2001]. Reprinted by permission. (b) Datavisor HiRes. Adapted from n-vision Inc. [1998].

3.1.2.2 Hand-Supported Displays (HSDs). These are personal graphics displays that the user holds in one or both hands in order to periodically view a synthetic scene. This means that the user can go in and out of the simulation as required by the application. HSDs are similar to HMDs in their use of special optics to project a virtual image in front of the user. In

addition, HSDs incorporate features not present in HMDs, namely push buttons used to interact with the virtual scene.

An example of a hand-supported graphics display is the virtual binoculars SX shown in Figure 3.7a [NVIS Inc., 2003]. These are constructed to resemble the look and feel of regular binoculars, to help the realism of the simulation. Internally, however, virtual binoculars incorporate two miniature LCOS displays and a tracker, which measures the user's viewing direction. The computer then updates the graphics based on tracker and pushbutton information, similar to trackball-mediated interaction.

The use of LCOS displays results in a high-resolution image (1280 x 1024 pixels) and low granularity (1.6 arc-minutes/pixel). However, the disadvantages mentioned earlier for CRT-based HMDs, namely large weight (about 1 kg, without the tracker), and high unit cost (\$19,900) persist. As illustrated in Figure 3.7b, an LED light source illuminates the LCOS display through a polarizer and cube beamsplitter. The light directed at the reflective LCOS panel bounces back to the eye, passing through another polarizer and the eyepiece optics. The virtual binoculars allows variable distance focusing as well as zooming on the scene, using a mouse button on the top of the device (close to the user's fingertips). This is connected to the computer running the simulation on a serial line. The tracker measuring the viewing vector is placed at the opposite end of the enclosure from the LCOS panel. This minimizes interference if the tracker is electromagnetic, and counterbalances somewhat the weight of the optics.

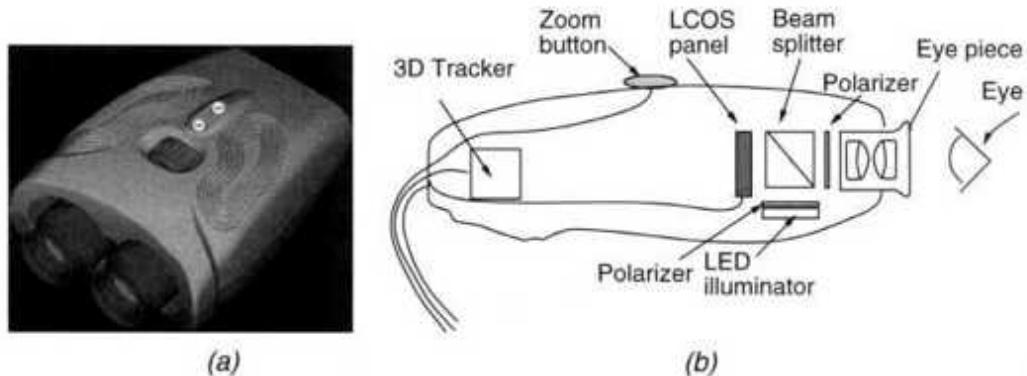


Fig. 3.7 The NVIS Inc. virtual binoculars SX: (a) appearance; (b) optics arrangement. Adapted from NVIS Inc. [2003]. Reprinted by permission.

3.1.2.3 Floor-Supported Displays. The HMDs and HSDs previously discussed rely on 3D trackers to measure the user's head position. When the user moves his or her head he or she expects the displayed image to move in the opposite direction. If the time delay between corresponding head and image motions is too large, simulation sickness may occur (as detailed in Chapter 7). In order to alleviate the problem of simulation sickness, it is necessary to have almost instantaneous response to the user's head motion, which suggests the use of a mechanical tracker (as discussed in Chapter 2).

Floor-supported displays use an articulated mechanical arm to offload the weight of the graphics display from the user. More importantly, floor-supported displays integrate sensors directly in the mechanical support structure holding the display. If six sensors are used, it is possible to determine the position and orientation of the end of the supporting arm relative to its base. This concept was developed by NASA in their counterbalanced CRT-based stereoscopic viewer (CCSV) [McDowall et al., 1990] and is used in the modern Boom3C display produced by Fakespace Labs [Fakespace Labs, 2001]. As illustrated in Figure 3.8a, raw analog data from the arm sensors are first converted into floating-point angles (based on internal calibration constants). Then direct-kinematics equations associated with open kinematic links are used to obtain the 3D position of the end of the arm. These parameters are then sent to the graphics workstation, which updates the images for the two eyes (assuming stereo graphics).

Each joint of the Boom3C supporting arm shown in Figure 3.8b has built-in optomechanical shaft encoders with a position resolution of 0.1° . As stated in Chapter 2, the major advantage that mechanical trackers have is their low latency. The tracking update rate depends on the serial communication line setting (140 datasets/sec for 19.2 kbaud, for example), while the latency is negligible (0.2 msec). The use of mechanical tracking eliminates the jitter associated with 3D magnetic or ultrasound trackers. Additionally, tracking is not adversely affected by either magnetic fields (as for magnetic trackers) or ultrasound background noise (as in ultrasound trackers). Similar to hand-supported displays, the Boom3C allows

intermittent use of computer keyboards without having to repeatedly take off and then put back on the HMD helmet. The disadvantage is less freedom of motion due to a dead zone created by the supporting pole in the center of the work space. The Boom3C work envelope is a hollow cylinder of 1.8 m (6 ft) diameter and 0.8 m (2.5 ft) height, with an exclusion core.

Floor-supported displays offer larger fields of view and superior graphics resolution than HMDs or hand-supported displays. This is due to the use of larger optics and CRT tubes, with weight supported by the mechanical arm. The graphics resolution of the Boom3C is 1280 x 1024, similar to the high-end Datavisor HMD presented earlier. However, the use of special large-expansive, extra-perspective (LEEP)-type optics in the Boom3C results in a field of view that is considerably larger than that of the Datavisor HMD.

Another type of floor-supported display is the WindowVR illustrated in Figure 3.9 [Virtual Research Systems, 1999]. It consists of a high-resolution LCD flat panel display (21 in. diagonal, 1600 x 1200 pixels), a 3D tracker, and two handles equipped with pushbuttons and switches. The weight of the display is supported by cables connected to an overhead counterbalanced arm. The user holds the LCD panel at arm's length and navigates using the handle buttons (which emulate a trackball). The simplicity of the device and the lack of complex optics make it intuitive to use. The feeling is that of viewing the simulation through a window (from which the name WindowVR). Unlike the Boom3C, there is no restrictive space inside the work envelope since the supporting arm is off-axis. Because the WindowVR relies on third-party trackers to measure the position/orientation of the display (such as InterSense 300 inertial trackers), it inherits the problems associated with latencies and tracker noise. Furthermore, only monoscopic graphics can be displayed, whereas the Boom3C allows stereoscopic graphics. Thus, choosing between the two depends on application-specific requirements and available budgets (the Boom3C is more expensive).

3.1.2.4 Desk-Supported Displays. Excessive display weight becomes an issue for HMDs and hand-supported personal displays due to the user's fatigue, which can lead to neck and arm pain. Even for floor-supported displays, excessive weight is undesirable, as it increases inertia when the

display is rotated and can lead to unwanted pendulum oscillations. One category of displays where weight is not an issue is desk-supported displays. Unlike previously discussed personal displays, desk-supported displays are fixed and designed to be viewed while the user is sitting. Thus the user's freedom of motion is limited when compared to HMDs or HSDs.

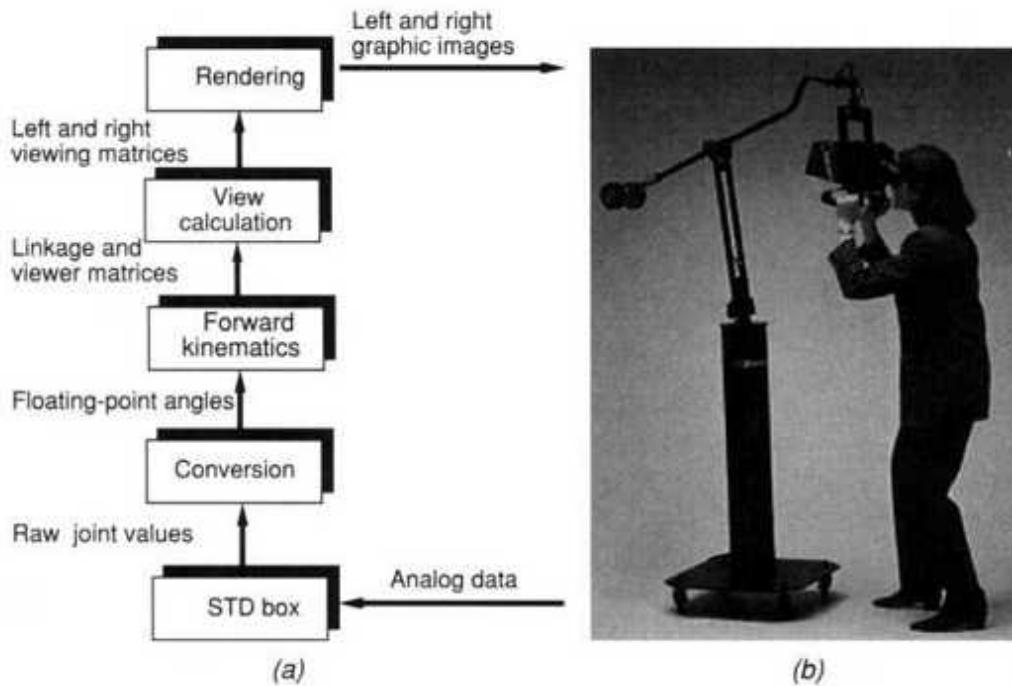


Fig. 3.8 The Boom3C floor-supported display: (a) data flow. Adapted from McDowall et al. [1990]. Reprinted by permission. (b) General appearance. Courtesy of Fakespace Labs Inc.

An interesting type of desk-supported displays are autostereoscopic ones, which produce a stereo image while viewed with unaided eyes. Such displays have a special "column-interlaced" image format, which alternates individual columns assigned to the left-eye view and the right-eye view. Thus, rather than using two display panels (one for each eye), auto stereoscopic displays present the stereo-pair images simultaneously on a single panel [NASA, 1998]. Special optics or illumination mechanisms need to be used in order to assure that each eye sees only its corresponding image columns, which results in the perception of two images, and through parallax, a single stereo image floating in front of the autostereoscopic display. The advantage of this approach is the ability to present a stereo

image without requiring the user to wear any vision apparatus. One disadvantage is reduced horizontal image resolution (since half of the columns display one eye's image and half the other eye's image). Other disadvantages are increased system complexity (due to the added illumination or viewing optics or dedicated hardware integrated within the display) and increased cost (compared with CRTs or flat panel displays of equal size).

Within the autostereoscopic display category one distinguishes passive and active displays. Passive autostereoscopic displays do not track the user's head and thus restrict the user's eyes to be within a small area for the stereo perception to be realized. One such display is the DTI 2018XL Virtual Window, shown in Figure 3.1Oa [Dimension Technologies Inc., 2002]. The Virtual Window uses thin-film transistor (TFT) LCD color arrays, which are controlled for spatial separation of the stereo-pair images. Each LCD pixel is back-lit by a number of very narrow light sources, as illustrated in Figure 3.1Ob [Eichenlaub and Martens, 1992]. These column light sources are formed by a number of fluorescent lamps, a reflector, and a plastic light guide with parallel-etched prisms. Light then passes through a variable diffuser and the LCD panel and forms stereo viewing zones in front of the DTI 2018XL [Eichenlaub, 1998]. The relation between the backlighting distance d and the distance to the user D determines a stereo viewing cone where the user has to be to see an image in stereo, otherwise the effect is lost. The resolution of the DTI 2018XL in monoscopic (2D) mode is 1280 x 1024, whereas in stereo mode it is only 640 x 1024 owing to the spatial multiplexing of two images on a single display. Selection of the display mode (mono or stereo) is done with a 2D/3D switch on the front panel of the display. The weight of the display is substantial, namely 11.25 kg (25 lb), but it is supported by the desk.

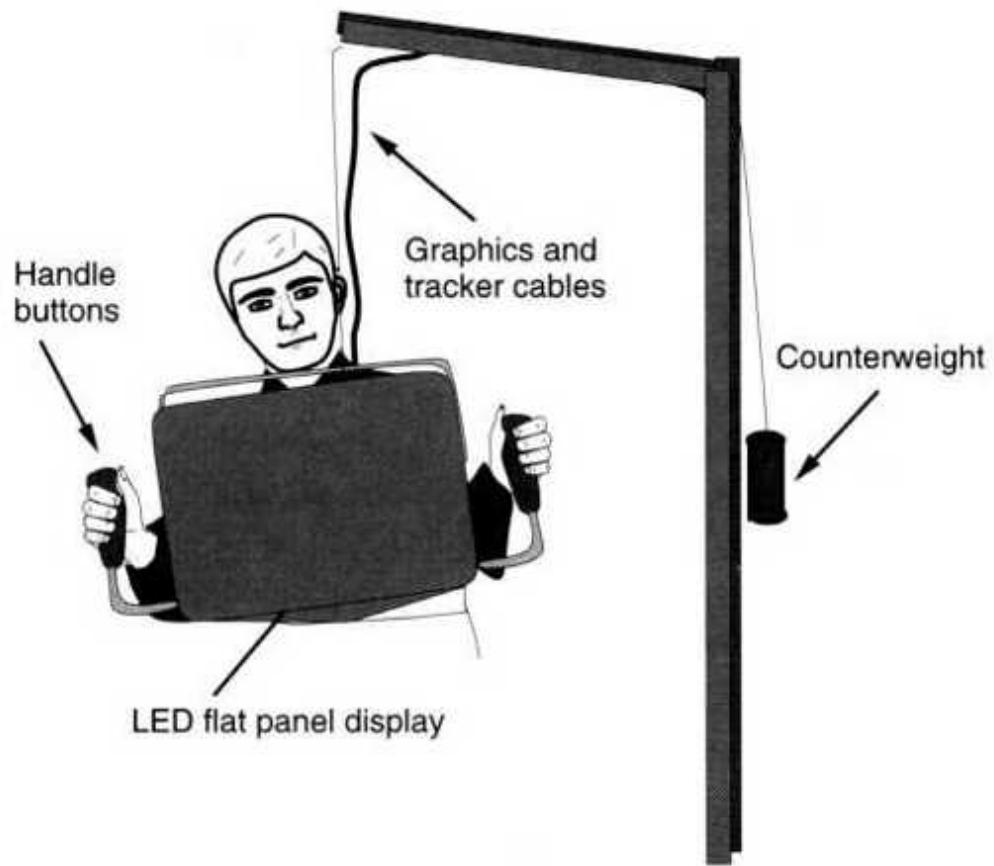
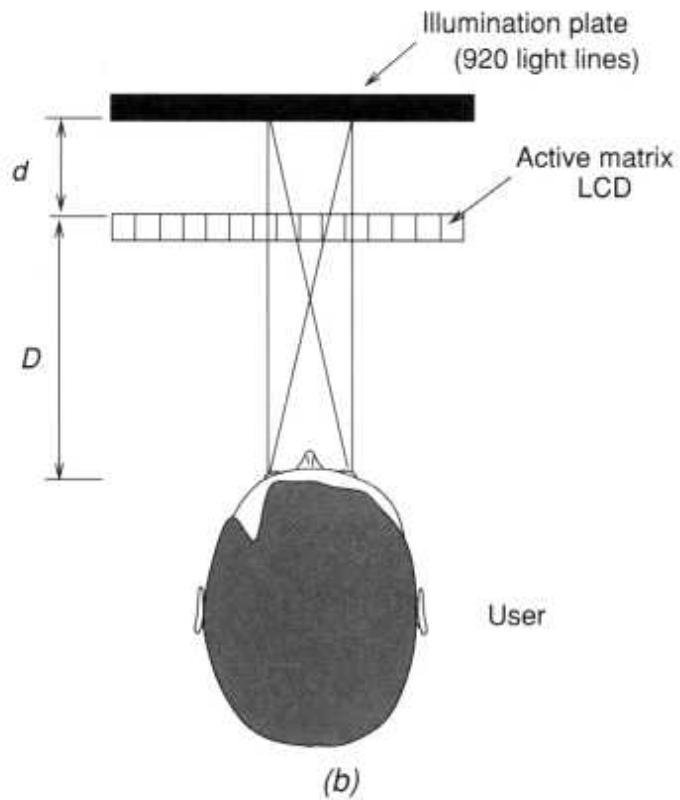


Fig. 3.9 The WindowVR monoscopic display. Adapted from Virtual Research Systems [1999]. Reprinted by permission.



(a)



(b)

Fig. 3.10 Autostereoscopic display. (a) Outside appearance. Courtesy of DTI. (b) Active matrix backlighting. Adapted from Eichenlaub and Martens [1992]. Reprinted by permission.

Active autostereoscopic displays alleviate the problem of limited viewing volume by tracking the user's head and performing a real-time adaptation of

the column interlaced image based on the user's instantaneous position. An example of such a display is the Ecomo4D display [Elsa Ag., 2000] shown in Figure 3.11. The device uses a 18.1-in. (46-cm)-diameter, 1280 x 1024 TFT flat panel display, similar to the one used in the DTI Virtual Window. The column viewing separation is done in a different way, however, namely with a prism mask placed over the display. A pair of cameras integrated on the top portion of the display frame tracks the user's X, Y, Z position in real time (20 times/sec). The camera images are passed through an image recognition stage and then sent to the tracking unit. This controls the position of a stepper motor, which translates the prism mask horizontally to account for changes in the user's position. This allows changes in viewing angle as large as $\pm 25^\circ$ to be accommodated when the user is 65 cm (about 26 in.) from the display.

The Ecomo4D autostereoscopic display uses mechanical means to adapt the column-interlaced image to the user's head position. This has disadvantages associated with any mechanical system, namely potential for breakage and noisy operation. A better approach would be to perform the image adaptation using electronic means. Hentschke [2001] proposed a system by which third-party head-tracker information is used to adapt the autostereoscopic image. This approach relies on an added hardware unit called the autostereoscopic adaptation coder (AAC). The AAC receives as input the two images from the graphics computer as well as the head tracker information. These data are used to change in real time the RGB subpixel arrangement in the column-interlaced image (output). In a normal (2D) display mode, the RGB subpixels are adjacent in a row. However, Hentschke's approach introduces subpixels that are intentionally left dark in the merged image. Such subpixels correspond to areas that the AAC determines to be visible to both eyes, or to boundary subpixels. In this way each eye sees its corresponding view of the 3D scene, based on the momentary user's head position, without mechanically changing the barrier stripe film or prism mask placed over the display. The drawback, of course, is a further reduction in image resolution, since a number of pixels are sacrificed (left dark). Table 3.1 summarizes the characteristics of the personal graphics displays discussed in this chapter.

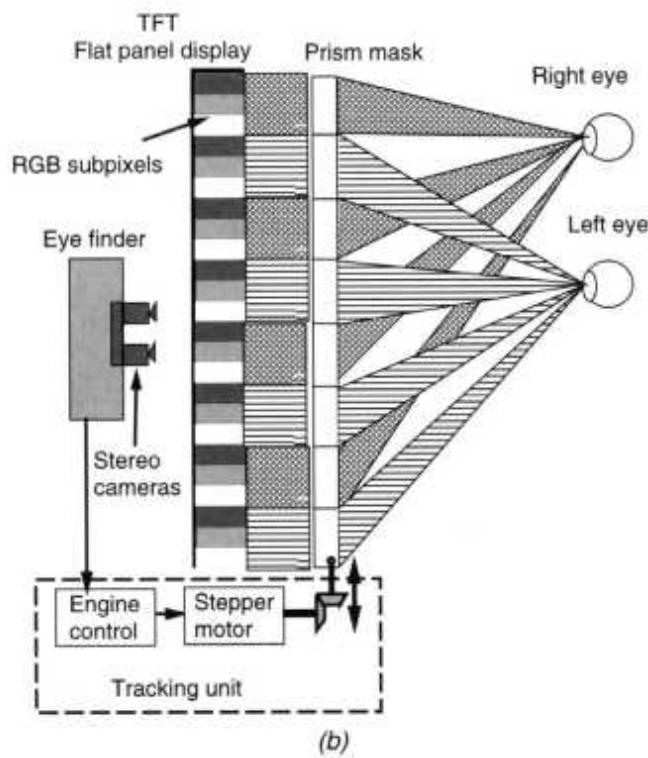
3.1.3 Large-Volume Displays

Table 3.1 makes apparent the price/performance differences that exist in personal graphics displays. Furthermore, it is hard to justify the high price of professionalgrade products when only one viewer is accommodated. Ideally, several users located together should be able to look at the same image, share ideas, and work collaboratively.

Definition Graphics displays that allow several users located in close proximity to simultaneously view a stereo or monoscopic image of the virtual world are called large volume displays.



(a)



(b)

Fig. 3.11 Elsa Ecomo4D autostereoscopic display. (a) Outside appearance. Courtesy of Elsa Ag. (b) Position-adaptive backlighting. Adapted from Dresden 3D [2001]. Reprinted by permission.

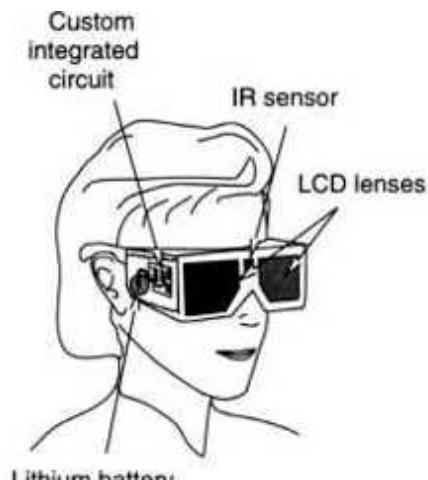
TABLE 3.1. Performance Comparison of Various Personal Graphics Displays^a

Manufacturer, Display Name	Type	Resolution (pixels)	FOV (H × V)	Weight (g)	Price (10 ³ \$)
Olympus Eye-Trek	AMLCD, FMD	267 × 225	30° × 23°	100	0.5
Daeyang cy-visor	LCOS, LCD, FMD	800 × 600	60° × 43°	160	1
Kaiser ProView XL35	AMLCD, HMD	1024 × 768	28° × 21°	992	20
n-vision Inc. Datavisor	CRT, HMD	1280 × 1024	78° × 39°	1,587	35
NVIS Inc. Virtual binoculars SX	LCOS, LCD, HSD	1280 × 1024	42° Diagona	1,000	19.9
Fakespace Labs Boom3C	CRT, FSD	1280 × 1024	85° × H	NA	≤100
Virtual Research WindowVR	Flat panel, FSD	1280 × 1024	21 in. Diagonal	NA	13.9
Dimension Technologies Virtual Window	TFT, LCD, autostereo	1280 × 1024 2D, 640 × 1024 3D	18.1 in. Diagonal	11,250	7
Elsa Ag. Ecomo4D	TFT, LCD, autostereo	1280 × 1024 2D, 640 × 1024 3D	18 in. Diagonal	17,000	15

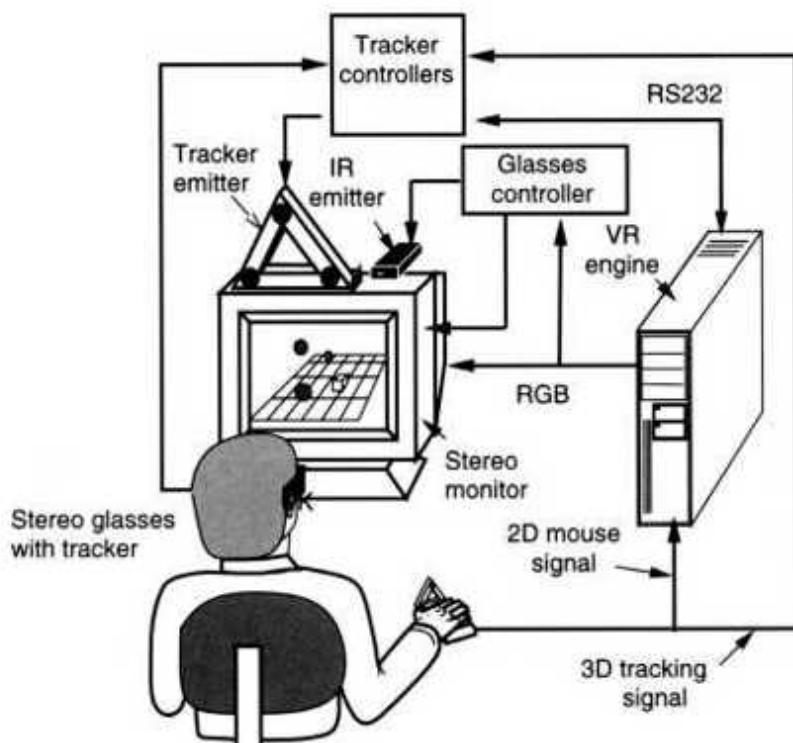
^aFOV, Field of view; H, horizontal, V, vertical. AMLCD, Active matrix liquid crystal display; FMD, face-mounted display; LCOS, liquid crystal on silicon; HMD, head-mounted display; CRT, cathode ray tube; HSD, hand-supported display; FSD, floor-supported display; TFT, thin film transistor. NA, Not available.

Depending on the type and size of the display they use, large-volume displays can be further classified as monitor-based (single or side-by-side CRTs) and projectorbased (workbenches, CAVEs, display walls, and domes). Their progressively larger work envelope improves users' freedom of motion and natural interaction capability compared to personal displays. The stereo large-volume displays generally use a temporal-interlaced graphics format, meaning that the same display alternates the stereo-pair images in rapid succession.

3.1.3.1 Monitor-Based Large-Volume Displays. The smallest large-volume stereo display uses active glasses in connection with one stereo-ready monitor. As illustrated in Figure 3.12a, each user wears a set of shutter glasses and looks at the monitor. The stereo-ready monitor is of a special design that is capable of refreshing the screen at double the normal scan rate, or between 120 and 140 scans/sec. A computer sends two alternating, slightly offset images to the monitor, as illustrated in Figure 3.12b. An infrared (IR) emitter located on top of the CRT display is synchronized with the RGB signal and controls the active glasses in a wireless mode. The IR controller directs orthochromatic liquid crystal shutters to close and occlude one eye or the other alternately.



(a)



(b)

Fig. 3.12 Single CRT-based large-volume display. (a) Active stereo glasses. From Burdea [1993]. Reprinted by permission. (b) Communication diagram. From Burdea and Coiffet [1993]. © Editions Hermès. Reprinted by permission.

In this way the brain registers a quick sequence of right- and left-eye images and fuses the two by stereopsis (thus the depth perception effect).

The extremely short close/open time of the shutters (a few milliseconds) combined with the 120-Hz refresh rate of the monitor results in a flicker-free image. This image is much sharper than that of LCD-based HMDs (the graphics resolution of the CRT is preserved) and is not tiring even for long simulation times. However, the shutters filter out a portion of the light, so the images as seen by the user will appear less luminous than on a normal screen. Furthermore, there needs to be direct line of sight between the IR emitter and the IR receiver integrated with the active glasses. Otherwise the stereo glasses stop their shutters and the stereopsis is lost.

Stereo glasses allow the user to see static stereo scenes, as the image will not change to account for the user's viewing angle. For some VR applications it is better to change the image according to the user's viewing direction, much like HMDs do. Therefore a head tracker can be added to the system. This can be an ultrasonic head tracker, as illustrated in Figure 3.12b, or of some other type. The tracker emitter is then placed on top of the monitor near the infrared controller and the active glasses modified to incorporate the tracker receiver (in our case three ultrasonic microphones). The tracker controller converts sound phase difference into head 3D information and transmits it at up to 50 datasets/sec to the graphics computer. Software uses the head 3D position data to recalculate the virtual scene and the simulation loop is closed. Additional input devices such as sensing gloves or trackballs can be added to the system in order to increase the VR simulation interactivity.

The direct-line-of-sight requirement of ultrasound trackers (see the discussion in Chapter 2) is easily satisfied, as the user looks at the screen during the simulation. However, when several users look at the CRT monitor, only one person has control over the virtual scene perspective, as only one head tracker exists. The person controlling the simulation view has to be within the tracker active area, while the other participants can look from further away.

The immersive sensation felt when wearing tracked stereo glasses is different than that of HMDs [Akka, 1992]. When wearing HMDs users feel surrounded by the virtual world. With stereo glasses the user views the CRT

display as a window into a virtual world. As with a real window, the user expects the scene to change as his or her position relative to the display changes. Human factors studies reported by Akka showed that it is better to exaggerate the image response to the user's head motion, since the screen fills a smaller portion of the field of view than HMDs do. If the user is sitting at 45 cm from a monitor that has 30-cm-wide display, then the display fills only 37° of the user's 180° horizontal field of view. Implementing an exaggeration factor to affect the field of view requires an estimated default user-to-screen distance u (for example, 30 cm). If the head tracker detects a different distance u from the user's head to the display, then the projection distance should be multiplied by a factor k given by [Akka, 1992]

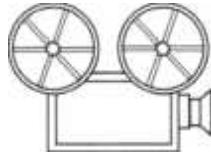
$$k=r(u-U)+U \quad (3.1)$$

where U is the default distance and r is the responsiveness factor. An r of 1.25 is an optimal responsiveness factor for both field of view angle and image rotation. Since r amplifies equally the noisy 3D tracker data, jitter will be increased at the edges of the work envelope.

Stereo glasses have transmittance rates between 29% and 32% for the high-end versions. This means that about two thirds of the brightness of the CRT image is not perceived by the user. Furthermore, they can only work within the range of the IR transmitter, which is 4-12 ft, depending on the model. On the positive side, stereo glasses are light (60-100 grams) and can be worn on top of regular glasses. They automatically switch to 100% transmittance when the user looks away from the stereo monitor. Prices of stereo glasses vary between the high-end CrystalEyes3, costing \$800 [StereoGraphics, 2001], and the consumer-grade 3D Glasses, which cost \$100 [I-glasses, 2001]. Shutter glasses that cost under \$100 are also available. These get the synchronization signal through a wire, such that the IR emitter is eliminated and the user's freedom of motion reduced compared to wireless models.

The user's FOV grows with the reduction in the distance to the monitor and with the size of the screen. The distance to the monitor cannot be too short, else eye strain and discomfort occur due to lost focus. The alternative

is to use larger monitors, or several monitors placed side by side. Recently vendors have replaced CRTs with flat monoscopic multipanel displays integrated in a single structure [Isdale, 2000], as shown in Figure 3.13. The PV290 display shown here uses three TFT panels, each with a resolution of 1280 x 1024 pixels [Panoram Technologies, 2000a]. The result is a composite image size of 111 x 29 cm² and a resolution of 3840 x 1024 pixels. The primary RGB input of each panel is connected to one of multiple graphics pipelines on the computer running the simulation (such as an HP J5000 workstation). A secondary RGB input is provided to accept images from VCRs, DVDs, laptops, and networked computers, with the input selection done externally through a USB bus. All computer/video cables are connected to a control box, and a single umbilical cable is connected to the back of the PV290 display. This is done in order to avoid cluttering the user's workspace with a myriad of individual cables.



VC 3.1

Providing a single video feedback obtained by tiling multiple displays requires proper synchronization. Unsyncronized and unshielded CRTs placed side by side will produce imaging artifacts due to the mutual interference of their magnetic fields. Even LCD displays can produce image misalignments, as illustrated in Figure 3.14a [Quantum 3D, 2001]. This cockpit view simulation is output by three graphics pipelines that are asynchronous. During the simulation of a fast tilting maneuver the misalignment becomes apparent in the discontinuity of the cloud area of the scene. Fortunately, both software and hardware methods exist for image synchronization and generation locking (genlock) tiled video displays. When these methods are applied, the result is a composite, large-screen visual channel, as illustrated in Figure 3.14b. More details on multigraphics pipeline synchronization are given in Chapter 4.

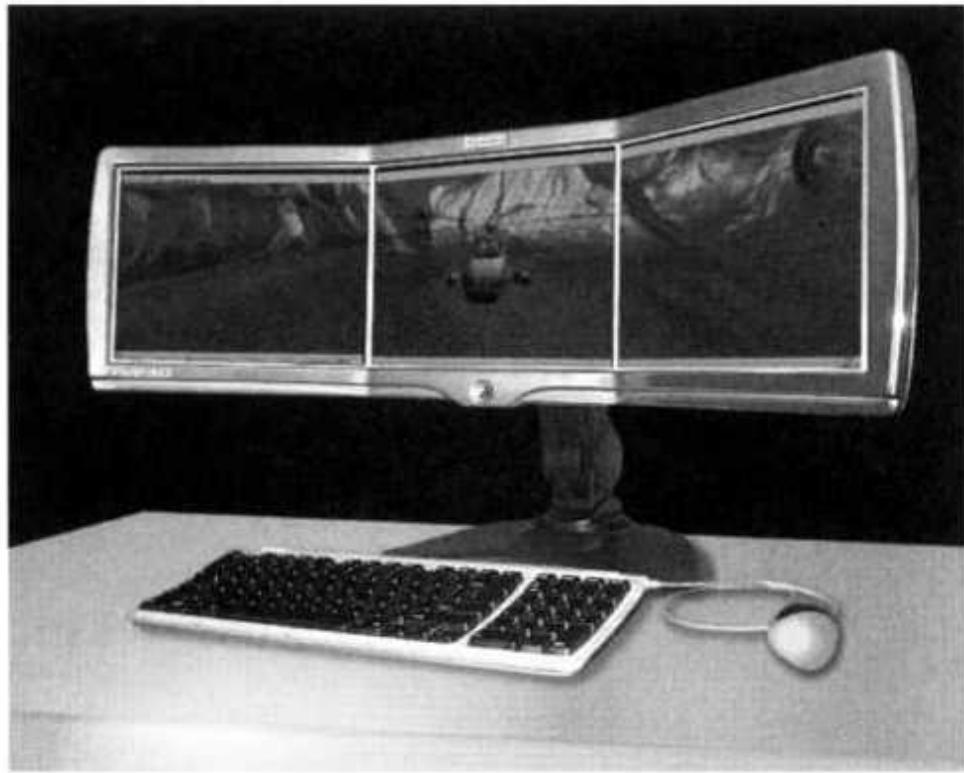


Fig. 3.13 The PV290 three-panel display. Courtesy of Panoram Technologies.
© 2003 Panoram Technologies Inc. All rights research.

3.1.3.2 Projector-Based Displays. These represent the solution of choice for allowing many closely located users to participate in a VR simulation. Such graphics displays have traditionally incorporated CRT projectors to produce the stereo-pair image. A CRT projector uses three tubes (R, G, B) to produce a high-resolution (1280 x 1024 pixels) image at 120 Hz [Barco, 1999a]. When operating in framesquential stereo mode, the projector splits the number of scan lines in two, and the user wearing active glasses sees a stereo image refreshed at 60 Hz. Special "fast green" coating is needed for the phosphor used in the CRT tube to reduce its visual persistence, otherwise the stereo effect is lost since both images will be projected at the same time.

CRT projectors have a number of drawbacks related to cost and their inability to project bright images. High-end CRT projectors have luminance on the order of 200-300 lumens, such that they are adversely affected by

ambient light. This requires light-tight enclosures (or drapes) to be constructed around the display, consuming additional space and resources. More recently digital projectors have started to replace the older CRT ones, as they have an order-of-magnitude higher luminance. The key component of a digital projector is the digital micromirror device (DMD) [Younse, 1993] developed by Texas Instruments Inc. The DMD consists of a chip with an array of very small (16 gm) aluminum mirror elements, as illustrated in Figure 3.15a. Each of the micromirrors is supported by two posts and torsion hinges to float above the chip surface in a matrix arrangement. A 5-V potential applied at the chip RAM address will electrostatically deflect the corresponding micromirror by ± 100 depending on the voltage sign. Figure 3.15b illustrates three such chips integrated in a digital projector [Lubell, 2000]. A high-intensity light source is first sent through a combination of prisms to split its three primary colors. Each colored light beam is then bounced off a DMD chip and then recombined to form the virtual scene imagery. Each DMD is driven by frame signals received from the graphics computer, modulating the intensity of the R, G, B subpixels. Higher pixel light intensities means less micromirror deflection and therefore less voltage applied to a DMD array element. The result is a compact projector that outputs a high-resolution (1280 x 1024) and very bright (thousands of lumens) image. Two such projectors can be stacked and synchronized to produce a high-resolution stereo image.

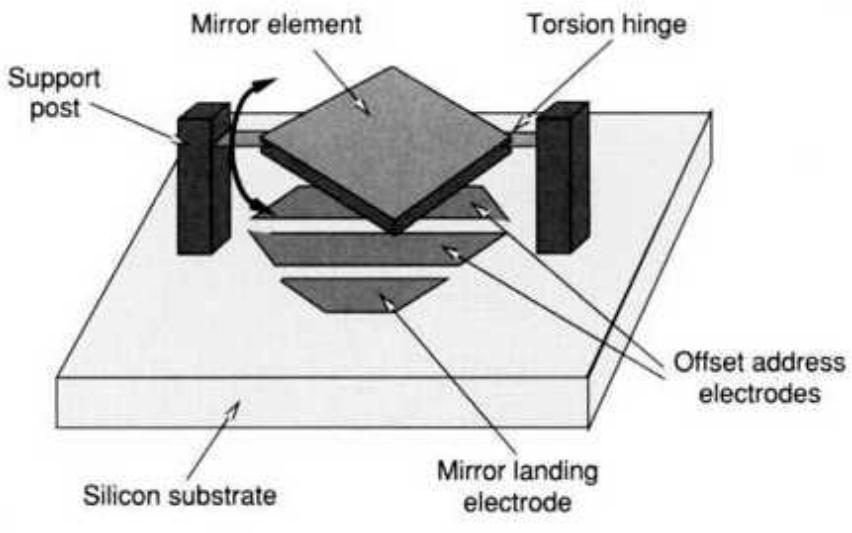


(a)

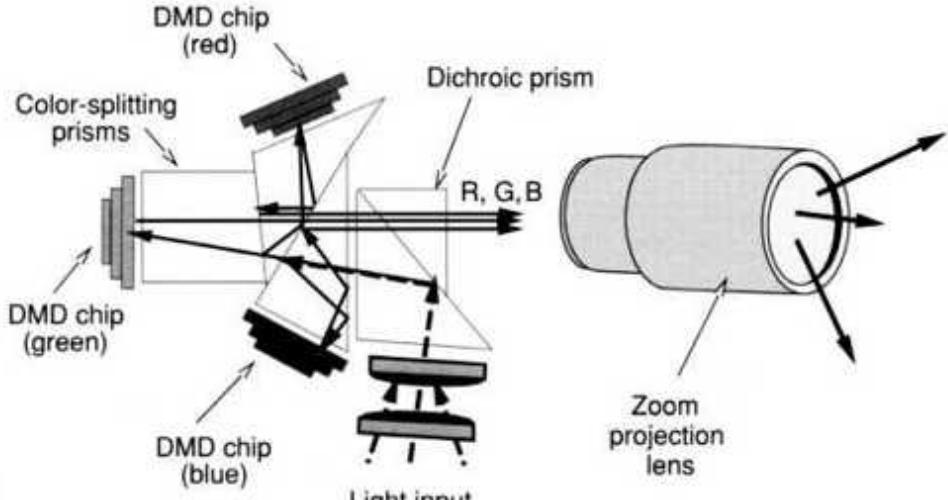


(b)

Fig. 3.14 Three-channel out-the-window display: (a) image misalignment during the simulation of a fast tilting maneuver; (b) correctly aligned images. From Quantum3D [2001]. Reprinted by permission.



(a)



(b)

Fig. 3.15 Digital Micromirror Device (DMD) display: (a) the micromirror element. Adapted from Younse [1993]; (b) the Texas Instruments DMD-based projector. Adapted from Lubell [2000]. © 1993, 2000 IEEE. Reprinted by permission.

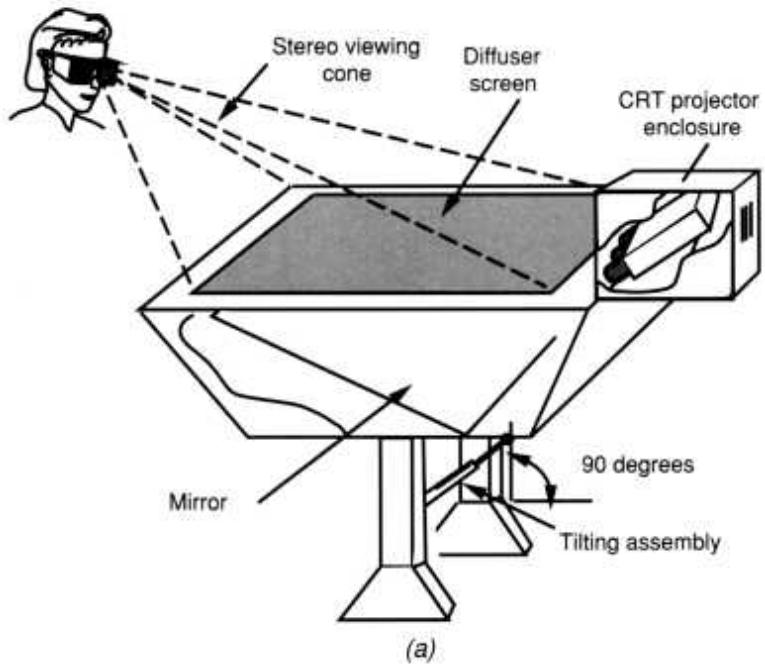
One of the first projector-based large-volume displays was the Immersive Workbench [Fakespace Systems, 1998]. The workbench display, invented by the German National Computer Science Institute, projects a 3D scene on a large horizontal tablesize display. The image is output by a CRT projector, bounces off a mirror, and is reflected upward through a diffuser screen, as

illustrated in Figure 3.16a. The first Fakespace version used a single Electrohome CRT projector placed horizontally outside the 0.91 m x 2.54 m x 3.65 m workbench wooden enclosure. This required additional floor space and draping to prevent outside light from being reflected off the workbench mirror. Modern workbenches, such as the Baron [Barco, 1999b], incorporate the CRT projector in the workbench enclosure itself.

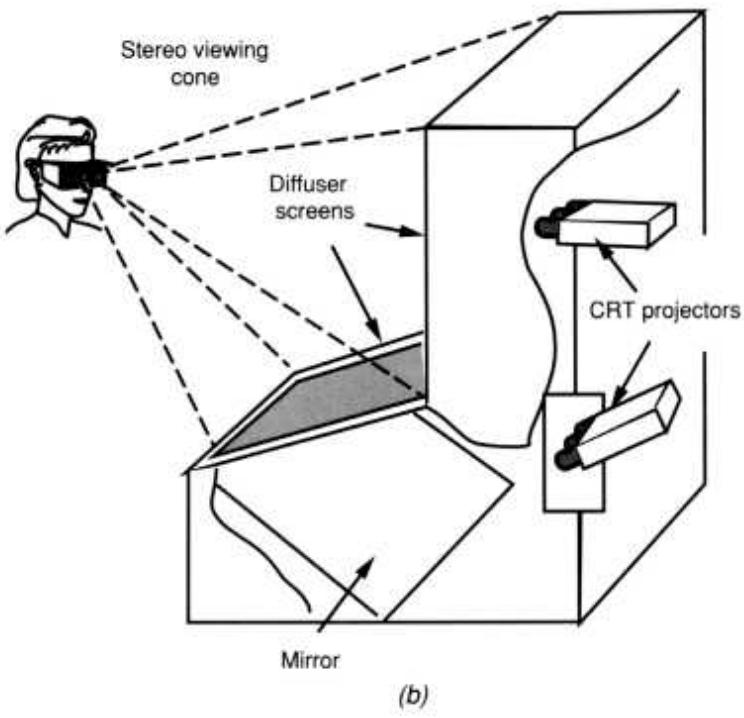
Several viewers wearing active glasses and sitting around the display can simultaneously see 3D objects floating on top of the workbench in a stereo viewing cone. This cone depends on the position of the user and the degree of tilting of the workbench surface. If the workbench is horizontal, tall 3D objects will be clipped at the side opposite the user (the so-called stereo collapse effect). Modern workbenches therefore incorporate a tilting mechanism (manual or motorized) to allow the user to control the workbench angle based on application needs. The workbench can thus be tilted anywhere from horizontal to vertical positions. An alternative is the so-called L-shaped workbench, such as the V-Desk 6, illustrated in Figure 3.16b [Trimension Systems Ltd., 2001]. Instead of using a motorized table, the V-Desk 6 incorporates two fixed screens and two CRT Barco projectors. The top projector is aimed toward the vertical screen, while the second projector image is bounced off a mirror placed at the lower part of the display. The result is a very compact enclosure (1.48 m x 1.87 m x 2.05 m) that creates a large stereo viewing volume, allowing a small number of users to interact with the 3D scene. The V-Desk 6 is a turnkey VR system since it incorporates IR controllers, active glasses, InterSense trackers for the user's head and hands, as well as a sound system. When several users view the stereo scene simultaneously, correct perspective is only provided to the primary user, who is wearing the head tracker. The other (secondary) users will then see visual artifacts (such as tilting of tall virtual buildings), depending on the head motion of the primary user. When tried by the authors, the large weight of the InterSense tracker mounted on the rim of the active glasses became very noticeable.

Another type of projector-based display is the CAVE, which was invented at the Electronic Visualization Laboratory of the University of Illinois at Chicago [Cruz-Neira et al., 1993]. It consists of a 10-ft cubical structure, as

illustrated in Figure 3.17 [Pape et al., 1997]. The version shown here has four CRT projectors, one each for the front, the left and right sides, and the floor. Each projector is driven by a different graphics pipeline signal received from a four-pipe computer. The three vertical panels use retroprojection, with the projectors placed on the surrounding floor and reflecting the image off mirrors. The image shown on the floor display is produced by a projector placed on top of the CAVE, and is reflected downward by a mirror assembly. This mirror is oriented to create shadows behind the users rather than in front, in order to minimize visual artifacts. The side and floor panels are tightly assembled to reduce discontinuities at the seams and the projectors are synchronized to reduce flicker. Users wearing active glasses see a very convincing 3D scene, including objects that seem to grow upward from the floor. The size of the CAVE accommodates up to 12 users at a time, as long as they stay still or move together. Commercial versions of the CAVE include the Immersive WorkRoom (4.0 m wide x 10.7 m deep x 7.6 m high) from Fakespace Systems and the smaller ReActor (3.0 m wide x 2.9 m deep x 2.8 m high) from Trimension Systems Ltd. A newer version called RAVE (for "reconfigurable virtual environment") offers additional flexibility since it is constructed of four modules. Each 3.0-m-wide x 2.9-m-deep x 3.7-m-high module has its own screen and projector assembly. This gives users the added flexibility of changing the viewing configuration from a cube, to two L-shaped displays, to a wall-like configuration, based on application-specific needs.

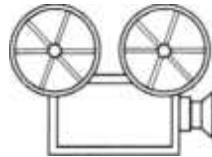


(a)



(b)

Fig. 3.16 Workbench displays. (a) Single-projector tilting configuration. Adapted from Barco Baron [1999b]. Reprinted by permission. (b) Dual-projector L-shaped configuration. Adapted from Trimension Systems Ltd. [2001]. Reprinted by permission of SEOS Ltd.



VC 3.2

CAVE's cost (excluding the high-end multipipe graphics workstation) is about \$300,000, while the RAVE costs about \$500,000. It is therefore not surprising that various research groups around the world are working on less expensive CAVE variants. One example is Future Lab's PC CAVE developed in Austria [Delaney, 2001]. Researchers replaced the high-end SGI computer used in the classic CAVE with a cluster of Linux PCs with NVIDIA graphics cards (discussed in Chapter 4) and other commodity hardware components. This reduced the overall system cost to less than \$100,000 without significant performance loss compared to the traditional SGI highend system [Belleman et al., 2001].

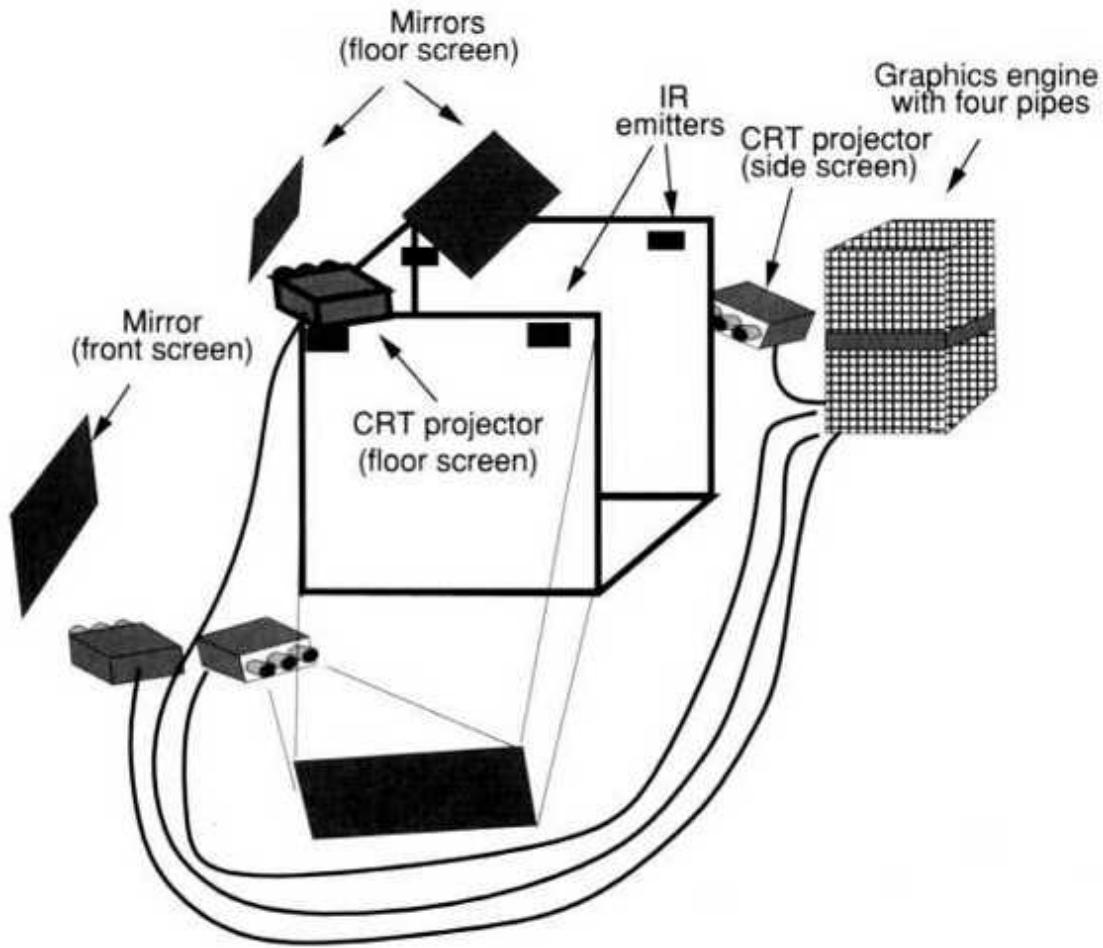


Fig. 3.17 The four-projector CAVE® display. Adapted from Pape et al. [1997]. Reprinted by permission of Electronic Visualization Laboratory, University of Illinois at Chicago.

CAVE-like displays become too small for more than a dozen users. Graphics displays that produce large enough images to accommodate many more viewers are walls and domes. Such displays have image dimensions of 7.11 m x 2.13 m for the PanoWall [Panoram Technologies, 2000b], 7.32 m x 2.44 m for the WorkWall [FakeSpace Systems, 1999], and an even larger diameter for the V-Dome [Trimension Systems Ltd., 2000]. In order for a single projector to cover such large image surfaces, it needs to be moved far from the screen, which in turn reduces image brightness substantially. Furthermore, the larger the screen, the lower is the ratio of pixels/surface area that the eye sees, which means increased image granularity. If a single projector with a resolution of 1280 x 1024 is used with the PanoWall, then

this ratio is 8.6 pixels/cm². Such an image resolution may be inadequate for visually demanding applications typically associated with wall-size displays. In order to simultaneously satisfy the requirements of large image size, image brightness, and high resolution, it is necessary to use projector arrays.

Definition A projector array is a large-volume display formed of a number of functionally identical projectors calibrated to display a tiled composite image of the virtual scene.

The PanoWall projector array, illustrated in Figure 3.18, incorporates three projectors placed side by side behind the screen. The compound image of the three 1280 x 1024 pixel projectors has 3200 x 1024 pixels (subtracting the overlap). This is 2.5 times better resolution and much higher luminosity than possible with a single projector. In order to preserve image quality it is necessary to carefully synchronize the three projectors and assure color uniformity and brightness. Calibration is controlled through a PC (laptop) that communicates over a serial line with a processor array.

Very important for the image quality is the blending of the array images, which is done by overlapping approximately 25% of adjacent image tiles. Unfortunately, this approach increases luminosity in the overlapping zones to twice the brightness of the surrounding ones. In order to compensate for this undesirable visual artifact, the image coming from the multipipe graphics computer (or from an image generator array) is preprocessed. The array processor used by the PanoWall partially dims the pixel intensities before sending the image to the projector array. Figure 3.19a [Panoram Technologies, 2001] shows a tiled image produced by an array of four projectors. It is clear that the overlap zones from two projectors have twice the brightness of their surrounding pixels. The problem is even worse for the central zone (the so-called four-corner region), which has four times the brightness of the rest of the image. To deal with this, Panoram Technologies developed a seamless matrix blending approach, which uses a two-dimensional look-up table to adjust the intensities of each tile pixel. This way the array processor can adjust pixel intensities and produce the uniform brightness seen in Figure 3.19b. Since each projector has its own visual

signature, the array processor uses projector-specific look-up tables for image blending computations.

Domes are large, spherical displays that produce a 360° FOV surrounding an audience that can be as large as 400-500 viewers. To produce such imagery, domes rely on multiple projectors arranged radially around a semispherical rear-projection screen [SEOS, 2000]. The number of projectors depends on the application and the dome dimensions. The V-Dome, used in high-tech theaters, for example, incorporates seven Barco projectors (1280 x 1024 pixels each). These projectors need special optics to pre-distort the image prior to projection on the curved screen. Five of the projectors are peripheral, while two are placed at the cap of the dome, which has a total projection area of 639 m². The price of such a display (excluding the image generator) is over 2 million dollars. Domes used in military flight training can have more than 20 projectors owing to the need for increased image resolution.

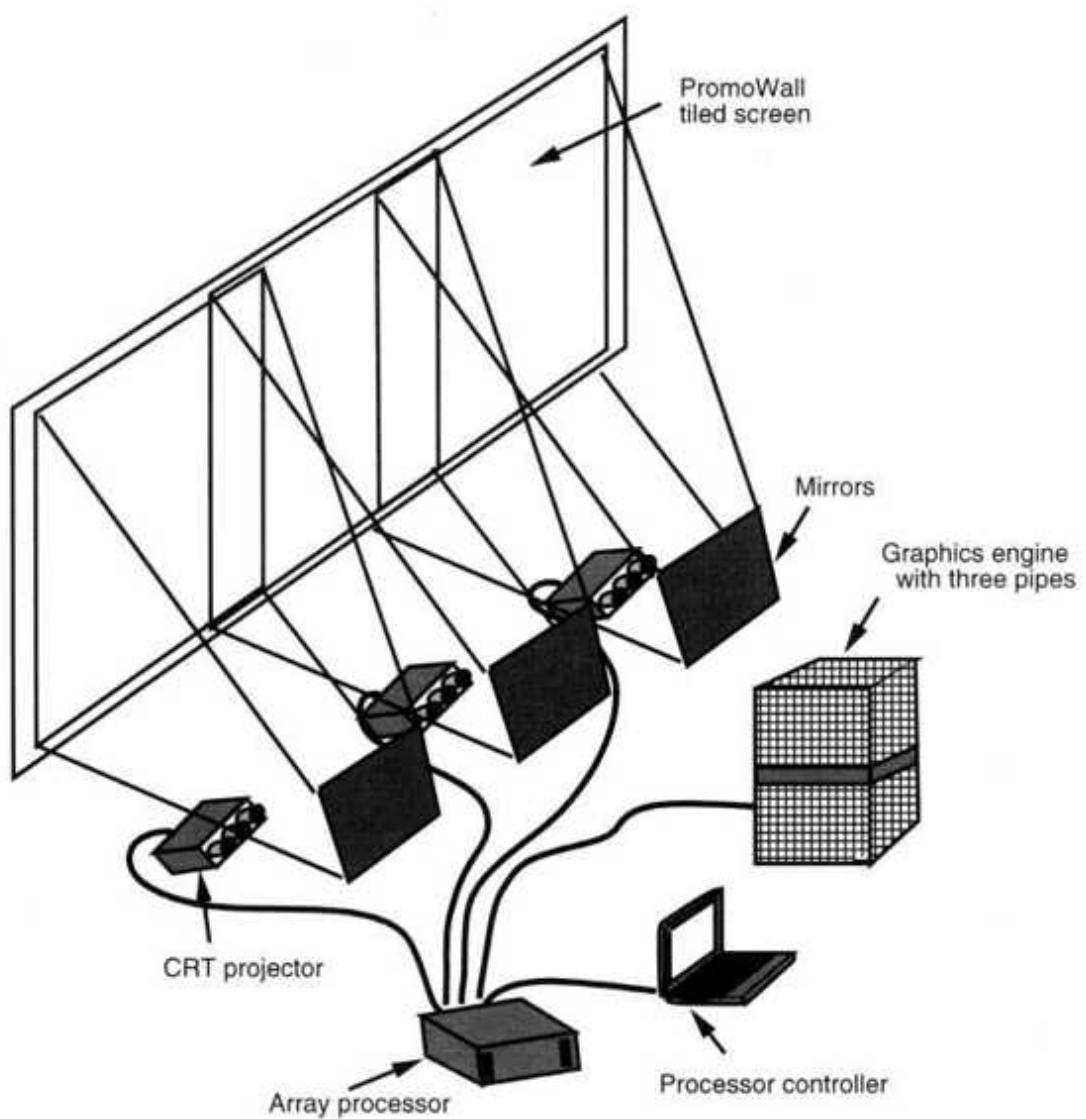


Fig. 3.18 The three-projector PanoWall display. Adapted from Panoram Technologies [2000b]. © 2000 Panoram Technologies, Inc. Reprinted by permission.

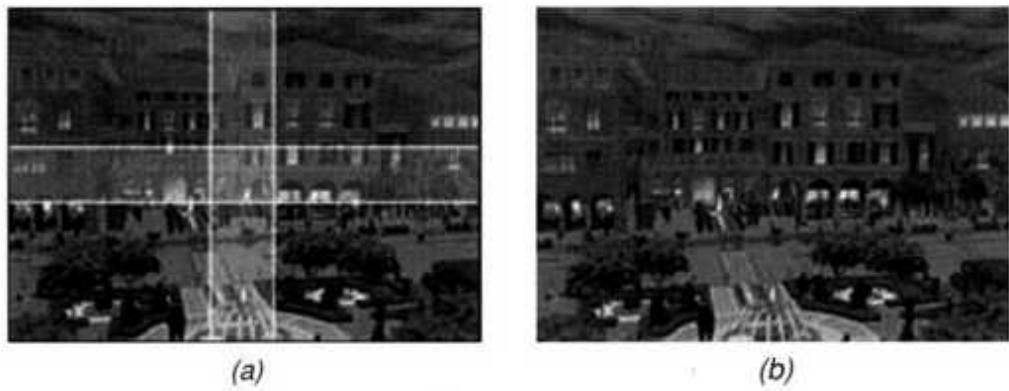


Fig. 3.19 Tiled image produced by a four-projector array: (a) overlap regions; (b) edge blending result. From Panoram Technologies [2001]. © Panoram Technologies, Inc. Reprinted by permission.

Smaller projector-based displays rely on field-sequential graphics, IR emitters, and active glasses to produce stereo scenes. Large theater-like domes generally display monoscopic graphics, relying on the large 360° FOV to immerse the audience. An alternative is to use pairs of polarized projectors and provide each viewer with inexpensive polarized glasses. These "passive" stereo glasses have lenses polarized differently for the right and left eyes. Therefore each eye sees the image component of the stereo pair with matching polarization. The brain then integrates the two images for the stereo effect. Passive glasses cost about \$1 each, clearly a less expensive solution for large groups of people looking at a stereo image than using active glasses.

Table 3.2 summarizes the characteristics of the large-volume graphics displays discussed in this chapter. The resolution is given in million pixels/m² in order to eliminate the dependence on display shape. Listed prices do not include the computer driving the large-volume display. It becomes apparent from this data summary that prices increase significantly with the size of the display, while at the same time the resolution drops by orders of magnitude. Increasing the resolution of very large displays to that of workstations would involve many more projectors and an even larger cost than those given in Table 3.2.

TABLE 3.2. Comparison of Various Large-Volume Graphics Displays

Manufacturer, Display Name	Type	Resolution (10^6 pixels/m 2)	Image Size	Number of Users	Price ^a (10^3 \$)
StereoGraphics CrystalEyes3	Active glasses	18.2	0.36×0.2 m 2	~4	2.6
Panoram PV290	3-panel monitor	12.2	1.11×0.29 m 2	~3	23
Barco Baron	Tilt workbench	1.9	1.36×0.71 m 2	~4	80
Trimension V-Desk 6	L-shaped workbench	1.0	1.36×1.73 m 2	~4	173
Fakespace Immersive WorkRoom	4-wall CAVE	0.1	$3.0 \times 3.0 \times 4$ m 3	12	300
Fakespace RAVE	Modular CAVE	0.2	$2.3 \times 2.4 \times 4$ m 3	Variable	500
Panoram PanoWall	Wall (3 projectors)	0.2	7.11×2.13 m 2	Variable	300
Trimension V-Dome	Dome (7 projectors)	0.009	21 m diameter	400	2172

^aPrice does not include the computer driving the simulation.

3.2 SOUND DISPLAYS

Definition Sound displays are computer interfaces that provide synthetic sound feedback to users interacting with the virtual world. The sound can be monoaural (both ears hear the same sound) or binaural (each ear hears a different sound).

Sound displays play an important role in increasing the simulation realism by complementing the visual feedback provided by the graphics displays previously discussed. Assume a user is looking at a virtual ball bouncing in a virtual room that is displayed on a CRT monitor. The user's mind says that he or she should also hear a familiar "plop-plop-plop" sound. When sound is added, the user's interactivity, immersion, and perceived image quality increase. In this scenario simple monoaural sound is sufficient, as the ball is always in front of the user and being displayed by the monitor. Let us now consider another user, who looks at the same virtual world, this time

displayed by an HMD. If the ball bounces away, outside the field of view (as balls in the real world sometimes do), then the user cannot tell where the ball went based on visual information or on monoaural sound alone. Now the simulation needs a device that provides binaural sound in order to localize the "plop-plop-plop" sound in 3D space relative to the user.

This example illustrates an important distinction within the sound feedback modality. Highly immersive virtual reality simulations should have 3D sound, also called virtual sound, in addition to graphics feedback. The distinction between 3D sound and stereo sound is illustrated in Figure 3.20 [Burdea and Coiffet, 1993]. Stereo sound on headphones seems to emanate inside the user's head. In other words it is not externalized, as real sounds are. When the user wears simple stereo headphones, the violin will turn to the left following the user's head motion. A 3D sound presented on the same headphones or on speakers contains significant psychoacoustic information to alter the user's perception into believing that the recorded sound is actually coming from the user's surroundings [Currell, 1992]. In Figure 3.20, the 3D sound is synthesized using head tracker data, and the virtual violin remains localized in space. Therefore its sound will appear to move to the back of the user's head. Sound in a real room bounces off the walls, the floor, and the ceiling, adding to the direct sound received from the violin. The realism of the virtual room therefore requires that these reflected sounds be factored in.

3.2.1 The Human Auditory System

Three-dimensional sound displays cannot be effective without an understanding of the way we localize sound sources in space. Humans perceive sound through vibrations arriving at the brain via the skeletal system or via the ear canal. Within the scope of this book we are interested in the ear's ability to detect the position of a sound source relative to the head.

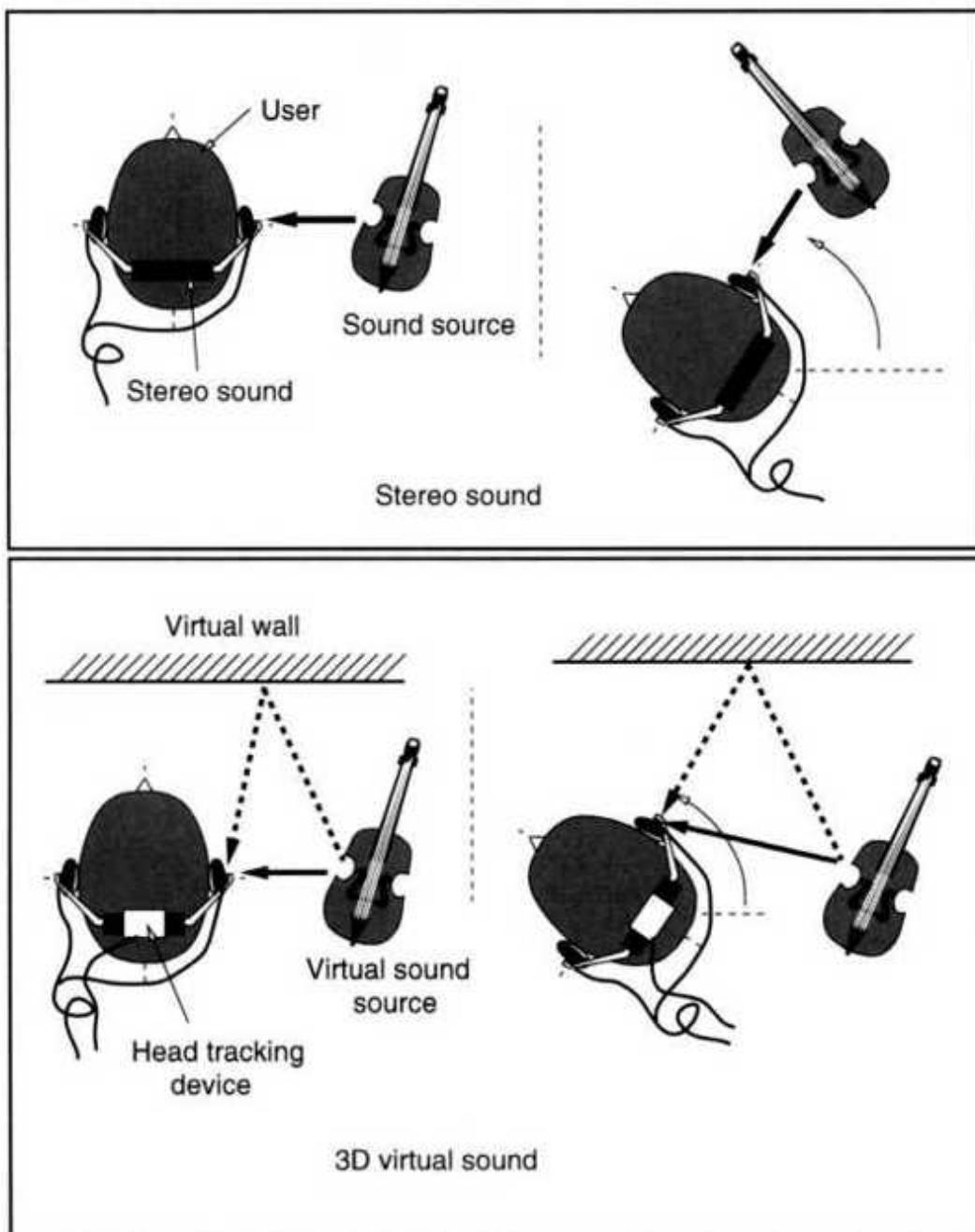


Fig. 3.20 Stereo sound versus 3D virtual sound. From Burdea and Coiffet [1993]. @ Editions Hermès. Reprinted by permission.

3.2.1.1 The Vertical-Polar Coordinate System. In order to measure position it is necessary to establish a head-attached system of coordinates. If such a system of coordinates is Cartesian, then the location of a sound source relative to the head can be expressed by the triad (x , y , z). Alternatively, the

same 3D sound source position can be expressed in a spherical system of coordinates called the vertical-polar coordinate system [Duda, 1997]. As illustrated in Figure 3.21, the sound source location is uniquely determined by three variables, namely azimuth, elevation, and range. The azimuth is the angle θ between the nose and a plane containing the source and the vertical axis z. The source elevation is the angle ϕ made with the horizontal plane by a line passing through the source and the center of the head. Finally, the range r is the distance to the source measured along this line. The sound source azimuth can vary anywhere between $\pm 180^\circ$, while the elevation has a range of $\pm 90^\circ$. Finally, source range can only be positive and larger than the head radius (assuming a simplified spherical head model). The brain estimates the source location (azimuth, elevation, and range) based on intensity, frequency, and temporal cues present in the sound perceived by the left and right ears.

3.2.1.2 Azimuth Cues. Since sound has a fixed propagation velocity in air, it will reach first the ear that is closer to the source. As illustrated in Figure 3.22a, the sound wave reaches the right ear last, since it has to travel an extra distance of $a\theta + a \sin \phi$. The difference in the arrival time of the sound at the two ears, called the interaural time difference (ITD), is given by

$$\text{a ITD} = -(9 + \sin \theta) C \quad (3.2)$$

where a is the head radius, c is the speed of sound (about 343 m/sec), and θ is the source azimuth. This time difference is maximum when θ is 90° and zero when the source is directly in front of or directly behind the head.

A second cue used by the brain to estimate the source azimuth is the intensity of sound reaching the ears, the so-called interaural intensity difference (IID). As illustrated in Figure 3.22b [Wenzel, 1992b], the closest ear hears a sound with higher intensity than that reaching the distant ear. This phenomenon, called the head-shadow effect, is detectable for sounds with high frequencies (above 1.5 kHz) [Kalawsky, 1993]. For lower frequencies the ITD dominates for azimuth localization as long as the source is not far from the user. For very low frequency sound (below 250 Hz), however, the room reverberations confuse the auditory system and ITD

cannot be estimated. In that case the brain adapts by focusing on the starting of the sound transients and discarding the low-frequency steady-state information.

3.2.1.3 Elevation Cues. If the head is modeled as having simple holes for ears, than there are source locations in space where the time and intensity cues are the same. The so-called cones of confusion result in perceptual reversal, or front-back confusion. A source that is actually behind the user is perceived as being in front, or vice versa. In reality ears are not simple holes, on the contrary, the outer ear, or pinna, is very important, as sound is reflected by it and guided to the inner ear [Duda,1997]. As illustrated in Figure 3.22c, sound coming from a source located above the user's head has quite a different reflection path than sound coming from a source in front of the user. Some frequencies are amplified and others attenuated. This attenuation is due to interference between the direct sound and that reflected by the pinna. Since the path difference between the direct and pinna-reflected sound changes with the elevation angle, the pinna provides the primary cue for source elevation. The user's face and shoulders geometry also influences the way the sound is reflected toward the outer ear.

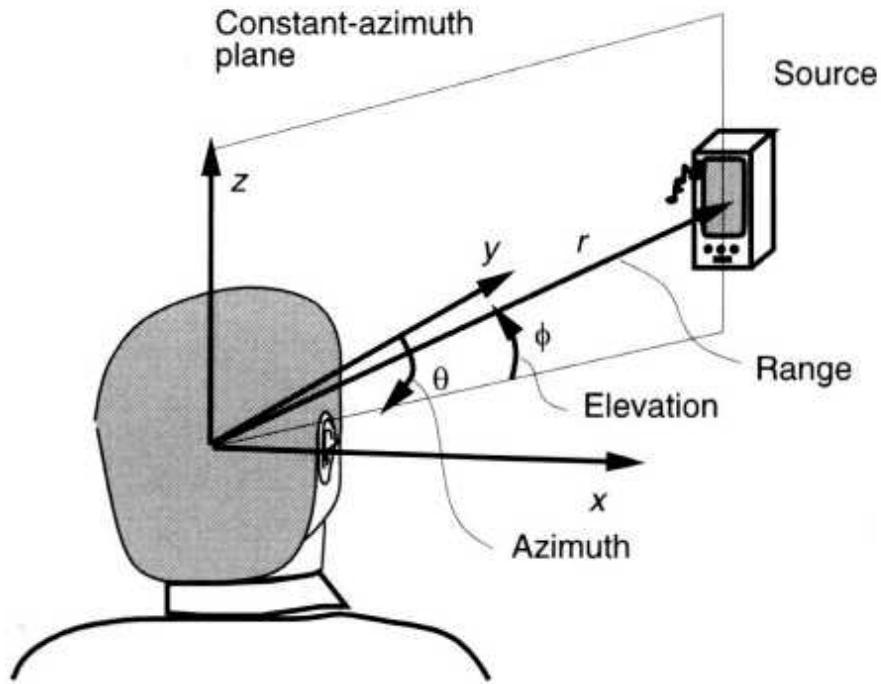


Fig. 3.21 The vertical-polar coordinate system used to locate 3D sound sources.

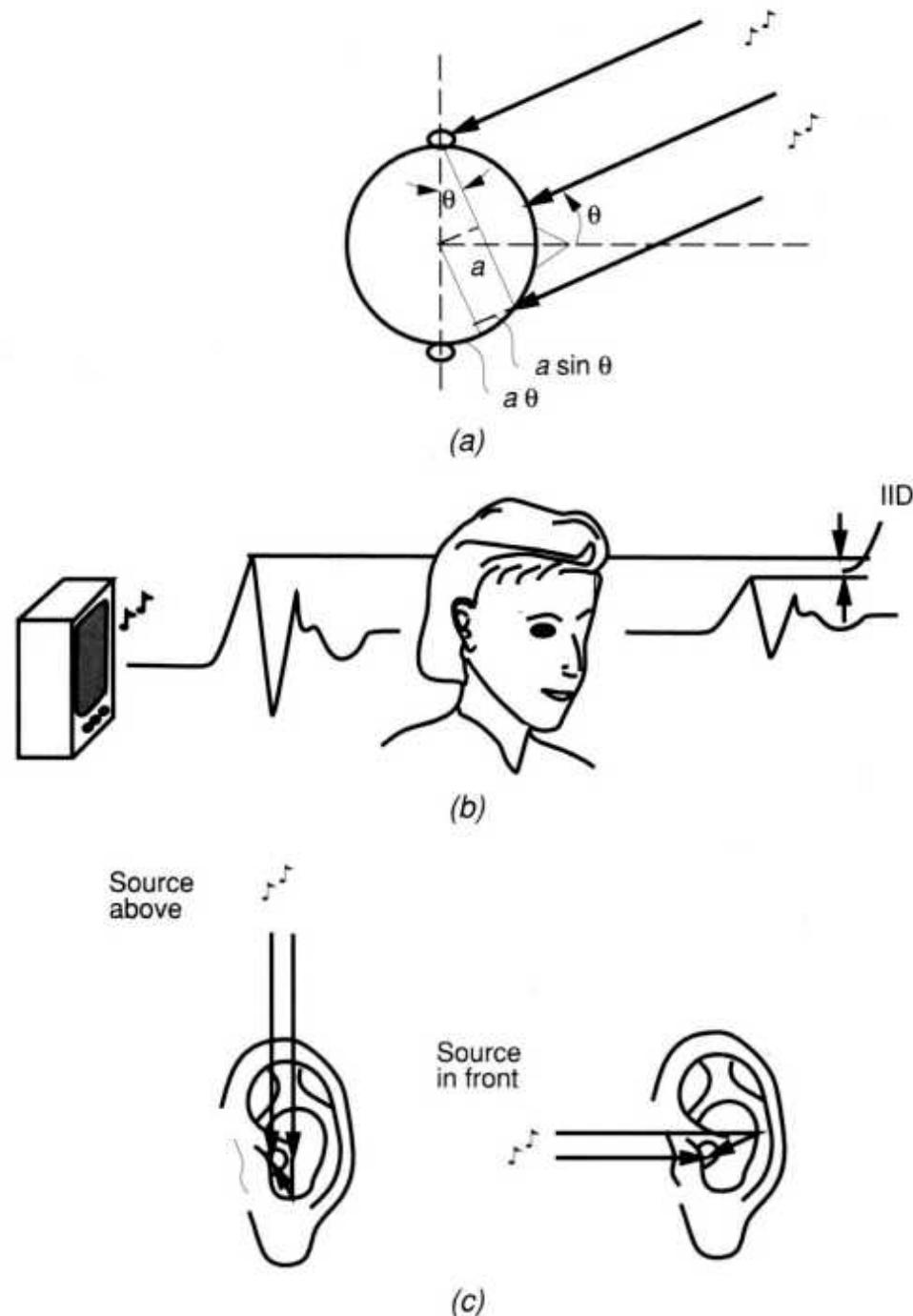


Fig. 3.22 Sound location cues. (a) Interaural time difference. Adapted from Kalawsky [1993]. © Addison-Wesley Publishers Ltd. 1993, reprinted by permission of Pearson Education Ltd. (b) Interaural intensity difference.

Adapted from Wenzel [1992b]. Reprinted by permission of NASA Ames Research Center. (c) Sound path variation with source elevation.

3.2.1.4 Range Cues. Prior knowledge of a given sound source combined with its perceived loudness are used by the brain to estimate range (or distance) from the user. A normally high-energy sound source, such as a siren, when perceived as a faint sound is interpreted as being distant. Conversely, when a normally faint sound, such as a whisper, is heard, it is interpreted as coming from someone close by. Another range cue is motion parallax, or the change in sound source azimuth when the user is translating his or her head. Large motion parallax indicates a source nearby (small range). Sources that are far away will have no change in azimuth in response to head translations. Yet another major range cue is the ratio of sound coming directly from the source versus that which is first reflected off the user's surroundings (walls, furniture, floor, etc.). The energy of the direct sound drops off with the square of the source range. However, the energy of the reflected sound does not change much with range. Therefore sound sources at distances greater than 1 m produce a small ratio of direct versus reverberated sound. These sources are said to be in the far field. Sources at small range from the user (near field) have a large ratio of direct versus reverberated sound.

3.2.1.5 Head-Related Transfer Function. Three-dimensional sound hardware design assumes that the location of the sound source is known. It then needs a model of the corresponding sound reaching the inner ear. The modeling task is complicated, however, by phenomenon multidimensionality, individual anatomical variations, and our partial understanding of the auditory system. An alternate approach is to place people inside a dome with spatially localized sound sources (loudspeakers) and fit them with miniature microphones placed close to the eardrum [Wenzel, 1992a]. When the loudspeakers are turned on in sequence, the microphone output is stored and digitized. It is thus possible to measure the response in the form of two functions (one for each ear), called head-related impulse responses (HRIRs). The corresponding Fourier transforms, called head-related transfer functions (HRTFs), capture all the physical cues used in source localization. As discussed in the foregoing, HRTFs depend on the source

azimuth, elevation, range, and frequency. For sources in the far field, HRTFs depend only on azimuth, elevation, and frequency. Each person has his or her HRTF signature, as no two persons have exactly the same outer ear and torso geometry.

3.2.2 The Convolver

Once a user's HRTF is experimentally determined, it is possible to take any sound, apply the pinna transforms as finite impulse response (FIR) filters, and play back the sound to the same user over a headset. The user then has the sensation of hearing that sound as coming from a virtual speaker placed in space accordingly. This signal processing technique, called convolving, was developed by researchers at NASA, and works well to produce 3D sound. Tests showed that subjects had a very high recognition rate, especially when hearing the sounds convolved using their own HRTFs. However, when somebody else's HRTF was used, the spatial recognition rate dropped (as if a user was "listening through another person's ears"). As reported by Wenzel [1992b], the incidence of front-back reversals increased from 5% when one listens to real sound sources to 10% when one listens to sound synthesized based on one's own HRTF. Errors increased further to 30% when somebody else's FIRs were used.

At the moment it is not practical to design a 3D sound processor customized for each person's HRTF. The compromise is to design hardware around some generic HRTF and accept the resulting inaccuracies. Such 3D sound generation hardware is also needed due to the high computational load involved and the real-time requirement of VR simulations. Foster [1992] estimated that it takes 30-50 MIPS for each sound source image. Therefore a computation model of an environment containing six walls and four sound sources requires about one billion computer operations per second [Foster, 1992; Chapin and Foster, 1992]. Such computations require dedicated hardware to assure good overall system response. Otherwise the CPU will saturate, which in turn will degrade the graphics pipeline throughput (as discussed in the next chapter).

The first virtual 3D sound generator was developed by Crystal River Engineering under contract to NASA in 1988 [Foster, 1988]. This real-time

digital signal processor, called appropriately the Convolvotron, consisted of a set of PC-compatible dual cards placed in a separate enclosure. Current convolvotrons are much more compact owing to advances in digital signal processing (DSP) chips and electronic miniaturization. They consist of "convolving engines" that process each sound source, as illustrated in Figure 3.23. The head position data from a 3D tracker, such as those discussed in Chapter 2, is sent to the host computer over an RS232 line. Each convolving engine on the Convolvotron board then calculates the new position of the corresponding simulated sound source relative to the user's head. These data are used to calculate the two new HRTFs for the left and right ears, based on a look-up table of impulse responses. The head tracker resolution is much better than the table position index step size used to obtain the corresponding HRTFs. Therefore HRTFs are interpolated with linear weighting functions at the sample rate. Next the filters are applied by the convolving engine to the sound input (after it was first digitized). The convolved sound from convolver I is then summed to that of convolver 2 and so on, until all outputs are added. The compound sound, corresponding to all 3D sound sources, is converted to an analog signal and then sent to the HMD headphones.

An example of an audio workstation for VR simulations is illustrated in Figure 3.24 [Lake Technology, 2002]. The Huron 20 consists of a separate enclosure housing DSP cards and I/O cards plugged on a common 24-bit Huron bus (256 kHz). A separate PC bus is used to control and synchronize the DSP and 1/0 cards. The DSPs are in charge of the convolving computations, while the 1/0 cards accept analog and digital audio signals from the user's microphone and other sources and output 3D sound to the user. Two users, each wearing a graphics display (HMD or active glasses) and head trackers, receive visual feedback from a graphics engine. The same head tracking information is sent by the graphics engine to the Huron 20 audio workstation over an Ethernet line. This information is then used in the convolving of the simulation sound sources, as previously discussed. Sound sources can be produced by the graphics workstation in response to user's actions, or can be the user's voice input. The processing software allows the simulation of wall reverberations and sound occlusions when the two users are in different simulated rooms. This way they can hear each other through the "wall". The Huron 20 workstation assumes the role of an audio

coprocessor, offloading the graphics engine. Such a distributed architecture approach is typical of multiuser environments, as discussed in more detail in Chapter 4.

3.2.3 Speaker-Based Three-Dimensional Sound

The high price of the Huron 20 (approximately \$40,000) makes it impractical for consumers who want to add 3D sound to their PC or home theater electronics. Such setups typically do not incorporate head trackers, and display sound on multiple loudspeakers rather than on headphones.

The simplest multispeaker audio system is the stereo format, which produces sound that appears to come from the plane defined by the two speakers. An improvement is the quad format, where two speakers are in front and two are behind the user. Another setting is the "5.1 surround" format, in which three speakers are in front of the user, two are lateral (left and right), and one is a subwoofer. Such multichannel audio systems produce a richer sound than stereo, but are more expensive and complex and occupy more space. Furthermore, the sound clearly comes from the loudspeakers, not the environment, and seems to stick around the perimeter of the room. Since HRTFs are not used, the illusion of sound coming from a location other than the speakers cannot be realized [Kraemer, 2001].

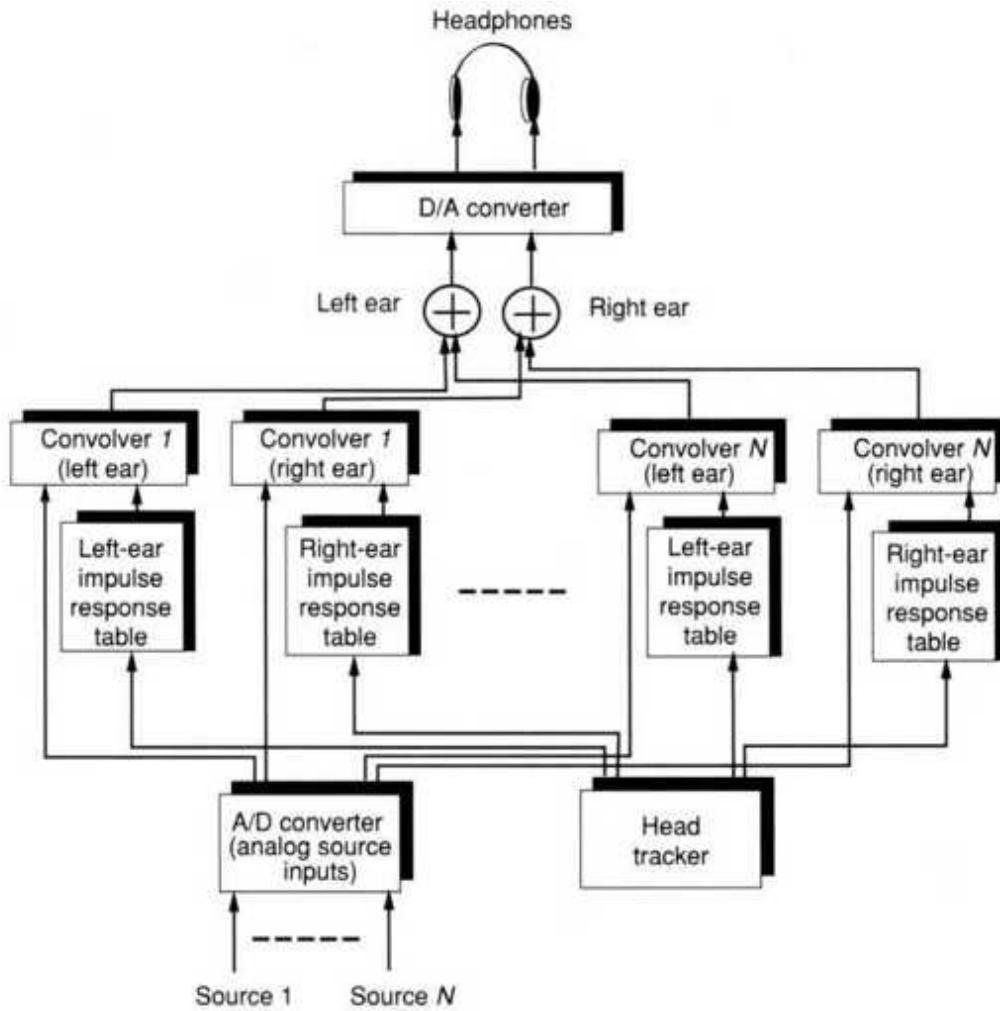


Fig. 3.23 The generic Convolvotron block diagram.

In recent years a new generation of inexpensive PC 3D sound cards has appeared. They use DSP chips to process stereo or 5.1-formatted sound and convolve it to output true 3D sound. As illustrated in Figure 3.25, the PC loudspeakers are placed left and right of the monitor, oriented parallel with it and facing the user. In this way HRTFs can be retrieved from the look-up table, knowing the approximate position of the user's head (facing the PC), within a "sweet spot" zone. It is thus possible to create the illusion of many more phantom speakers surrounding the user and create effective azimuth localization, as long as the user remains in the sweet spot zone.

Unlike headphones, the loudspeaker-based 3D sound systems cannot isolate the sound for each ear. Thus the sound reaching the left ear is a

mixture of the sound coming from the corresponding loudspeaker (S_{left}) and the cross-talk from the other loudspeaker (S_{right}) [Duda, 1997]. The situation is mirrored for the right ear. This cross-talk effect is expressed by

$$l'mft 1 _ HI,I HI,r S_{left} Y_{right} Hr,I Hr,r S_{right} \quad (3.3)$$

where HI,I is the HRTF between the left speaker and the left ear, HI,r is the HRTF between the right loudspeaker and the left ear, and so on. Here Y_{left} and Y_{right} are known (the output of the convolving process), and the loudspeaker output needs to be computed to assure cross-talk cancellation. Mathematically this means inverting the matrix in Equation (3.3) in order to determine the unknown S_{left} and S_{right} that provide proper 3D cues despite cross-talk.

Several companies have developed 3D sound cards that process six discrete channels of digital sound (such as Dolby Digital) and present them on two loudspeakers. Examples are the TruSurround cards produced by SRS and the SonicFury 3D produced by Videologic. Such cards provide simplified 3D sound capabilities that are very much appreciated by video game players, their intended target application. Games that support such cards give the players the ability to hear the direction from which an opponent approaches or shoots, providing a definite advantage compared to the PC sound cards of the past. Unfortunately, at this time, not all the computations required for 3D sound generation are done by the sound card hardware. The gap needs to be made up by the CPU, and this can adversely affect the game frame refresh rate. This drops by 10-30% (depending on the particular system) when the 3D sound processing is enabled.

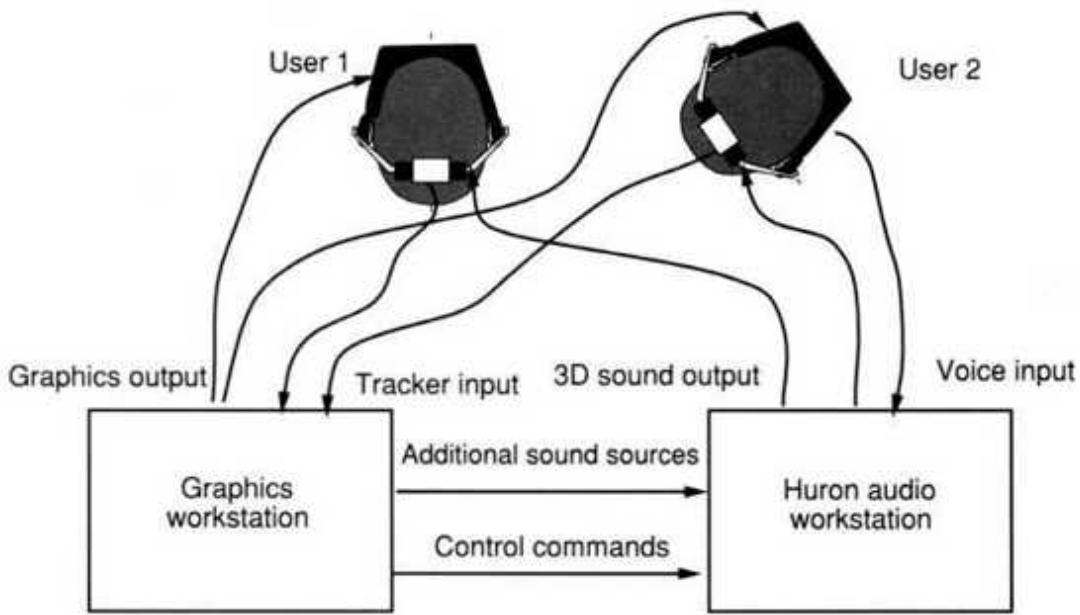


Fig. 3.24 The Huron 20 audio workstation. Adapted from Lake Technology [2002]. © Lake Technology Ltd. Reprinted by permission.

3.3 HAPTIC FEEDBACK

The last category of 1/0 devices discussed in this chapter are haptic interfaces. Named after the Greek term *hapthai* (meaning "touch"), they convey important sensorial information that helps users achieve tactile identification of virtual objects in the environment and move these objects to perform a task [Cutt, 1993; Burdea, 1996]. When added to the visual and 3D audio feedback previously discussed, haptic feedback greatly improves simulation realism. It becomes mandatory wherever the visual feedback is corrupted (occluded objects that need to be manipulated) or lacking entirely (dark environments). The discussion in this chapter is limited to feedback to the user's hand and wrist because the hand has the highest density of touch receptors in the body. Furthermore, systems that provide feedback to the whole body are either in the research stage at this time or are too specialized to be of use outside their intended settings.

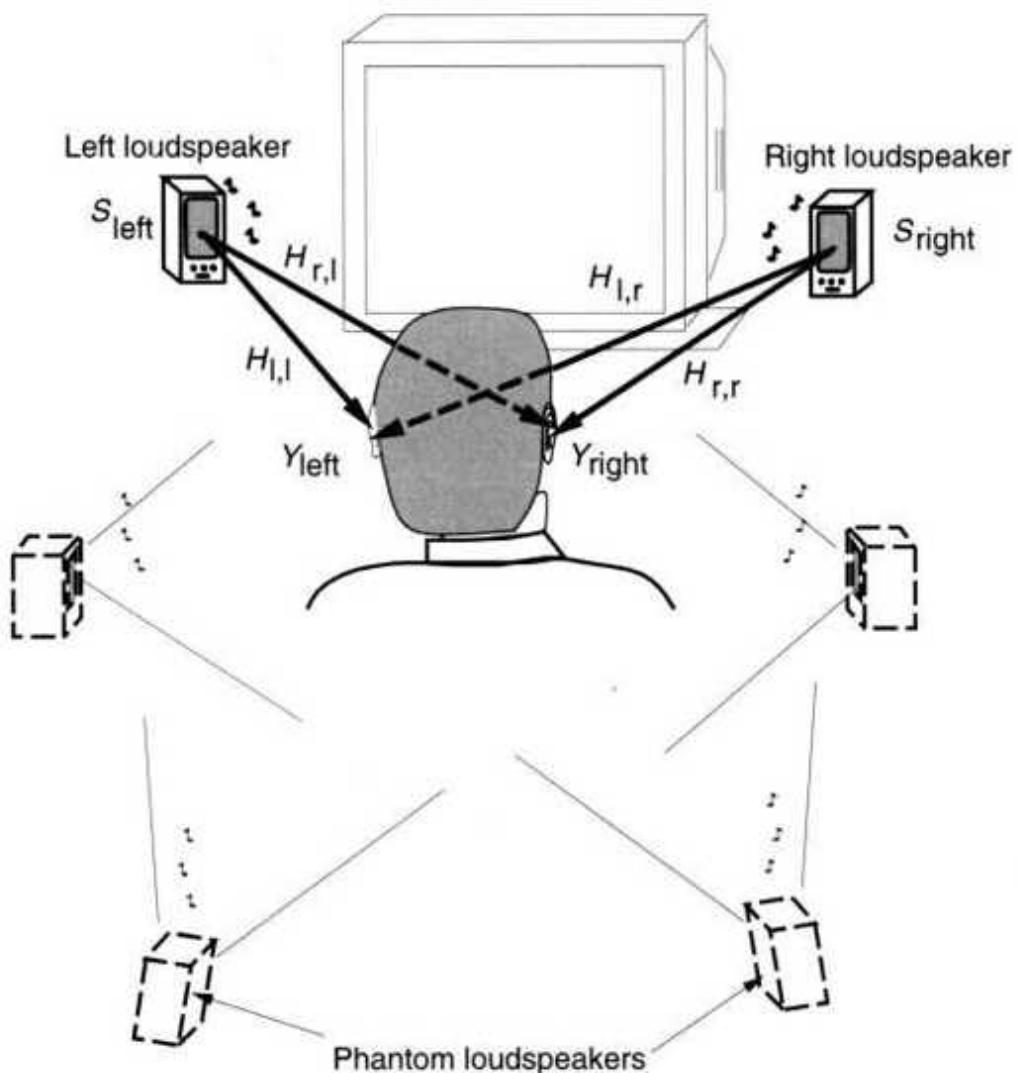


Fig. 3.25 Speaker-based 3D audio. Adapted from Kraemer [2001]. © 2001 IEEE. Reprinted by permission.

Haptic feedback groups the two modalities of touch and force feedback. Consider a hand pushing very lightly against a desk. The sensors that respond first are the fingertip touch sensors. If the hand pushes harder, then the muscles in the hand and forearm start to contract. The level of force applied is now felt by sensors on muscle ligaments and bones (kinesthesia) as opposed to fingertips. The technical literature intermixes the terms "touch" and "force" feedback, so clarification is needed.

Definition Touch feedback conveys real-time information on contact surface geometry, virtual object surface roughness, slippage, and temperature. It does not actively resist the user's contact motion and cannot stop the user from moving through virtual surfaces.

Definition Force feedback provides real-time information on virtual object surface compliance, object weight, and inertia. It actively resists the user's contact motion and can stop it (for large feedback forces).

Designing good haptic feedback interfaces is a daunting task in view of the specific requirements they need to meet. The first is user safety and comfort. While the user interacts with virtual objects, the forces he or she feels are real. These contact forces need to be large (in the simulation of rigid objects), but not large enough to harm the user. In this context a good design is also fail-safe, so that users are not subject to accidents in case of computer failure. Yet another issue is portability and user comfort. The difficulty with force-feedback actuators is the need to provide sufficient force while still keeping the feedback hardware light and unintrusive. If haptic interfaces are too heavy and bulky, then the user will get tired easily and will prefer a less cumbersome open-loop control. Heavy feedback structures can be gravity counterbalanced, but this further increases complexity and cost. Portability also relates to ease of use and installation at the simulation site. Haptic feedback interfaces should be self-contained, without requiring special supporting construction, piping, or wiring.

All these conflicting requirements call for a compromise, which can only be found if the human haptic characteristics are taken into account. In the context of this book we look at the sensorial characteristics of the hand from the engineering perspective. The hardware designer is interested in the manual force exertion capability, which helps in selecting the appropriate feedback actuators. The hardware designer is also interested in manual degrees of freedom and mobility, which determine the number of feedback actuators and their range of motion. Similarly, the control engineer is interested in the sensorial bandwidth of the hand, which determines how fast the control loop needs to be closed. Finally, the human factors specialist is interested in the safety and comfort issues related to feedback to the hand.

3.3.1 The Human Haptic System

Input to the human haptic system is provided by the sensing loop, while output to the environment (in this case the haptic interface) is mediated by the sensory-motor control loop. The input data are gathered by a multitude of tactile, proprioceptive, and thermal sensors, while the output is forces and torques resulting from muscular exertion. The system is not balanced, in the sense that humans perceive haptically much faster than they can respond.

3.3.1.1 Haptic Sensing. The skin houses four types of tactile sensors (or receptors), namely Meissner corpuscles (the majority), Merkel disks, Pacinian corpuscles, and Ruffini corpuscles. When excited they produce small electrical discharges, which are eventually sensed by the brain. The slow-adapting (SA) sensors (Merkel and Ruffini) maintain a constant rate of discharge for a long time in response to a constant force being applied on the skin. On the contrary, the fast-adapting (FA) ones (Meissner and Pacinian) drop their rate of discharge so fast that the constant contact force becomes undetected. Such different behavior is related to the particular type of contact the tactile sensors detect. If the contact has a strong time variation (vibrations on the skin, or accelerations), then FA receptors are needed. Conversely, if the contact is quasi steady state (edges, skin stretch, static force), then SA receptors are used. As a consequence, Merkel and Ruffini receptors are most sensitive to low-frequency stimuli (0-10 Hz), while Meissner and Pacinian sensors respond to stimuli of higher frequencies (50-300 Hz). The receptive field of a tactile receptor determines its spatial resolution. If the sensor has a large receptive field, its spatial resolution is low. Conversely, the Meissner and Merkel receptors, which have small receptive fields, have high spatial resolution. Table 3.3 summarizes the characteristics of the four tactile sensors (or receptors) present in the skin [Burdea, 1996].

The spatial resolution of the skin varies depending on the density of receptors it has. The fingertips, where such density is the highest, can discriminate two contacts that are at least 2.5 mm apart. However, the palm cannot discriminate two points that are less than 11 mm apart, and the user feels as if only one contact point exists [Shimoga, 1992]. Spatial resolution

is complemented by temporal resolution when two contacts occur on the skin close in time. The skin mechanoreceptors have a successiveness threshold of only 5 msec, which is much smaller than the corresponding value for the eye (25 msec).

Temperature sensing is realized by specialized thermoreceptors and nociceptors. The thermoreceptors are located either in the epidermis (sensitive to cold) or dermis (sensitive to warmth). Their spatial resolution is less than that of mechanoreceptors. Nociceptors are sensitive to contact temperatures that are outside the interval of -15°C to 45°C, and trigger a pain response in the cortex.

TABLE 3.3. Comparison of Various Skin Mechanoreceptors^a

Receptor Type	Rate of Adaptation ^b	Stimulus Frequency (Hz)	Receptive Field	Detection Function
Merkel disks	SA-I	0–10	Small, well defined	Edges, intensity
Ruffini corpuscles	SA-II	0–10	Large, indistinct	Static force, skin stretch
Meissner corpuscles	FA-I	20–50	Small, well defined	Velocity, edges
Pacinian corpuscles	FA-II	100–300	Large, indistinct	Acceleration, vibration

^aBased on Seow [1988], Cholewiak and Collins [1991], and Kalawsky [1993].

^bSA, slow-adapting; FA, fast-adapting.

Another type of sensing of interest to the haptic interface designer is proprioception, or the perception of one's own body position and motion. This is due to nerve endings, as well as Pacinian and Ruffini corpuscles, located at the skeletal articulations. The amplitude of the receptor potential discharge is a function of joint position, while its frequency corresponds to joint velocity. The accuracy with which the body can position its limbs depends on the proprioceptive resolution, or the smallest change in joint position that can be detected. This "just noticeable difference" is smallest for the hip (0.2°) and shoulder (0.8°) joints and largest for the finger (2.5°) and toe (6.1°) [Kalawsky, 1993; Tan et al., 1994]. This reflects the body's need to minimize Cartesian errors at the limb extremities, errors which are most dependent on the sensitivity of shoulder and hip proprioceptive sensors.

Proprioception is complemented by kinesthesia, which senses the muscle contraction or stretching. This is realized by Golgi organs located between muscles and their corresponding tendons, and by muscle spindles located between individual muscle fibers. The Golgi organs measure muscle tension, while muscle spindles determine the rate of increase in muscle length (stretch). Force sensation is also affected by muscle fatigue. This results in a perceived increase in force magnitude, even though the actual force produced by the muscles stays constant [Johnes and Hunter, 1992].

3.3.1.2 Sensory-Motor Control. The tactile, proprioceptive, and kinesthetic sensing is used by the body's sensory-motor control system to affect the forces applied on the haptic interface. Key aspects of the human sensory-motor control are maximum force exertion capability, sustained force exertion, force tracking resolution, and force control bandwidth.

Finger contact forces depend on whether the action is volitional or reflex, on the way objects are grasped, as well as on user's gender, age, and skill. The grasping geometry can be classified into precision grasp and power grasp, as illustrated in Figure 3.26 [Cutkosky and Howe, 1990]. Precision grasps are used in dextrous object manipulation where contact is made at the fingertips. Power grasps allow higher stability and force exertion because the object is held between the palm and the closed fingers, but lack dexterity (fingers are locked around the grasped object). Clinical studies [An et al., 1986] measured the human manual force exertion capability. It was found that males can exert a maximum force of 400 N during power grasping, while female hand strength ranges from 60% to 80% of that of males. Table 3.4 illustrates the contact force dependence on grasp configuration and gender. Note that such high forces are not the ones haptic interfaces need to produce. A large grasping force cannot be maintained by a user for a long time without experiencing discomfort and (eventually) pain. In pinch grasping, for example, discomfort and pain were reported when the exerted force was 25% of maximum for as little as 10 minutes [Wiker et al., 1989].

Other aspects of interest with regard to force exertion are force tracking resolution and force control bandwidth. Force tracking means controlling the force to match either a constant or a time-varying target value. Humans are

better at tracking constant target forces than they are when forces are time-varying. For example, finger contact forces stay within 11-15% of constant-force targets (the larger errors corresponding to larger force targets) [Srinivasan and Chen, 1993]. The target force values were kept well outside the fatigue zone in order to avoid its degrading effects on the tracking ability of the subjects. In a subsequent study the tracking of constant-torque targets was realized with 10-14% error [Jandura and Srinivasan, 1994]. Control bandwidth refers to the rapidity with which humans respond to haptic stimuli. The output loop for finger force exertion has a bandwidth of 5-10 Hz, which is much less than the sensing bandwidth of the proprioceptors and of mechanoreceptors (20-400 Hz or more) [Shimoga, 1992].

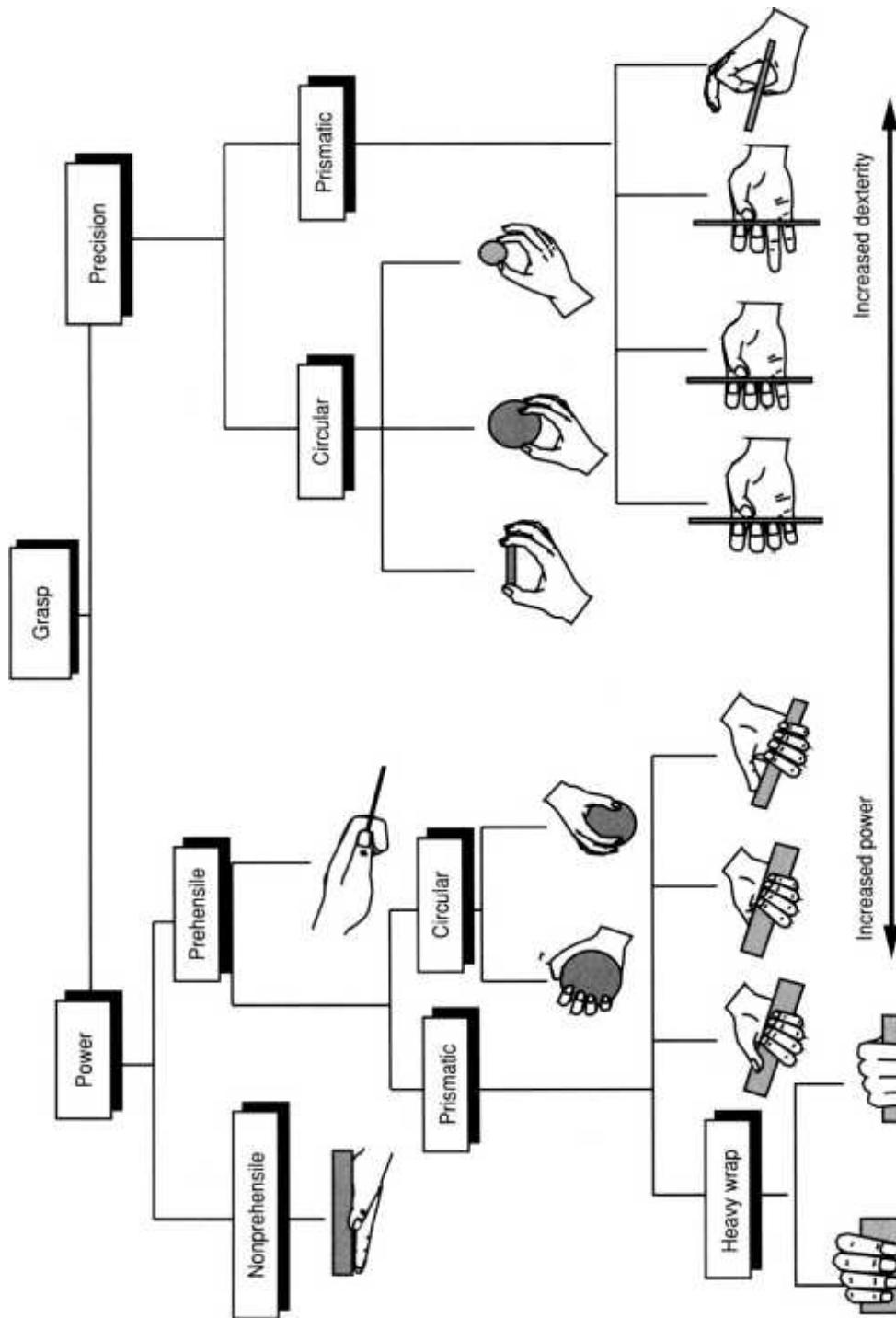


Fig. 3.26 Human grasp geometries Adapted from Cutkosky and Howe [1990]. © Springer-Verlag GmbH. Reprinted by permission.

3.3.2 Tactile Feedback Interfaces

The first category of haptic interfaces discussed here is tactile feedback devices for the hand. There are many ways in which skin tactile receptors can be stimulated, ranging from air bellows and jets, to electrical actuators

producing vibration, to micropin arrays, direct electrical stimulation, and functional neuromuscular stimulations. Electrotactile feedback provides electric pulses to the skin (with varying width and frequency). Neuromuscular stimulation provides the signal directly to the user's primary cortex. Both techniques are considered by the authors to be risky and are not discussed here. Instead we discuss commercially available interfaces for the hand that provide vibrotactile feedback and temperature feedback.

3.3.2.1 The Tactile Mouse. The computer mouse is a standard interface, serving as an open-loop navigation, pointing, and selecting device. By open-loop we mean that the information flow is unidirectional, being sent from the mouse to the computer (either as relative X, Y position increments or as button states). The interface loop is therefore closed through the vision and auditory channels (highlights of selected menu options and sounds confirming an action).

The standard way of using the mouse requires that the user look at the screen all the time, lest control be lost. Tactile feedback can remedy this by adding another cue in response to the user's actions, one that can be felt even if the user looks away. Further advantages stem from the ability to characterize menu options, icons, or virtual objects haptically, such that they will feel differently.

In view of the advantages that tactile feedback can bring to mouse-mediated interactions, it is not surprising that a number of tactile mice have been introduced on the market. One example is the iFeel Mouse illustrated in Figure 3.27a [Logitech Co., 2002]. Its outside appearance, weight (132 grams), and price (\$40) are similar to those of standard computer mice. The difference is the addition of an electrical actuator that can vibrate the mouse outer shell. As illustrated in Figure 3.27b [Rosenberg and Martin, 2001], the actuator shaft translates up and down in response to a magnetic field produced by its stationary element. The shaft has a mass attached to it, creating inertial forces of more than 1 N, which are felt by the user's palm as vibrations. The actuator is oriented perpendicular to the mouse base, such that the vibrations occur in the Z direction. This design minimizes the negative effects vibrations could have on the X-Y mouse translation and

resulting pointing inaccuracy. The mouse pad needs to be thicker than usual, and elastic in order to absorb the reaction forces from the supporting desk. Furthermore, the iFeel mouse uses optical position measurement rather than the mechanical ball used by many other mouse designs. This solution is necessary because vibrations produced by the actuator can interfere with the ball-roller assembly used to measure X-Y displacements. The optical sensor data are used by a microprocessor located in the mouse to determine the amount of mouse translation. This data are sent to the host computer over a universal serial bus (USB) line that can also provide power. The host software detects contact between the screen arrow controlled by the mouse and haptically enabled window borders, icons, or surfaces. As a result, haptic commands indicating the onset and type of tactile feedback are sent to the mouse processor. The processor then converts the high-level commands into vibration amplitude and frequency and drives the actuator through an actuator interface. The sensation that the user has is that of a haptic "bump" if only an impulse command is sent by the PC, or of haptic textures if complex amplitude-modulated commands are sent.

TABLE 3.4. Hand Force Distribution (in N) by Grasp Type and Gender^a

Gender of Subjects	Power Grasp	Tip Pinch ^b	Pulp Pinch ^c	Key Pinch ^d
Male	400	65	61	109
Female	228	45	43	76

^aBased on An et al. [1986].

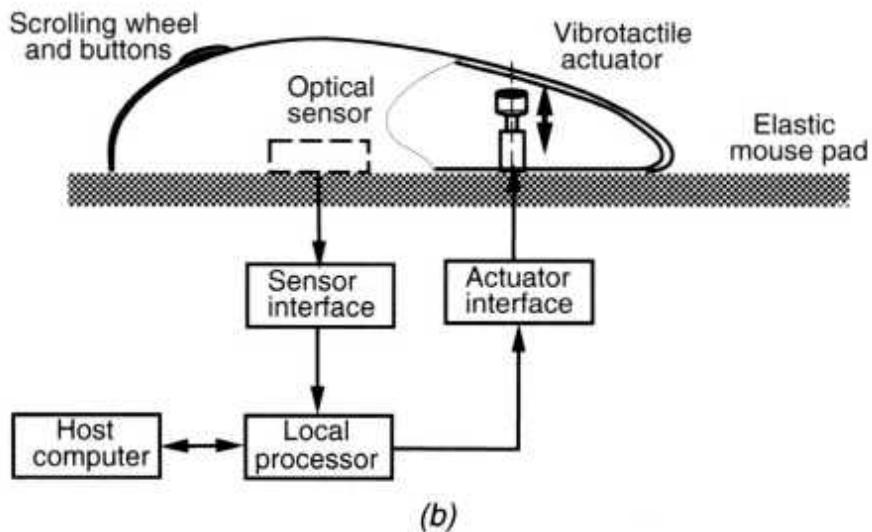
^bUses fingertips of thumb and index

^cUses the fingertips of thumb, index, and middle

^dThumb over index middle phalanx



(a)

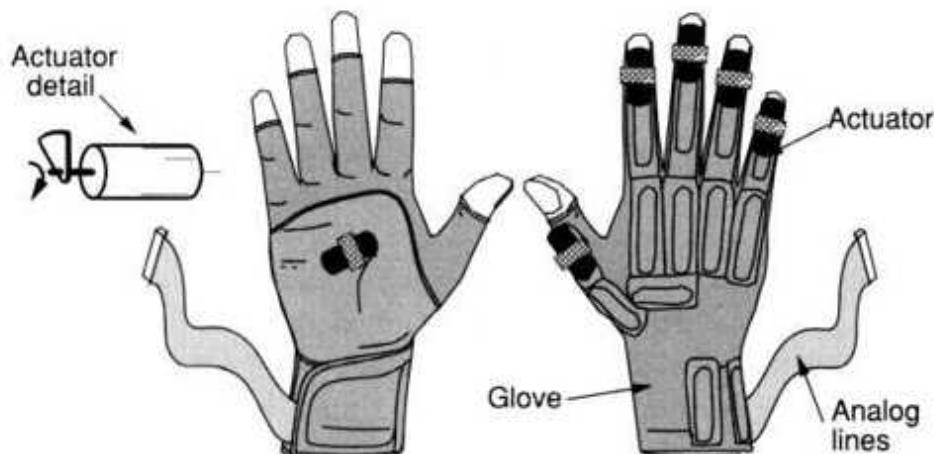


(b)

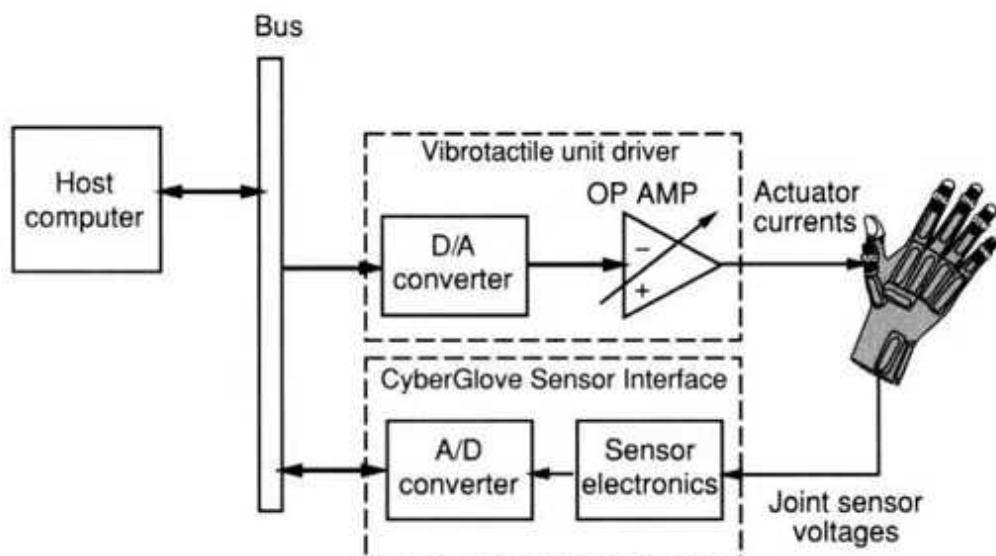
Fig. 3.27 The iFeelTM tactile feedback mouse. (a) Outside appearance. From Logitech Co. [2002]. © 2002 Logitech. All rights reserved. Logitech, the Logitech logo, and other Logitech marks are owned by Logitech and may be registered. Reprinted by permission. (b) Tactile feedback system. Adapted from Rosenberg and Martin [2001]. © 2001 Immersion Co. Reprinted by permission.

3.3.2.2 The CyberTouch Glove. This is another haptic interface that provides vibrotactile feedback to the user [Immersion Corporation, 2002a]. As illustrated in Figure 3.28, the device is a CyberGlove retrofitted with six

vibrotactile actuators (one on the back of each finger and one in the palm) [Tremblay and Yim, 2000]. Each actuator consists of a plastic capsule housing a DC electrical motor. The motor shaft has an off-centered mass, which produces vibrations when rotated. Thus, by changing the speed of rotation, it is possible to change the vibration frequency between 0 and 125 Hz. Each actuator applies a small force of 1.2 N, which is felt by the skin mechanoreceptors and kinesthetic receptors (through the finger bones). During VR simulations the CyberGlove reads the user's hand configuration and transmits the data to the host computer over an RS232 line. These data, together with those of a 3D tracker attached to the wrist, are used to drive a virtual hand, which is then displayed by the graphics display. Whenever the fingers or palm of the virtual hand interact with virtual objects, the host computer sends commands necessary to activate the vibrotactile actuators. These signals are received by the actuator driver unit, which applies the corresponding currents using D/A converters and operational amplifiers. In this way the feedback loop is closed haptically and the user feels the vibrations on the skin. The CyberTouch glove is most suitable for dexterous manipulation tasks, where contact is at the fingertips, since it has the ability to provide feedback to individual fingers. This is a definite advantage over the iFeel mouse previously described. Furthermore, by exciting more than one actuator at a time it is possible to generate complex tactile feedback patterns with a relatively light structure (5 oz). Finally, the CyberTouch glove significantly increases the user's freedom of motion compared with the iFeel mouse, which requires that the user's hand be kept on the desk. This increased functionality comes at a price: The CyberTouch glove costs \$15,000 (including the CyberGlove support).

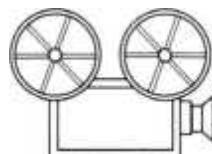


(a)



(b)

Fig. 3.28 The CyberTouch glove. Adapted from Tremblay and Yim [2000]. © 2000 Immersion Co. Reprinted by permission.



VC 3.3

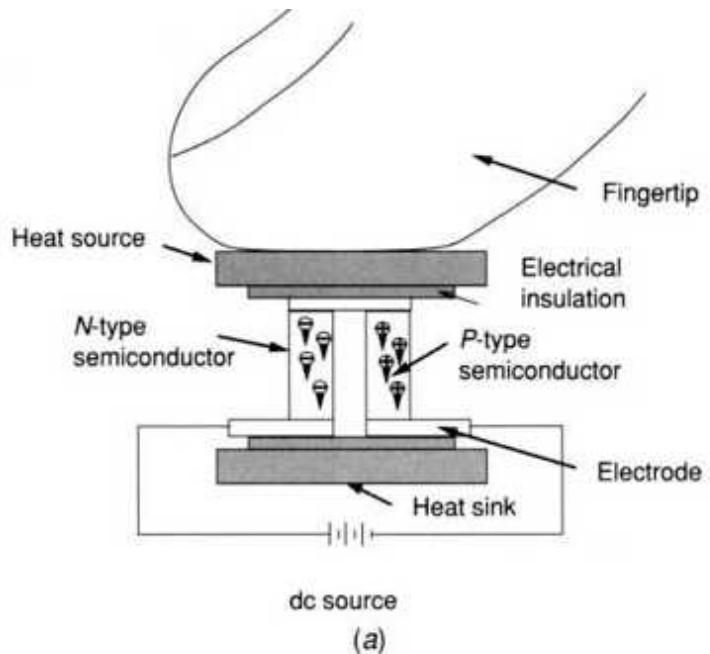
3.3.2.3 The Temperature Feedback Glove. This glove allows users to detect thermal characteristics that can help identify an object material. Such

variables are surface temperature, thermal conductivity, and diffusivity. Materials that have high conductivity (such as steel) will feel cold when grasped, while those with low conductivity (wood) will feel warm. This is due to the heat flow from the finger to the object (the finger is usually warmer) [Monkman and Taylor, 1993]. Adding temperature feedback to manipulation tasks in VR increases the simulation realism. Temperature feedback is also useful when navigating in very cold or very warm virtual worlds. Therefore we end our discussion of tactile feedback with a description of technology that provides computer-controlled temperature feedback to the fingertip.

A thermal actuator used to recreate the thermal signature of virtual objects needs to be light and compact, so that the user's freedom of motion is not diminished. It also needs to be clean and fast, so that rapid temperature changes can be produced. Such actuators are thermoelectric heat pumps that function on the Peltier principle. This states that a DC current applied to dissimilar materials placed in contact creates a temperature differential. Modern Peltier pumps consist of solid-state N- and P-type semiconductors sandwiched between ceramic electrical insulators, as illustrated in Figure 3.29a [Phywe Systeme, 2003]. The ceramic plates serve as thermal conductors and mechanical supports. One is called a heat source and the other a heat sink. When current from a DC source is applied to the heat pump, the P and N charges move to the heat sink plate, where they transfer heat. This results in a drop in temperature of the heat source plate and a corresponding rise in temperature of the heat sink plate. The larger the current, the larger is the temperature difference between the two plates, up to approximately 65°C. Our discussion on human haptic sensing mentioned that the user's thermal comfort zone is 13-46°C. Thus one such thermoelectric heat pump is sufficient for VR simulation purposes.

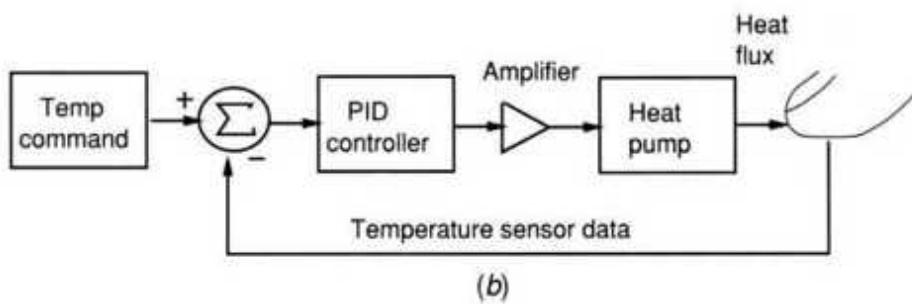
The Displaced Temperature Sensing System (DTSS) developed by C&M Research consists of eight thermodes and a control interface unit. Each thermode [Zerkus et al., 1993] consists of a Peltier heat pump and a thermocouple temperature sensor attached to the plate that is in contact with the skin. The user's fingertip temperature is measured in real time by the thermocouple and is then sent to the control interface, as illustrated in Figure

3.29b. Here the fingertip temperature is compared with the target temperature set by the host computer and sent to the DTSS control interface through an RS232 serial line. The temperature differential between target and actual fingertip temperatures is the input to a proportional-integrative-derivative (PID) controller. Its output is then sent to current amplifiers, which drive the thermoelectric heat pump, and the control loop is closed. Each fingertip and additional locations on the palm have similar thermode control blocks. The host computer can change the gains of the PID controller to implement different control laws, as required by the simulation. The controller safety software monitors the fingertip temperature in order to maintain them within the user's comfort zone and prevent burning in case of heat pump failure. The weight of the eight thermodes is 340 grams, assuming air cooling of the heat sink. Water cooling, which assures faster heat transfer and better system response, adds to the system complexity and weight.



dc source

(a)



(b)

Fig. 3.29 Temperature feedback glove. (a) Thermoelectric heat pump element. Adapted from Phywe Systeme [2003]. Reprinted by permission. (b) Thermode control diagram. Adapted from Zerkus et al. [1993]. Reprinted by permission.

3.3.3 Force Feedback Interfaces

Force feedback interfaces are devices that differ in several aspects from the tactile feedback interfaces previously discussed. First, the requirement to provide substantial forces to stop user's motion implies larger actuators, heavier structures (to assure mechanical stiffness), larger complexity, and greater cost. Furthermore, force feedback interfaces need to be grounded (rigidly attached) on some supportive structures to prevent slippage and potential accidents. Force feedback interfaces such as joysticks and haptic arms are not portable, since they are grounded on the desk or on the floor. More portable interfaces, such as force feedback gloves, are grounded on the user's forearm. This allows more freedom of motion for the user and more natural interaction with the simulation, at the expense of potential arm fatigue due to the interface weight. An important characteristic of force feedback interfaces is mechanical bandwidth.

Definition The mechanical bandwidth of a force feedback interface represents the frequency of force and torque refreshes (in hertz) as felt by the user (through finger attachments, handles, gimbals, etc.).

Mechanical bandwidth should not be confused with control bandwidth, which represents the frequency of command updates received by the interface actuators. The control bandwidth of force feedback interfaces will always be larger than their mechanical bandwidth, owing to the inertia of the feedback structure. Therefore, the larger the force feedback interface, the larger are its output forces and the smaller is its mechanical bandwidth.

3.3.3.1 Force Feedback Joysticks. These are some of the simplest, least expensive and most widespread force feedback interfaces today. These have a small number of degrees of freedom and a compact shape and produce moderate forces with high mechanical bandwidth. One illustrative example is the WingMan Force 3D joystick shown in Figure 3.30a [Logitech Co.,

2001], which costs \$60. The joystick has three degrees of freedom, two of which have force feedback, as well as analog buttons and switches used in gaming. The force feedback structure is housed in the joystick base and consists of two DC electrical actuators connected to the central handle rod through a parallel kinematic mechanism [Rosenberg, 1998]. Each actuator has a capstan drive and pulley, which moves a gimbal mechanism composed of two rotating linkages. The two actuator-gimbal assemblies are oriented perpendicular to each other, allowing the tilting of the central rod front-back and sideways (leftright). The tilting of the joystick is measured by two digital encoders coaxial with the motor shafts. These angles are processed by the joystick onboard electronics (sensor interface) and sent to the host PC over a USB line. Also sent to the computer is the status of the joystick buttons after their analog signal is digitized by an A/D converter. The computer then changes the simulation based on the user's actions and provides feedback in case of haptic events (shooting, explosions, inertial accelerations). These commands are transformed into analog signals by the joystick D/A converter, amplified, and sent as driving currents to the DC actuators. In this way the control loop is closed and the user feels the vibrations and jolts or springlike forces produced by the joystick. These forces have a maximum of 3.3 N.

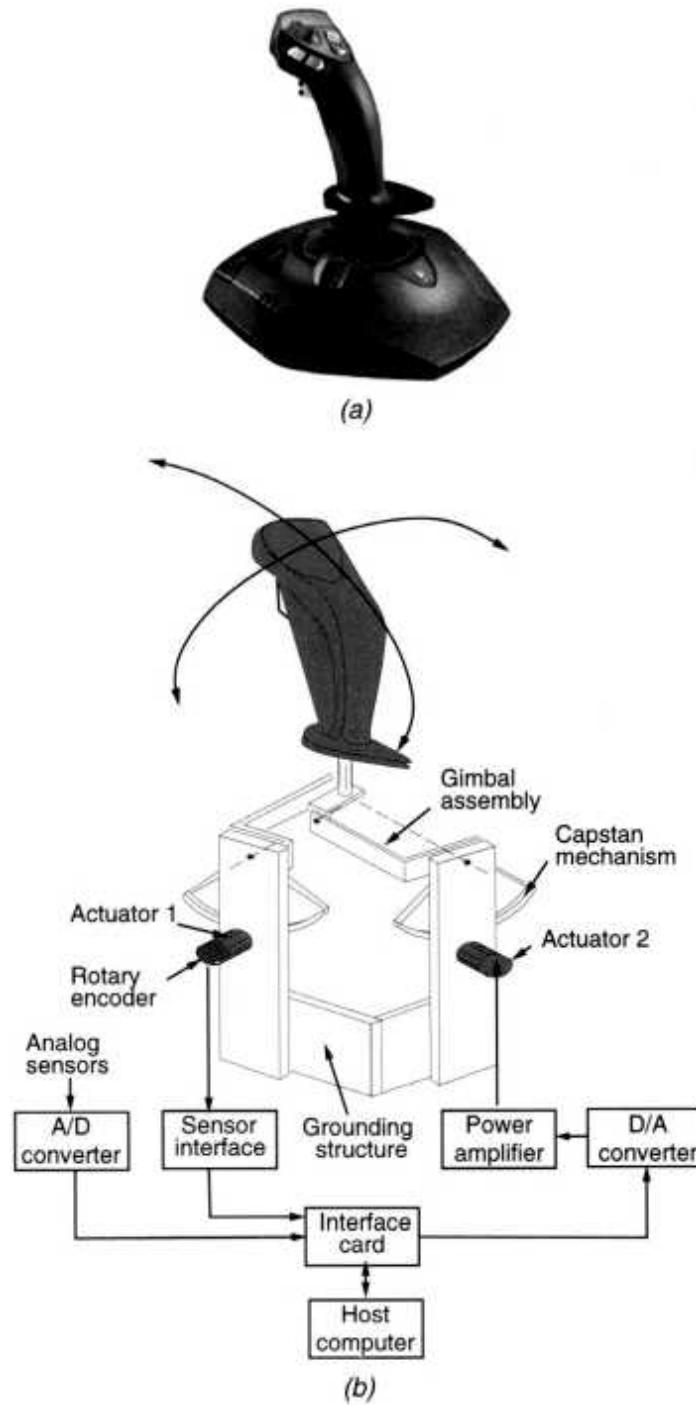
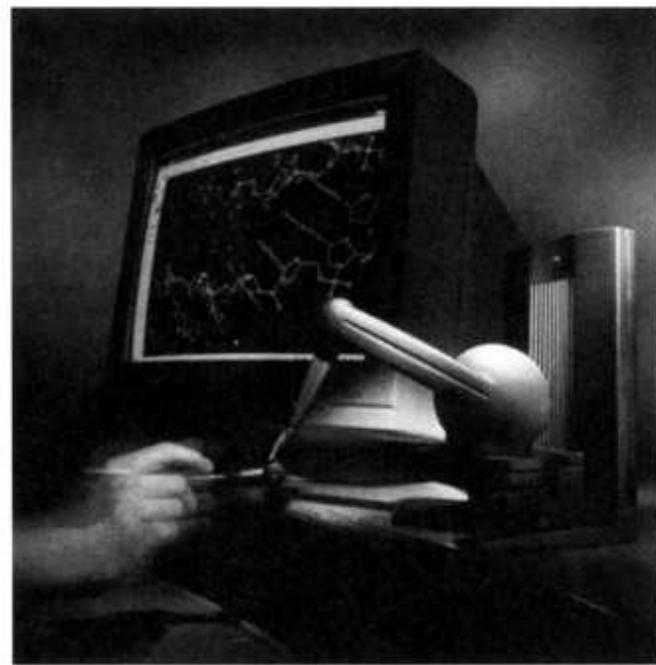
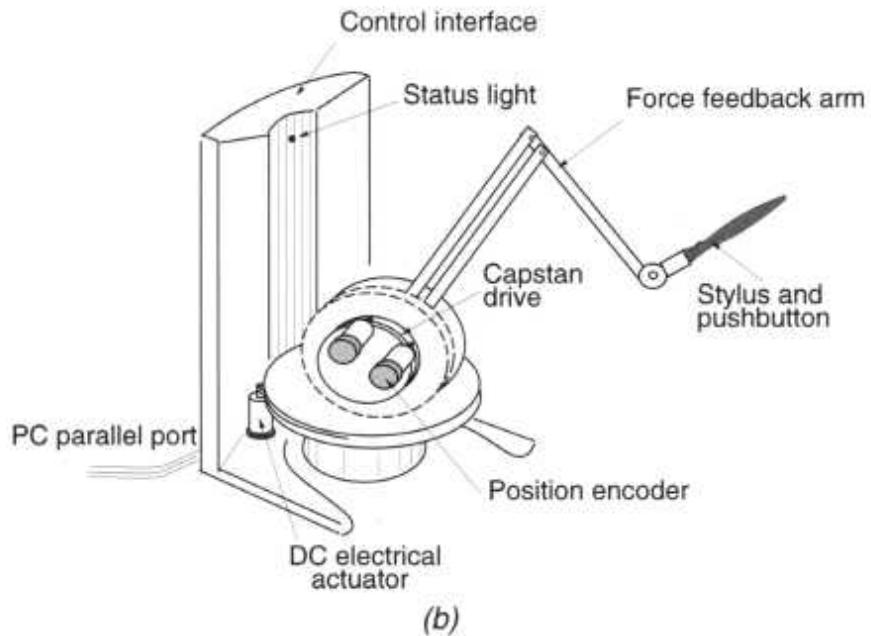


Fig. 3.30 The WingMan® ForceTM 3D joystick. (a) Outside appearance. From Logitech Co. [2001]. © Logitech. All rights reserved. Logitech, the Logitech logo, and other Logitech marks are owned by Logitech and maybe registered. Reprinted by permission. (b) The force feedback system. Adapted from Rosenberg [1998]. © 1998 Immersion Co. Reprinted by permission.

3.3.3.2 The PHANTOM Arm. This is used in simulations that need three to six haptic degrees of freedom as well as increased dexterity compared to joysticks. Earlier haptics research used master controllers intended for telemanipulation applications coupled with simulation workstations [Ouh-Young et al., 1988]. These were large mechanical structures with imbedded position sensors and electrical feedback actuators, and were ill suited for use outside an industrial setting. They were subsequently replaced by the much more compact Personal Haptic Interface Mechanism (PHANTOM) [Massie and Salisbury, 1994]. As illustrated in Figure 3.31a, the interface main component is a serial feedback arm that ends with a stylus. Of the six degrees of freedom of the arm, three are active, providing translational force feedback. The stylus orientation is passive, so no torques can be applied to the user's hand. The arm design cleverly places two of the three feedback actuators so that their weight counterbalances the arm weight. This obviates the need for active gravity compensation through biased motor torques. The resulting arm workspace is approximately that of the user's wrist, with the forearm resting on a support.



(a)



(b)

Fig. 3.31 The PHANToM Desktop force feedback arm. (a) Outside appearance. From SensAble Technologies Inc. [2002a]. Reprinted by permission. (b) Force feedback system.

The PHANToM Desktop uses three DC brushed motors with optical encoders placed at the actuator shafts and rotary potentiometers to measure

the handle orientation. Transmissions are done with cables and pulleys, and since the three actuators are nearly decoupled, the interface is perceived as having the same inertia (0.075 kg) and backdrivability (0.06 N) in all directions. By having small inertia and small friction (backdrivability) the PHANToM is well suited to simulate motion in free space (with no feedback forces present). The peak output force of the PHANToM Desktop is 6.4 N, while continuous force without overheating its actuators is only 1.7 N. Another important characteristic is the arm stiffness (3×103 N/m), which needs to be high to allow realistic simulation of contact with walls and other hard objects.

The PHANTOM actuators are controlled by an electronics assembly, which receives commands from the PC host over a parallel line. The control electronics is integrated with, and adjacent to, the feedback arm, resulting in a compact and selfcontained system. The local electronics consists of a D/A/D card, power amplifiers for the feedback actuators, conditioning electronics for the position sensors, and a status LED indicator. The haptic control loop runs at 1000 Hz, while the arm's mechanical bandwidth is significantly smaller. When tried by the authors the system seemed responsive and haptically convincing even with the low feedback force levels the Desktop model can produce.

The PHANToM Desktop's smaller output forces and its inability to feed back torques limit the type of applications it can be used for. Thus the manufacturer (SensAble Technologies, Inc., Woburn, MA) introduced Model 1.5/6.0, which is shown in Figure 3.32 [SensAble Technologies Inc., 2002b]. Its dimensions as well as maximum output force (8.5 N) are larger than those of the Desktop model. More importantly, the Model 1.5/6.0 outputs torques, which have maximum values of 170515 mN-m. The lower value corresponds to rotations about the handle axis, with the torque produced by an actuator housed inside the handle. The other two actuators are inside the segment connecting the handle to the two-linkage arm segment attached to the base. The continuous output forces/torques are much smaller, again owing to actuator overheating. More importantly, the torque mechanical bandwidth felt at the handle is (only) 15 Hz, which is much smaller than the 1000-Hz haptic control bandwidth previously mentioned.

The price of the PHANToM is high (\$16,000 for the Desktop and \$56,000 for the 1.5/6.0 model, including the haptic software running on the host computer). Nevertheless, with more than 1000 such interfaces in use, the PHANTOM is today's de facto standard professional haptic interface.

3.3.3.3 The Haptic Master Arm. The PHANToM Desktop force feedback system just described is extremely intuitive to use (as we use pens every day!). Additionally it is desktop-based, and has six degrees of freedom and a larger work envelope than simple joysticks. Certain tasks, such as VR-assisted rehabilitation, require a haptic interface that is capable of higher levels of sustained forces and larger work envelope. One such interface is the HapticMaster, produced in The Netherlands by FCS Control Systems. As shown in Figure 3.33 [FCS, 2001], the HapticMaster is a three-degree-of-freedom (3-DOF) cylindrical robot that can rotate about its base, move up and down, and extend its arm radially within a $0.64 \times 0.4 \times 0.36$ m³ work envelope. Its maximum output force (250 N) and stiffness (5×10^4 N/m) are much larger than the corresponding PHANToM capabilities. This is due in part to the much larger actuators used as well as its control scheme. Unlike the PHANToM, which uses a position-in, force-out servo loop (also called impedance control), the HapticMaster uses a force-in, position-out arrangement (also called admittance control). Users hold a simple handle mounted on a force sensor at the end of the arm. They can push around the HapticMaster, which provides user-programmable resistance. The control loop measures user's forces/torques at a very high rate of 2500 Hz and sets the end-effector position accordingly. This results in a mechanical bandwidth felt by the user of about 10 Hz [van der Linde, 2002]. This arrangement allows better simulation of hard, immovable objects (such as virtual walls). The disadvantages are larger apparent inertia (due to the massive structure of the arm) and higher system cost (\$34,000), since the force sensor and the position-feedback actuators incorporated in the HapticMaster are more expensive.



Fig. 3.32 The PHANToM 1.5/6.0, which produces forces and torques at the handle. From SensAble Technologies Inc. [2002b]. Reprinted by permission.

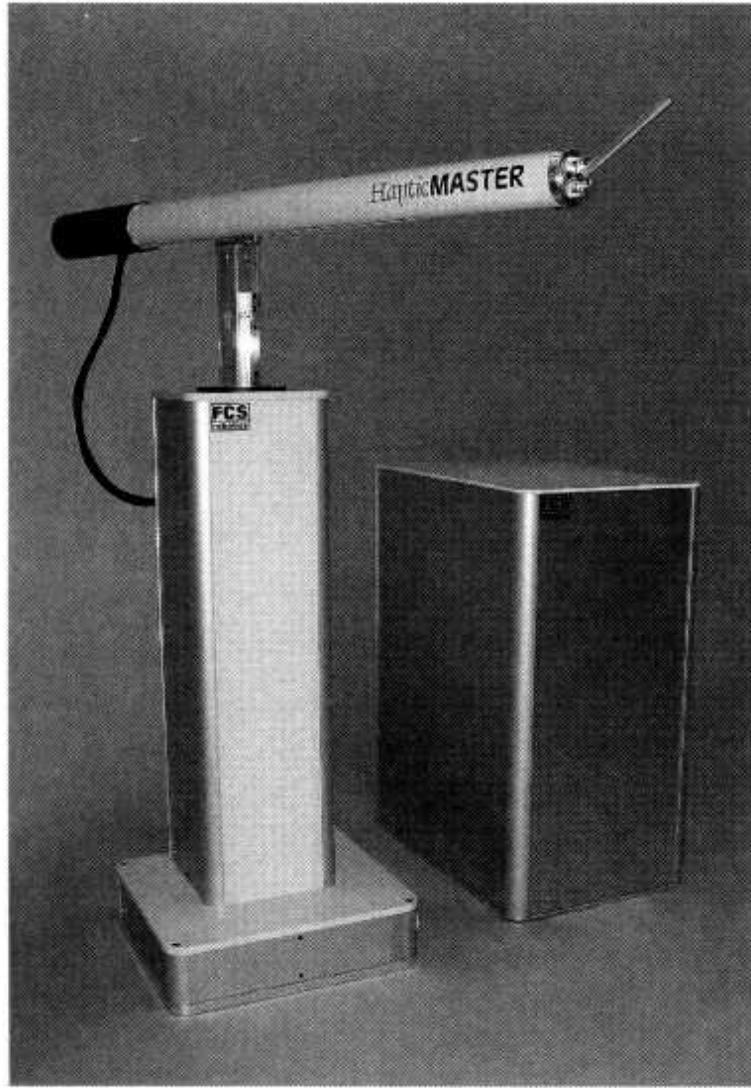
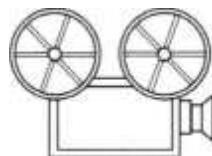


Fig. 3.33 The HapticMaster from FCS Control Systems [FCS, 2001]. Reprinted by permission.

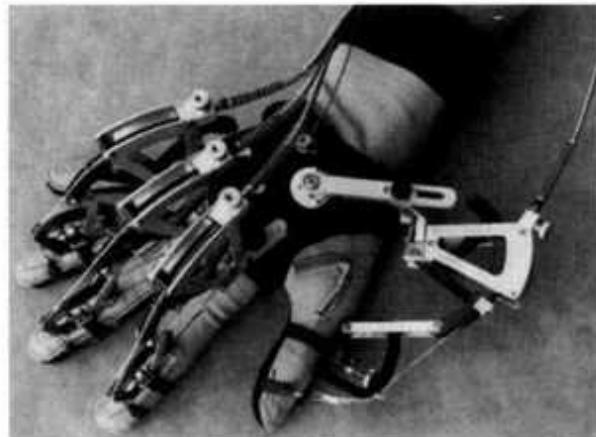
3.3.3.4 The CyberGrasp Glove. For tasks requiring more dexterity it may be necessary to control simulated forces on independent fingers rather than at the wrist, as joysticks and master arms do. In this case one uses force feedback gloves, such as the CyberGrasp shown in Figure 3.34a. The CyberGrasp system is a retrofit of the 22-sensor version of the CyberGlove, which it uses to measure user's hand gestures. As illustrated in Figure 3.34b [Virtual Technologies, 1999], the CyberGlove interface box transmits the resulting finger position data to the CyberGrasp force control unit (FCU) over an RS232 line. The same FCU receives wrist position data from a 3D

magnetic tracker worn by the user (such as the Ascension or Polhemus ones discussed in Chapter 2). The resulting hand 3D positions are sent to the host computer running the simulation over an Ethernet line (local area network, LAN). The host computer then performs collision detection and inputs the resulting finger contact forces into the FCU. The contact force targets are subsequently converted by the CyberGrasp FCU into analog currents, which are amplified and sent to one of five electrical actuators located in an actuator housing unit. The actuator torques are transmitted to the user's fingertips through a system of cables and a mechanical exoskeleton worn on top of the CyberGlove. The exoskeleton has the dual role of guiding the cables using two cams for each finger as well as of serving as a mechanical amplifier to increase the forces felt at the fingertip. The exoskeleton is attached to the cable guides and to the CyberGlove through finger rings, a back plate, and Velcro strips. The exoskeleton cable arrangement allows only unidirectional forces to be applied to the fingertips, in the direction opposing hand closure. The maximum force that can be produced at each finger is 16 N within a spherical work envelope of 1 m radius. This represents a significant increase in dexterity as well as in the user's freedom of motion compared to those allowed by the force feedback arms previously described. A version of the CyberGrasp that puts the FCU and associated electronics in a backpack increases the user's freedom of motion even more. The limitations then become the small tracker range and the weight that must be carried around by the user (backpack unit, cables, and glove). Most important is the weight of the part of the system that must be worn on the arm, which at 539 grams can lead to user fatigue. Further drawbacks of the CyberGrasp are system complexity and cost (\$39,000) and the inability to simulate the weight and inertia of the grasped virtual object. Furthermore, cable backlash and friction-induced hysteresis reduce the mechanical bandwidth of the CyberGrasp to 40 Hz (down from 1000 Hz control bandwidth) [Turner et al., 1998].

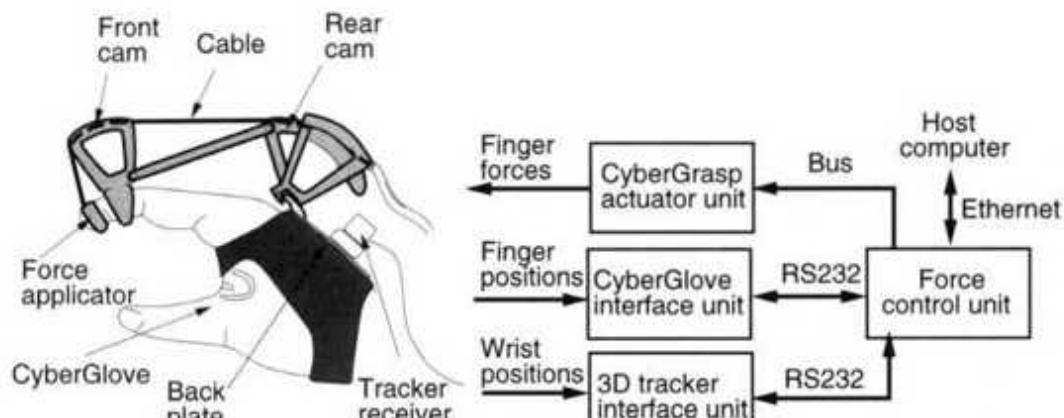


VC 3.4

3.3.3.5 The CyberForce. This device is an addition to the CyberGrasp in order to allow the simulation of object weight and inertia. As shown in Figure 3.35a, it consists of a mechanical arm attached to the desk and to the user's wrist at the CyberGrasp exoskeleton back plate. The mechanical arm position sensors are read by the FCU, eliminating the need for a separate 3D tracker. As illustrated in Figure 3.35b, the wrist and finger position data are sent by the FCU to the host computer over a LAN, as previously discussed. The resulting contact and weight/inertia force targets received from the host are sent to the CyberGrasp actuators and to the CyberForce actuator unit. The mechanical arm has three electrical actuators, which produce a maximum force of 8.8 N at the wrist. No torques can be produced, since the orientation degrees of freedom of the CyberForce are passive. The user's freedom of motion is reduced to the mechanical arm work envelope ($0.3 \times 0.3 \times 0.5$ m³), which is significantly smaller than that of the CyberGrasp. Furthermore, concerns related to potential accidents led to the integration of a safety switch that can disable the arm actuators when pressed by the user. Finally, the additional mechanical arm increases the system complexity and cost (\$56,000) above those of the CyberGrasp.

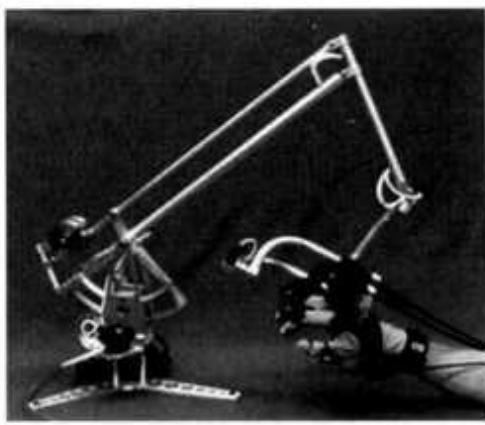


(a)

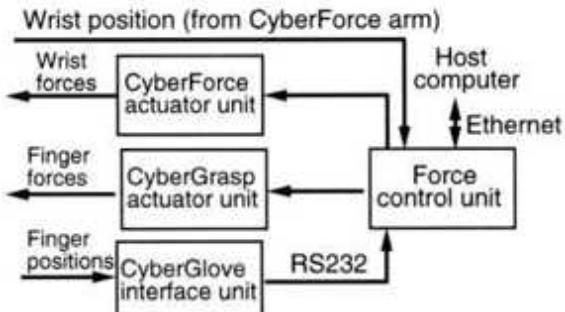


(b)

Fig. 3.34 The CyberGrasp force feedback glove. (a) Outside appearance. (b) The force feedback system. Adapted from Virtual Technologies [1999] and Kramer et al. [2000]. © 1999, 2000 Immersion Co. Reprinted by permission.



(a)



(b)

Fig. 3.35 The CyberForce force feedback system: (a) outside appearance; (b) the force feedback system. Adapted from Virtual Technologies [2001]. © 2001 Immersion Co. Reprinted by permission.

TABLE 3.5. Haptic Interfaces for the Hand

Product Name	Type of Feedback	Number of Actuators	Maximum Force (N)	Weight (g)	Bandwidth (Hz)	Price (10 ³ \$)
iFeel Mouse	Vibrotactile	1	1.18 @30 Hz	132	0–500	0.04
CyberTouch glove	Vibrotactile	6	1.2 N @125 Hz	142	0–125	15
DTSS X10	Temperature	≤8	NA	340	NA	20
WingMan 3D joystick	Force	2	3.3	NA	0–333	0.06
PHANTOM Desktop	Force	3	6.4	75	NA	16
PHANTOM 1.5/6.0	Force	6	8.5	90–108 (apparent)	15 (rotation)	57
Haptic Master	Force	3	250	NA	10	34
CyberGrasp glove	Force	5	16	539	40	39
CyberForce arm	Force	8	8.8 (translation)	NA	NA	56

Table 3.5 summarizes the characteristics of the haptic interfaces discussed in this chapter. It is easy to see that at the time of this writing there is a significant difference between the capabilities and price of consumer and professional haptic interfaces. This is in no doubt due to the novelty factor

as well as the small market for professional haptic interfaces. These will necessarily change in the future as haptic feedback becomes more accepted into the mainstream VR simulation domain.

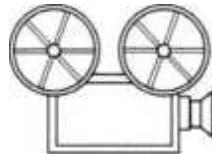
3.4 CONCLUSION

This chapter introduced VR-specific output devices, designed to provide graphics, auditory, and haptic feedback to the user. These interfaces vary in complexity from the Olympus HMD to CAVEs, from PC 3D sound cards to the HURON 3D audio workstation, and from tactile mice to the CyberForce. Many systems currently in the research and development stages have necessarily been left out. The discussion here was instead aimed primarily at familiarizing the student with the various principles, capabilities, and design tradeoffs of current commercial VR output interfaces. The source of the real-time feedback mediated by these interfaces is the computer (or group of computers) running the VR simulation. This VR engine is the subject of the next chapter.

3.5 REVIEW QUESTIONS

1. How is depth perceived by the human vision system? Make a drawing and explain.
2. What hardware is necessary to take the output of a PC graphics accelerator and display it on a monoscopic HMD?
3. What are differences between older head-mounted displays (HMDs) and newer face-mounted displays (FMDs) in terms of ergonomics and optics?
4. What is relationship between HMD field of view and resolution, and why is it important?
5. What is a Boom 3D display and how does it work? What advantages does it have compared to an HMD?
6. Draw and explain the architecture of the CAVE.

7. What are tiled displays? Give examples and explain their functioning.
8. What is the difference between stereo sound and 3D Sound?
9. What are the human auditory range cues?
10. What is the HRTF and how is it implemented by the Convoltotron?
11. Describe the functioning of 3D audio chips. Make a drawing and explain.
12. How do speaker-based 3D audio systems work?
13. How does touch feedback differ from force feedback?



VC 3.5

14. What human sensors are responsible for touch?
15. What are the ideal characteristics of haptic feedback actuators?
16. Describe the CyberTouch Glove and the iFeel Mouse. How do they work? What tactile sensations can they reproduce? Make a drawing and explain.
17. What is temperature feedback? How is it realized? Make a drawing and explain.
18. What is the PHANToM haptic interface? What kinds of actuators and transmission mechanism does it use? Make a drawing and explain.
19. What is the difference between the PHANToM Desktop and 1.5/6.0 models?

20. What is the CyberGrasp? Where are its actuators placed and how is force feedback produced in this case?
21. Why does the CyberForce add a mechanical arm to the wrist? What are its advantages and disadvantages compared to other approaches?

REFERENCES

- Akka, R., 1992, "Utilizing 6D Head-Tracking Data for Stereoscopic Computer Graphics Perspective Transformations," StereoGraphics Corp., San Rafael, CA.
- An, K.-N., L. Askew, and E. Chao, 1986, "Biomechanics and Functional Assessment of Upper Extremities," in Trends in Ergonomics/Human Factors III, Elsevier, Amsterdam, pp. 573580.
- Anon, 2001, "Head-Mounted Display Basics," Real Time Graphics, Vol. 10(2), pp. 6-9.
- Barco, 1999a, "The Vision Experience," company brochure, Barco Simulation Products, Kuurne, Belgium. Also online at www.barco.com.
- Barco, 1999b, "The Baron," company brochure, Barco Simulation Products, Kuurne, Belgium. Also online at www.barco.com.
- Belleman, R. G., B. Stolk, and R. de Vries, 2001, "Immersive Virtual Reality on Commodity Hardware," presented at ASCI 2001 Conference (ASCI2001), Heijen, The Netherlands, May/June 2001.
- Burdea, G., 1993, "Virtual Reality Systems and Applications" [Short Course], in Electro'93 International Conference, Edison, NJ.
- Burdea, G., 1996, Force and Touch Feedback for Virtual Reality, Wiley, New York.
- Burdea, G., and P. Couffet, 1993, La Réalité Virtuelle, Hermès, Paris.

Chapin, W., and S. Foster, 1992, "Virtual Environment Display for a 3D Audio Room Simulation," in SPIE '92 Stereoscopic Display and Applications, p. 12.

Cholewiak, R., and A. Collins, 1991, "Sensory and Physiological Bases of Touch," in M. Huller and W. Schiff (Eds.), *The Psychology of Touch*, Erlbaum, Mahwah, NJ, pp. 23-60.

Cruz-Neira, C., D. Sandin, and T. DeFanti, 1993, "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE," in Proceedings of SIGGRAPH '93, ACM SIGGRAPH, pp. 135-142.

Currell, C., 1992, "Virtual Audio: New Uses for 3-D Sound," in Proceedings of Cyberarts Conference, Pasadena, CA, pp. 104-108.

Cutkosky, M., and R. Howe, 1990, "Human Grasp Choice and Robotic Grasp Analysis," in S. Venkataraman and T. Iberall (Eds.), *Dextrous Robot Hands*, Springer Verlag, Berlin, pp. 5-31.

Cutt, P., 1993, "Tactile Displays: Adding the Sense of Touch to Virtual Environments," in Proceedings of Virtual Reality Systems '93 Conference, New York, pp. 120-122.

Daeyang, 2001, "Technical Specification: Cy-visor DH-4400VP," User Manual, Daeyang Co., Seoul, South Korea. Also online at <http://211.44.250.253/english/product/overview-m-1.html>.

Delaney, B., 2001, "Taking over the CAVE," *Real Time Graphics*, Vol. 10(4), pp. 12-13.

Dimension Technologies Inc., 2002, "Model 2018XLQ Specifications," online at www.dti3d.com/2018specs.asp.

Dresden 3D, 2001, "Dresden 3D Display Manual," Dresden 3D, Dresden, Germany. Also online at www.dresden3d.com.

Duda, R., 1997, "3-D Audio for HCI," San Jose State University, online paper available at www.engr.sjsu.edu/knapp/HCIROD3D/3D-home.htm.

Durlach, N., and A. Mayor (Eds.), 1995, Virtual Reality Scientific and Technological Challenges, National Academy Press, Washington, DC.

Eichenlaub, J., 1998, "A Lightweight, Compact 2D/3D Autostereoscopic LCD Backlight for Games, Monitor and Notebook Applications," in Proceedings of SPIE Stereoscopic Displays and Applications IX Conference, San Jose, CA, pp. 180-185.

Eichenlaub, J., and A. Martens, 1992, "3D Without Glasses," Information Display, Vol. 8(3), pp. 9-12.

Elsa Ag., 2000, "Elsa Ecomo4D," company brochure, Elsa Ag., Aachen Germany, 2000. Also online at www.elsa.com.

Fakespace Systems, 1998, "Virtual Model Displays," company brochure, Fakespace Systems Inc., Mountain View, CA.

Fakespace Systems, 1999, "Immersive WorkWall," company brochure, Fakespace Systems Inc., Mountain View, CA.

Fakespace Labs, 2001, "Boom3C," Fakespace Labs Inc., Mountain View, CA. Also online at www.fakespacelabs.com/productsboom3c.html.

FCS, 2001, "HapticMaster," company brochure, FCS Control Systems, Schiphol, The Netherlands. Also online at www.fcs-robotics.com.

Foster, S., 1998, Convoltotron User Manual, Crystal River Engineering, Groveland, CA.

Foster, S., 1992, "The Convoltotron Concept," in Proceedings of the Cyberarts Conference, Pasadena, CA, pp. 93-95,

Hatada, T., 1992, "Psychological and Physiological Analysis of Stereoscopic Vision," Journal of Robotics and Mechatronics, Vol. 4(1), pp. 13-19.

Hentschke, S., 2001, "Position-Adaptive Autostereoscopic Monitor (PAM)," U.S. Patent 6,307,585 B 1, October 23, 2001.

I-glasses, 2001, "3D Glasses," online at www.i-glasses.com.

Immersion Corporation, 2002a, "Vibrotactile Feedback Option for CyberGlove," online at www.immersion.com/products/3d/interaction/cybertouch.shtml.

Immersion Corporation, 2002b "Groundbreaking Haptic Interface for the Entire Hand," online at www.immersion.com/products/3d/interaction/cybergrasp.

Immersion Corporation, 2002c, "World's First Desktop Whole-Hand and Arm Force Feedback Device,"online at www.immersion.com/products/3d/interaction/cybergrasp.shtml.

Isdale, J., 2000, "3D Workstations," VR News, Vol. 9(1), Online at www.vrnews.com/issue archive/vrn0901/vrn0901tech.html.

Jandura, L., and M. Srinivasan, 1994, "Experiments on Human Performance in Torque Discrimination and Control," in Proceedings of 1994 ASME Winter Annual Meeting, Chicago, pp. 369-375.

Johnes, L., and Hunter, I., 1992, "Human Operator Perception of Mechanical Variables and Their Effects on Tracking Performance," Advances in Robotics, DSC-Vol. 42, ASME WAM, New York, pp. 49-53.

Julesz, B., 1971, Foundations of Cyclopean Perception, University of Chicago Press, Chicago.

Kalawsky, R., 1993, The Science of Virtual Reality and Virtual Environments, Addison-Wesley, Ltd., UK.

Kaiser Electro-Optics, 2001, "ProView XL35 & XL50," company brochure, Kaiser ElectroOptics Inc., Carlsbad, CA. Also online at www.keo.com/proviewxl3550.htm.

Kraemer, A., 2001, "Two Speakers Are Better than 5.1," IEEE Spectrum, 2001(May), pp. 71-74.

Kramer, J., M. Yim, M. Tremblay, and D. Gomez, 2000, "Force-Feedback Interface Device for the Hand," U.S. Patent 6,042,555, March 28, 2000.

Lake Technology, 2002, "Huron Virtual Acoustic Reality," company brochure, Lake Technology Ltd., Sydney, Australia. Also online at www.laketechnology.com.

Logitech Co., 2001, "WingMan Force 3D," online at www.logitech.com/cf/products/productoverview.cfm/2908.

Logitech Co., 2002, "iFeel Mouse," online at www.logitech.com/cf/products/productoverview.cfm/98.

Lubell, P., 2000, "D-Cinema," IEEE Spectrum, 2000(March), pp. 73-78.

Massie, T., and K. Salisbury, 1994, "The PHANToM Haptic Interface: A Device for Probing Virtual Objects," in ASME Winter Annual Meeting, DSC-Vol. 55-1, pp. 295-300.

McDowall, I., M. Bolas, S. Pieper, S. Fisher, and J. Humphries, 1990, "Implementation and Integration of a Counterbalanced CRT-Based Stereoscopic Display for Interactive Viewpoint Control in Virtual Environment Applications," in Proceedings of Stereoscopic Displays and Applications, SPIE Vol. 1256, pp. 136-146.

Monkman, G., and P. Taylor, 1993, "Thermal Tactile Sensing," IEEE Transactions on Robotics and Automation, Vol. 9(3), pp. 313-318.

NASA, 1988, "Autostereoscopic Displays for Scientific Visualization," Tech Briefs, Vol. 22(4), April.

n-vision Inc., 1998, "Datavisor," company brochure, n-vision Inc., Reston, VA. Also online at www.nvis.com/dv.htm.

NVIS Inc., 2003, "Virtual Binoculars," company brochure, NVIS Inc., Reston, VA. Also online at www.nvis.com/literature/vbsx_brochure.pdf.

Olympus Co., 2001, "FMD-200 Brochure," online at www.olympusamerica.com/cpg-section.

Ouh-Young, M., M. Pique, J. Hughes, N. Srinivasan, and F. Brooks, Jr., 1988., "Using a Manipulator for Force Display in Molecular Docking," in Proceedings of the 1988 IEEE International Conference on Robotics and Automation, IEEE Press, New York, pp. 18241829.

Panoram Technologies, 2000a, "PV290 DSK," company brochure, Panoram Technologies Inc., Sun Valley CA. Also online at www.panoramtech.com.

Panoram Technologies, 2000b, "PanoWall," company brochure, Panoram Technologies Inc., Sun Valley CA. Also online at www.panoramtech.com.

Panoram Technologies, 2001, "Seamless Matrix Blending," online at www.panoramtech.com/tech/info/smb/index.htm.

Pape, D., C. Cruz-Neira, and M. Czernuszenko, 1997, "CAVE User's Guide," University of Illinois at Chicago, online at www.evl.uic.edu/pape/CAVE/prog/CAVEGuide.html.

Pimentel, K., and K. Teixeira, 1993, Virtual Reality: Through the New Looking Glass, Windcrest McGraw-Hill, New York.

Phywe Systeme, 2003, "Peltier Heat Pump," Laboratory Experiment LEP 4.1.08, Gottingen, Germany. Online at www.phywe.com.

Quantum3D, 2001, "Multi-Channel Synchronization in Visual Simulation Systems," Quantum 3D, San Jose, CA.

Robinett, W., and J. Rolland, 1992, "A Computational Model for the Stereoscopic Optics of a Head-Mounted Display," Presence-Teleoperators and Virtual Environments, Vol. 1(1), pp. 45-62.

Rosenberg, L., 1998, "Method and Apparatus for Providing High Bandwidth, Low Noise Mechanical I/O for Computer Systems," U.S. Patent 5,731,804, March 24, 1998.

Rosenberg, L., and K. Martin, 2001, "Tactile Mouse Device," U.S. Patent 6,211,861, April 3, 2001.

Schott, D., 1999, "Reflective LCOS Light Valves and Their Application to Virtual Displays," in Proceedings of IDRC/Eurodisplay Conference, Berlin, Germany.

SensAble Technologies Inc., 2002a, "The PHANToM Desktop," online at www.sensable.com/haptic_products/image_s/p_2mol.jpg.

SensAble Technologies Inc., 2002b, "PHANToM 1.5/6.0 DOF," online at www.sensable.com/haptic_products/image_s/6dof.jpg.

SEOS, 2001, "SEOS Demonstrates LUCID Griffin, Theseus" Real Time Graphics, Vol. 9(6) pp.13-14, January.

Scow, K., 1988, "Psysiology of Touch, Grip and Gait," in J. G. Webster (Ed.), Tactile Sensing for Robotics and Medicine, Wiley, New York, pp. 13-40.

Shimoga, K., 1992, "Finger Force and Touch Feedback Issues in Dextrous Telemanipulation," in Proceedings of NASA-CIRSSE International Conference on Intelligent Robotic Systems for Space Exploration, Troy, New York, pp. 159-178.

Srinivasan, M., and J. Chen, 1993, "Human Performance in Controlling Normal Forces of Contact with Rigid Objects," Advances in Robotics, Mechatronics, and Haptic Interfaces, DSC-Vol. 49, pp. 119-125.

Stereographics, 2001, "CrystalEyes 3," company brochure, Stereographics Co., San Rafael CA. Also online at www.stereographics.com.

Tan, H., M. Srinivasan, B. Eberman, and B. Cheng, 1994, "Human Factors for the Design of Force-Reflecting Haptic Interfaces," in Proceedings of

ASME WAM, DSC-Vol. 55-1, ASME, New York, pp. 353-360.

Tremblay, M., and M. Yim, 2000, "Tactile Feedback Man-Machine Interface Device," U.S. Patent 6,088,017, July 11, 2000.

Trimension Systems Ltd., 2000, "V-Domes-Spherical VR Environments," online at www.trimension-inc.com.

Trimension Systems Ltd., 2001, "V-Desk 6," online at www.trimension-inc.com.

Turner, M., D. Gomez, M. Tremblay, and M. Cutkosky, 1992 "Preliminary Tests of an Armgrounded Haptic Feedback Device in Telemanipulation," in Proceedings of ASME IMECE, DSC-Vol. 64, Orlando, FL, pp. 145-149.

Virtual Research Systems, 1999, "WindowVR," company brochure, Virtual Research Systems Inc., Santa Clara, CA. Also online at www.virtualresearch.com.

Virtual Technologies, 1999, "CyberGrasp User's Guide," Virtual Technologies Inc., Palo Alto, CA. Also online at www.immersion.com/manuals/cybergrasp-usersguide_v2.0.pdf.

Virtual Technologies, 2001, "CyberForce User's Guide," Virtual Technologies Inc., Palo Alto, CA. Also online at www.immersion.com/manuals/cyberforceusersguide-v1.3.pdf.

van der Linde, R., 2002, "A New High-Performance Haptic Interface, Called the HapticMaster," in Eurohaptics'02, Edinburgh, U.K., pp. 1-5.

Wenzel, E., 1992a, "Localization in Virtual Acoustic Display," Presence-Teleoperators and Virtual Environments, Vol. 1(1), pp. 80-107.

Wenzel, E., 1992b, "Launching Sounds into Space," in Proceedings of Cyberarts Conference, Pasadena, CA, pp. 87-93.

Wiker, S., E. Hershkowitz, and J. Zik, 1989, "Teleoperator Comfort and Psychometric Stability: Criteria for Limiting Master-Controller Forces of

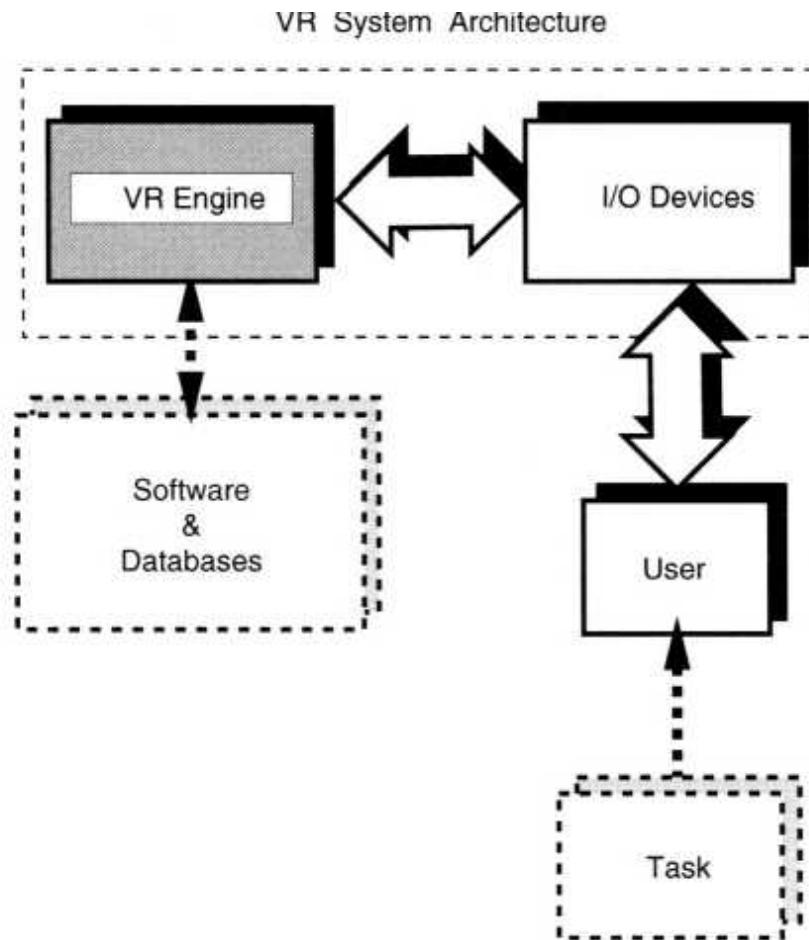
Operation and Feedback During Telemanipulation," in Proceedings of NASA Conference on Space Telerobotics, NASA, Greenbelt, MD, Vol. 1, pp. 99-107.

Younse, J., 1993, "Mirrors on a Chip," IEEE Spectrum, 1993 (November), pp. 27-31.

Zerkus, M., B. Becker, J. Ward, and L. Halvorsen, 1993, "Temperature Sensing in Virtual Reality and Telerobotics," Virtual Reality Systems, Vol. 1(2), pp. 88-90.

CHAPTER 4

COMPUTING ARCHITECTURES FOR VR



The last two chapters described devices used to mediate the user's input into, and feedback from, the VR simulation. Now we look at the computing hardware supporting such real-time interaction, which we call the "VR engine." This term is an abstraction, which corresponds to various physical hardware configurations, from a single computer to many networked computers supporting a given simulation.

Definition The VR engine is a key component of any VR system, which reads its input devices, accesses task-dependent databases, performs the required real-time computations to update the state of the virtual world, and feeds the results to the output displays.

During a VR simulation it is impossible to predict all user's actions and store all corresponding synthetic world states in memory. Therefore the virtual world is created (and deleted) in real time. Human factors studies suggest a minimum rate at which the world images (or frames) should be presented to the user. For smooth simulations at least 24 (the frame rate of movies) or, better, 30 frames/sec need to be displayed [Fuchs, 1992]. Therefore the VR engine needs to recompute the virtual world every 33 msec! This process alone results in a large computational load that needs to be handled in parallel with other tasks (such as I/O communication and storage).

Just as important for VR interactivity is the total simulation latency. This is the time between a user's input and the corresponding system feedback. Total system latency is the sum of the effects of sensor latency, transmission delays (to and from the VR engine), plus the time it takes to recompute and display a new world state (through graphics, audio, and haptic feedback). For the visual channel, for example, if total latency is over 100 msec, then the simulation quality is degraded significantly. Large latencies may even result in user dizziness and simulation sickness (discussed in Chapter 7).

Low latency and fast graphics, haptics, and audio refresh rates require a VR engine that has a powerful computer architecture. These architectures are designed around a rendering pipeline, a concept which we discuss first. Subsequently this chapter details PC-based and workstation-based VR engines, with emphasis on graphics. Finally, we describe distributed VR architectures. These are simulation systems designed for a single user or systems which make possible multiuser interaction in a single VR simulation. The discussion here concentrates on key concepts of what is a fastchanging area of technology.

4.1 THE RENDERING PIPELINE

The term rendering is generally associated with graphics. It represents the process of converting the 3D geometrical models populating a virtual world into a 2D scene presented to the user [Moller and Haines, 1999]. Within the context of this book we extend the meaning of rendering to include other sensorial modalities, such as haptics [Popescu, 2001].

Pipeline architectures have been used for many years in central processing unit (CPU) design as a way to speed up processing through parallel execution of several task components. Pipelining the rendering process means dividing it into stages and assigning these stages to different hardware resources.

4.1.1 The Graphics Rendering Pipeline

Graphics rendering has three functional stages, as illustrated in Figure 4.1 [Moller and Haines, 1999]. The first stage is the application stage, which is done entirely in software by the CPU. It reads the world geometry database as well as the user's input mediated by devices such as mice, trackballs, trackers, or sensing gloves. In response to the user's input the application stage may change the view to the simulation or change the orientation of virtual objects (such as a virtual hand). Other application stage tasks are discussed later as part of the haptics rendering pipeline.

The application stage results are fed to the geometry stage, which can be implemented either in software or in hardware. This stage consists of model transformations (translation, rotation, scaling, etc.), lighting computations, scene projection, clipping, and mapping. The lighting substage calculates the surface color based on the type and number of simulated light sources in the scene, the lighting model, the surface material properties, atmospheric effects such as fog, etc. As a result of lighting computations the scene will be shaded and thus look more realistic.

The last stage in the graphics pipeline is the rasterizing stage, which is done in hardware, in order to gain speed. This stage converts the vertex information output by the geometry stage (such as color and texture) into pixel information needed by the video display. One important function of the rasterizing stage is to perform antialiasing in order to smooth out the jagged

appearance of polygon edges. Antialiasing subdivides pixels into subpixel regions with an assigned color. The subpixel colors are summed and determine the R, G, B percentages in the resulting color assigned to the display pixel [Sense8 Co., 2001]. The more subpixels are used, the larger is the computational load of the rasterizing stage and the better is the image quality. As can be seen in Figure 4.2, the antialiased image does seem of much better quality than its counterpart that was not antialiased.

In order to speed up performance, the rendering pipeline stages themselves use parallel architectures. The application stage can run on several CPUs, a solution that is recommended when haptic feedback is provided to the user. The geometry stage runs on a number of geometry engines (GEs), each processing a portion of the scene geometry (a number of graphics objects, or primitives). Each GE, when finished processing its primitives, reads new ones from the application stage output stored in a first-in, first-out (FIFO) buffer. It then writes its results to the rasterizing stage FIFO buffer. This second buffer is read periodically by rasterizer units (RU's), which form the third stage of the rendering pipeline. Each RU renders a patch of frame pixels assigned to it, writes data to the frame (output) buffer, then reads new data from the FIFO buffer, and so on. The frame buffer contains the pixel color information (stored in its color buffer) and the depth (or z value) (stored in the Z buffer). Modern architectures use double buffering, meaning that there is a front buffer and a back buffer. The front buffer stores the image being displayed while at the same time a new scene is being written into the back buffer. Then the back buffer is displayed and a new image is rendered into the front buffer, and so on. This is done to reduce the flicker associated with the display hardware refresh.

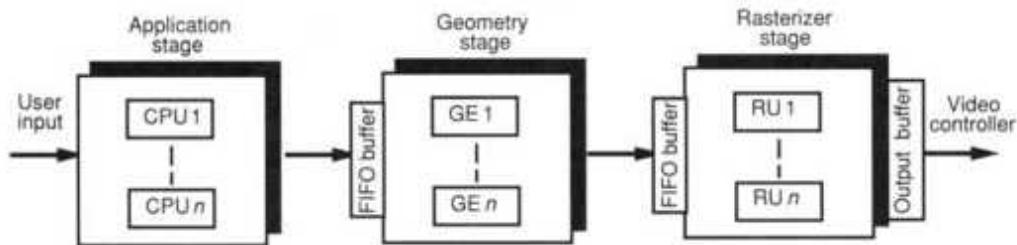


Fig. 4.1 The three stages of the OpenGL graphics rendering pipeline. Adapted from Moller and Haines [1999]. Reprinted by permission.

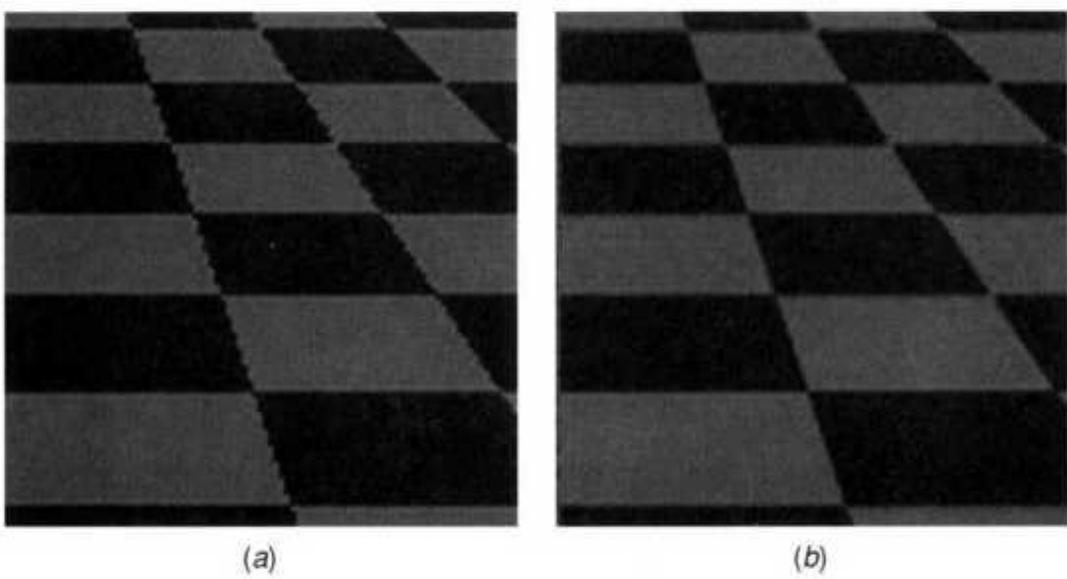


Fig. 4.2 The effect of antialiasing on image quality: (a) aliased polygons; (b) antialiased polygons. From Sense8 Co. [2001]. Reprinted by permission.

4.1.1.1 A Rendering Pipeline Example. An example that implements the geometry and rasterizing stages in hardware is the HP Visualize fx card illustrated in Figure 4.3 [Scott et al., 1998]. The CPU output data are received by an interface chip, which then directs them to the geometry board and to the texture accelerator. The least-busy GE on the geometry board starts working on the 3D data and when done sends them back to the interface chip. Here they are converted from floatingpoint to integer format necessary for the rasterizer. The two texture chips compute perspective-corrected textures, magnify the data, and pass them to the rasterizer, or store them in texture cache memory. The texture cache is reused for other objects, speeding up the pipeline. The RUs read data from the interface chip and from the texture chips, convert them to pixel format, and send them to the frame buffer. Finally, the video chip maps pixel colors to true colors (24-bit format), performs the D/A conversion, and does the video timing (synchronization).

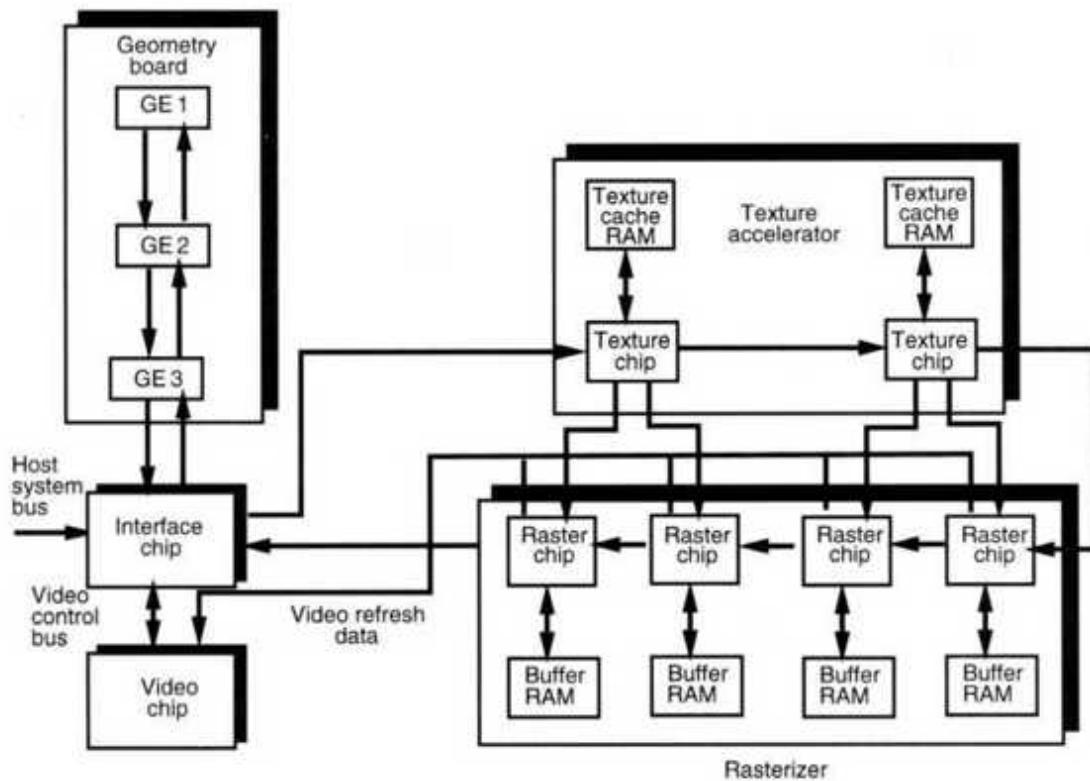


Fig. 4.3 The HP Visualize fx pipeline architecture. Adapted from Scott et al. [1998]. © 1998 Hewlett Packard Co. Reprinted by permission.

4.1.1.2 Graphics Pipeline Bottlenecks. The foregoing graphics pipeline discussion assumed an ideal case in which there is a perfect match between the output rate of one stage and the input rate of the following pipeline stage. In other words, the two FIFO buffers are never empty, which would correspond to a "starved" pipeline stage situation. In reality one of the three stages will be the slowest, will operate at 100%, and will dictate the overall pipeline throughput. This is called the bottleneck stage and needs to be identified in order to improve efficiency.

Before discussing techniques for finding the bottleneck stage, let us first look at the difference between the output of an ideal and that of a real pipeline. For the ideal case, illustrated in Figure 4.4a, the frame refresh rate is always inversely proportional to the scene complexity. For example, let us consider a moderately complex virtual world and a pipeline performance of only 300,000 Gouraud-shaded polygons/sec. The resulting refresh rate is

about 30 frames/sec when the scene complexity is about 10,000 polygons/scene (dashed line). If the scene complexity is reduced, the refresh rate grows exponentially [Burdea, 1993]. If the same pipeline needs to render a stereo scene, the performance drops by half because two images need to be computed by a single graphics board. Now the scene needs to have at most 5,000 polygons/scene for a stereo refresh rate of 30 frames/sec (neglecting the 20% synchronization overhead [Pimentel and Teixeira, 1993]).

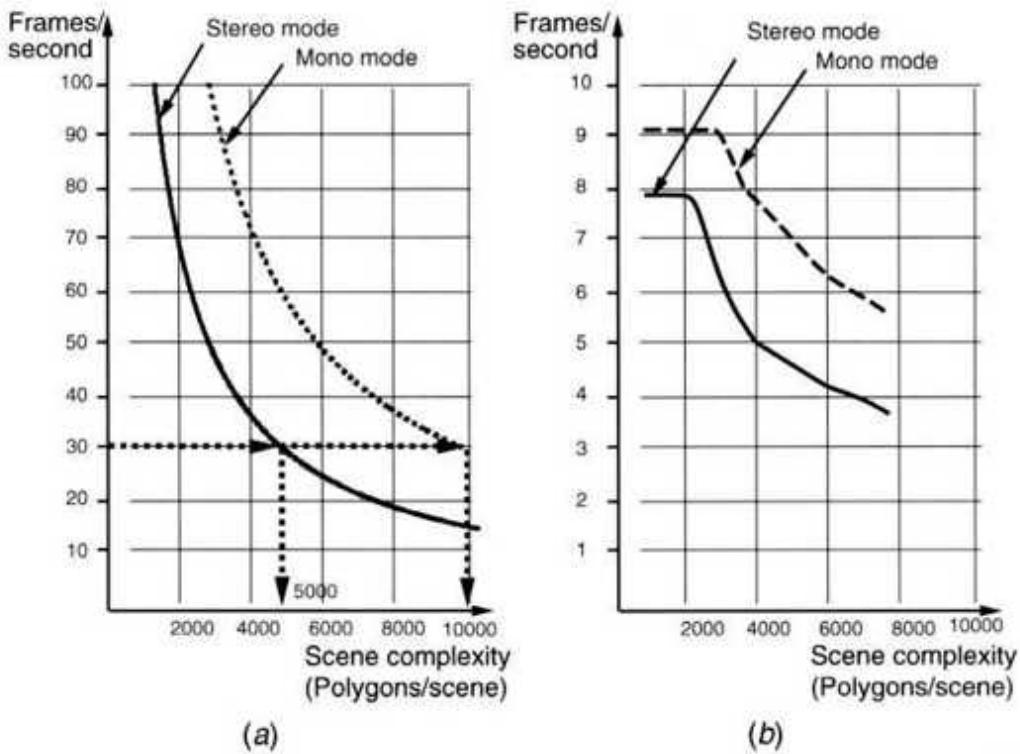


Fig. 4.4 Ideal versus real pipeline output as a function of scene complexity.
 (a) Ideal pipeline output. From Burdea [1993]. Reprinted by permission. (b) HP 9000 pipeline output. From Dinsmore et al. [1994]. Reprinted by permission.

Let us now consider the real case of an HP 9000 graphics workstation with a single graphics pipeline, illustrated in Figure 4.4b [Dinsmore et al., 1994]. It can be seen that there is a performance drop when the rendering mode is switched from monoscopic to stereoscopic, as previously discussed. What is interesting, however, is the flattening of the curves (in

both modes) when the scene complexity is below approximately 3,000 polygons. This unexpected artifact is due to a bottleneck in the application stage, and the pipeline is said to be CPU-limited. The slow CPU in these older computers operates at 100% and limits the maximum scene refresh rate to about 8 or 9 frames/sec, even for very low polygon counts.

Pipeline bottlenecks are by no means limited to the application stage. If, for a given CPU and graphics accelerator, the frame refresh rate increases when the number of light sources in the scene is reduced, the bottleneck is in fact in the geometry stage. Such a pipeline is said to be transform-limited [Moller and Haines, 1999]. Finally, if the pipeline output increases when the size or resolution of the display window is reduced, the bottleneck is in the rasterizing stage. In this case the pipeline is said to be fill-limited.

4.1.1.3 Graphics Pipeline Optimization. Once the bottlenecks are identified, measures need to be taken to eliminate them during an iterative process called pipeline optimization. One way to eliminate bottlenecks in the application stage is to replace the CPU with a faster one or to add another CPU (assuming this is supported by the hardware and operating system). Otherwise, keeping the current CPU requires that its load be reduced. One way to do that is to reduce the scene complexity by using 3D models with a lower polygonal count. This concept is illustrated in Figure 4.5, which depicts two versions of a virtual dinosaur [Viewpoint Co., 2002]. The high-resolution version (with 134,752 polygons) differs from the low-resolution one (1,016 polygons) by its head and legs detail. Nevertheless, the low-resolution model clearly conveys the idea of a dinosaur, while requiring the CPU to process 133 times fewer polygons!

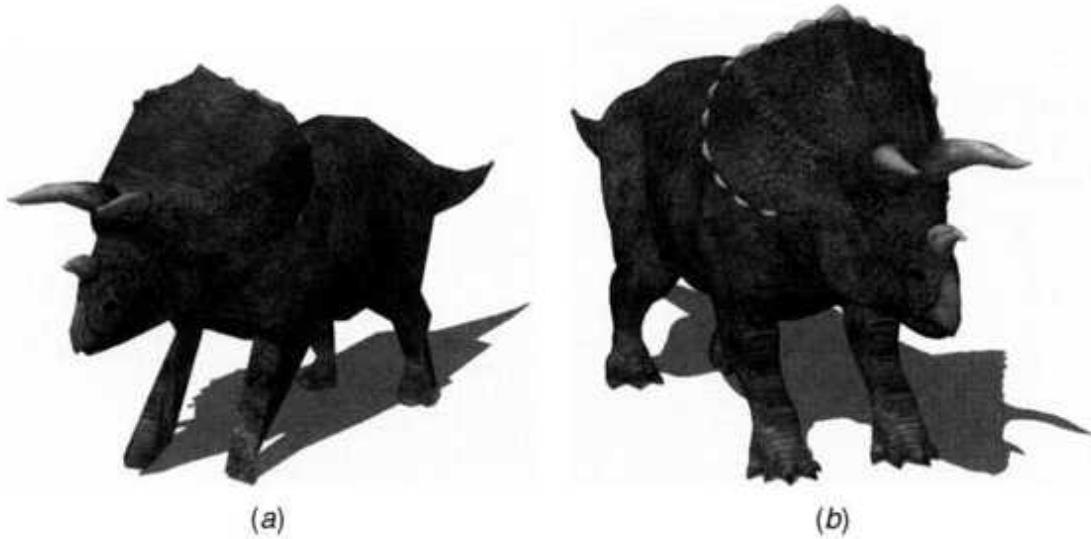


Fig. 4.5 A virtual dinosaur model: (a) low-resolution version (1,016 polygons); (b) highresolution version (134,752 polygons). © Viewpoint Corporation. Reprinted by permission.

Another way to reduce the load of the application stage is to optimize the simulation software. This can sometimes be done by the compiler, but it can always be done through clever programming. One way to reduce the CPU processing load is to use lower precision arithmetic instead of double-precision. Another way to improve pipeline performance is to write the code in a way that minimizes the number of divisions. Other real-time rendering software optimization techniques are given in Moller and Haines [1999].

If the pipeline bottleneck happens to be in the geometry stage, one needs to look at the computation load assigned to the GEs. One major component is lighting of the virtual scene. Illumination describes the intensity of light reflected by a pixel on the simulated object surface and seen by the viewer. The Phong illumination model [Bui-Tuong, 1975] is illustrated in Figure 4.6. First consider the case where there is a single light source in the scene. The object surface is illuminated by direct light from this point source, which is then reflected along the vector R at an angle θ with the surface normal N . At the same time the object is illuminated by ambient light, which is partially reflected toward the viewer. The viewer's location relative to the object

surface is given by vector V . Then I_A , the intensity of light of wavelength λ seen by the viewer, is given by [Foley et al., 1990]

$$I_A = I_a K_a O_d A + f_a t_l p A [K_j O_d; \cos \theta + K_s O_s a, \cos^2 a] \quad (4.1)$$

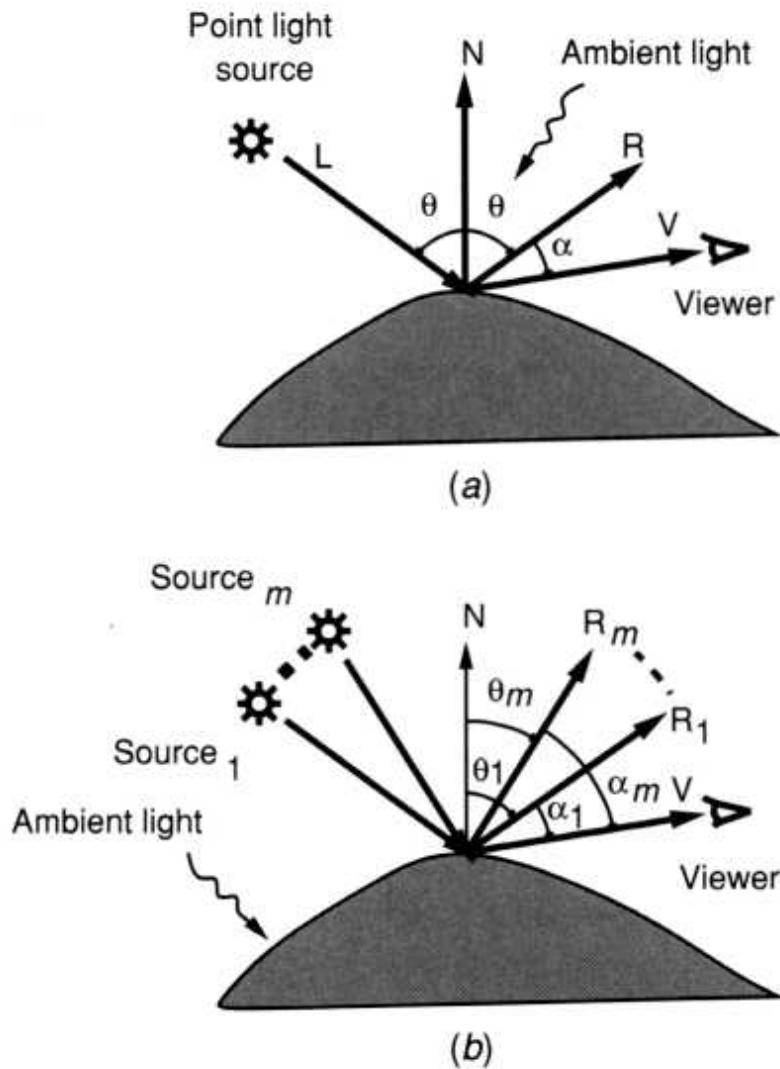


Fig. 4.6 The Phong illumination model: (a) single light source; (b) multiple light sources. Adapted from Bui-Tuong [1975]. © 1975 ACM Inc. Reprinted by permission.

where $I_a X$ is the intensity of ambient light,

K_a is the surface ambient reflection coefficient,

O_d ; is the object diffuse color,

f_a is the atmospheric attenuation factor,

$I_p A$ is the intensity of a point light source of wavelength X ,

K_d is the diffuse reflection coefficient,

K_s is the specular reflection coefficient, and

$O_s a$ is the specular color.

Specular reflection models light reflected from shiny surfaces such as a car exterior or a mirror. In this case there is a light spot close to the normal N where the color of the object is white. The reflected light is especially intense when the angle a between vectors R and V is small. The specular component of the reflected light I_a drops in value very quickly with increasing a . In Phong's model this is expressed by the power n of the cosine of a . The larger the power n , the more quickly does $\cos^n a$ and therefore the specular reflection drop to zero.

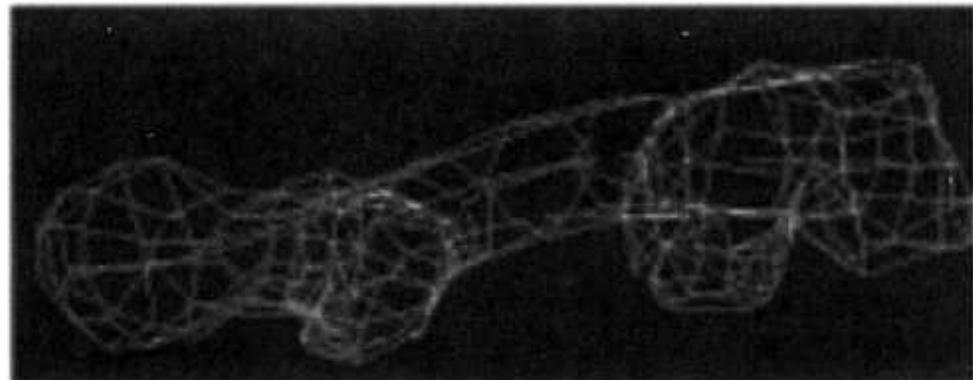
The Phong illumination model can be extended to the case where there are m simulated light sources, as in Figure 4.6b. In this case there will be multiple incidence angles $B_1, \dots, 0_{m-1}$ and corresponding angles a_1, \dots, a_m between the vectors R_1, \dots, R_m and the vector V . Therefore Equation (4.1) becomes

$$m h_i = I_a k_a O_d + f_a t' I_p [K_d O_d A \cos B_i + K_s O_s \cos^n a_i] r_i \quad (4.2)$$

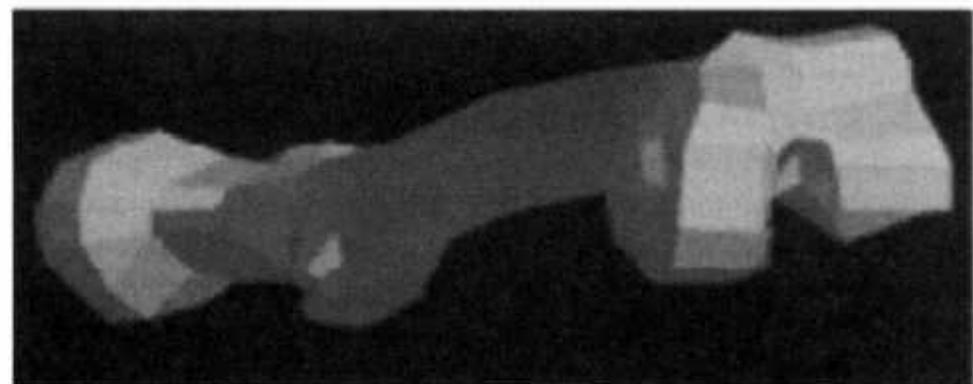
Equation (4.2) shows that the computational load for I_i grows with m , the number of simulated light sources. Therefore a given graphics pipeline will take longer to render the same scene if two light sources are present instead of one. The frame refresh rate will drop further if three sources are present, and so on. Eventually this may lead to a bottleneck in the geometry stage. In this case the pipeline can easily be optimized by reducing the number of virtual light sources in the simulation, assuming the simulation visual

requirements are met. Many graphics board manufacturers in fact specify the maximum number of light sources their product supports.

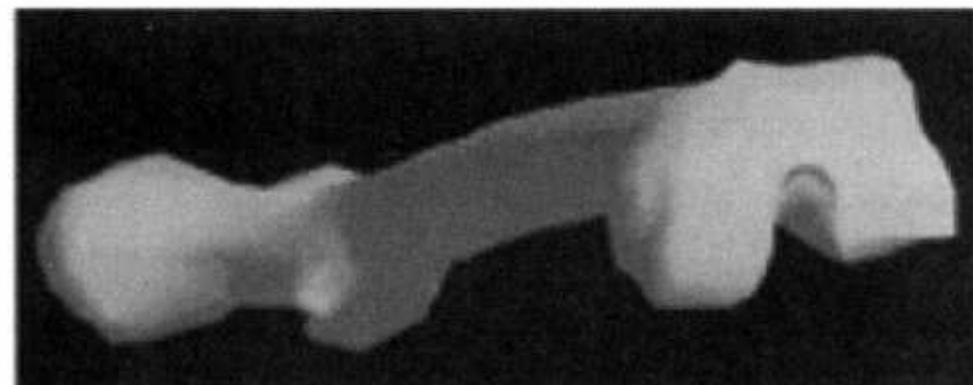
Another factor that influences the geometry stage computational load, and therefore the pipeline frame refresh rate, is the polygon shading mode. The simplest way to graphically represent a 3D object, such as the human femur depicted in Figure 4.7a, is in wireframe mode. The simplest shading mode for such an object is faceted or flat shading, as illustrated in Figure 4.7b. Here all points within a given polygon are given the II, for the center of that polygon. This simplification reduces the geometry stage computational load, but the resulting object looks tessellated. A more naturallooking appearance is obtained by intensity interpolation shading, or Gouraud shading [Gouraud, 1971]. This shading mode requires that the normal of each vertex of the polygonal mesh describing the surface of the virtual object be known. The vertex normal can be determined analytically or by averaging from polygon normals. The resulting object shape has improved smoothness, since color within a polygon is no longer constant (see Figure 4.7c). However, the computational load increases compared to flat shading, and the frame refresh rate decreases accordingly. Even more graphics realism can be obtained by Phong shading and texturing, which are discussed in Chapter 5.



(a)



(b)



(c)

Fig. 4.7 The human femur bone: (a) wireframe mode; (b) flat shading mode; (c) Gouraud shading mode. Code credit Daniel Gomez. From Burdea and Coiffet [1993]. © Editions Hermes. Reprinted by permission.

The pipeline throughput depends not only on the scene complexity, but also on the shape of the surface polygons. This is illustrated in Figure 4.8,

which shows the performance of a Silicon Graphics Inc. Onyx computer with Infinite Reality graphics accelerator as a function of surface elements. When the scene geometry is composed of 100-pixel independent quadrangles, then the rendering speed is roughly 2 million quads/sec. There is a small penalty for antialiasing and texturing, as these involve more computations in the rasterizing pipeline stage. However, when the scene is composed of meshed 50-pixel triangles, the same pipeline more than doubles its rendering speed. This is due to the architecture of the Infinite Reality accelerator, which is optimized to process triangle meshes (as discussed later in this chapter). Therefore, one way to optimize the pipeline is to use the type of polygon for which its rendering hardware was optimized.

Finally, if the pipeline is fill-limited (the bottleneck is in the rasterizing stage), optimization can be achieved by reducing either the size of the display window or the window resolution. Both approaches reduce the number of pixels in the displayed image, which means less work for the RUs. Assuming the rasterizer unit works at 100% capacity and the other two stages are at 80% capacity, the pipeline is said to be unbalanced. In order to balance the load, we can increase the model complexity (number of polygons). This will not change the throughput of a fill-limited pipeline, since the number of pixels in the display stays the same. By increasing the load of the application and geometry stages of the pipeline we will improve, however, the scene realism in terms of level of detail. Conversely, a transform-limited pipeline, where the GEs operate at 100%, can be balanced by increasing the display resolution or its size.

4.1.2 The Haptics Rendering Pipeline

The foregoing discussion was limited to the graphics feedback channel. Modern VR simulation systems implement additional sensorial modalities, such as haptics, that need to meet similar real-time constraints. This can be implemented through a multistage haptics rendering pipeline, as illustrated in Figure 4.9 [Popescu, 2001].

During the first stage of the haptics rendering pipeline the physical characteristics of the 3D objects are loaded from the database. These

include surface compliance, smoothness, weight, surface temperature, etc. The first stage of the pipeline also performs collision detection to determine which virtual objects collide, if any. Unlike the graphics pipeline, here only the colliding structures in the scene are passed down to subsequent pipeline stages.

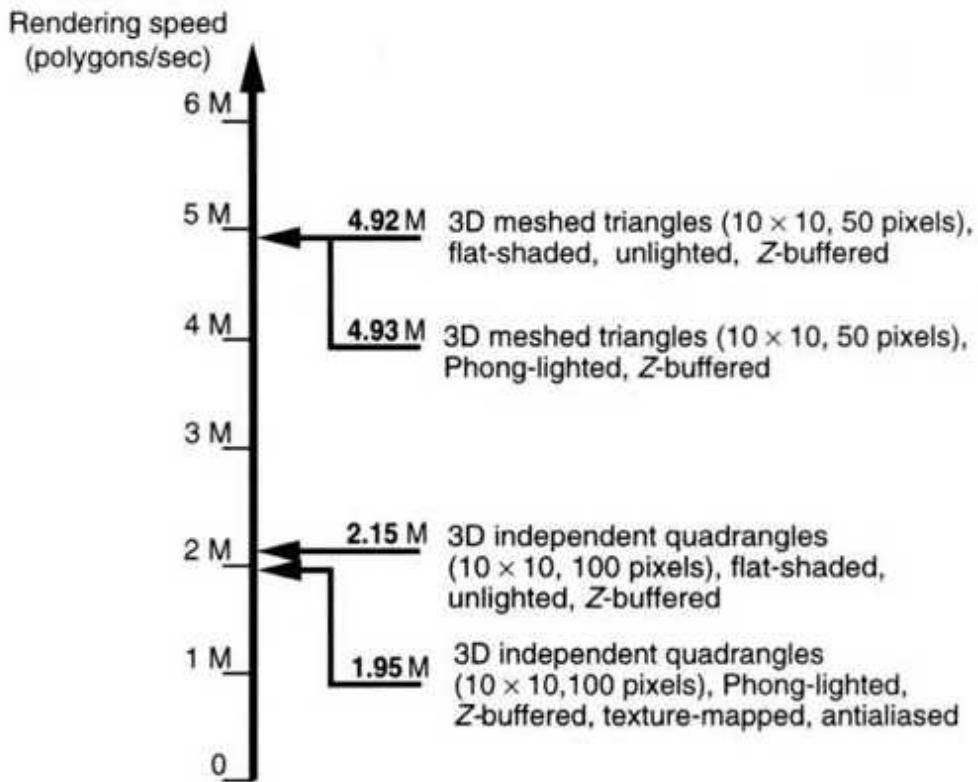


Fig. 4.8 Infinite Reality rendering pipeline performance versus shading mode and polygon size/type. Based on personal communication from Silicon Graphics, Inc., Rutgers University, New Brunswick, NJ (2002).

The second stage of the haptics rendering pipeline computes the collision forces, based on various physical simulation models. The simplest model is based on Hooke's law, where the contact force has a springlike dependence on the degree of surface deformation. Other models involve damping and friction forces and are more realistic, but also more computationally intensive. The more object contacts and the more complex the force shading, the higher is the chance of the pipeline becoming force limited. In this case optimization can be done by using simpler force models. The second stage

of the haptics rendering pipeline also involves force smoothing and force mapping. Force smoothing adjusts the direction of the force vector in order to avoid sharp transitions between polygonal surfaces, while force mapping projects the calculated force to the characteristics of the particular haptic display system.

The third and last stage of the haptics rendering pipeline is haptic texturing, which renders the touch feedback component of the simulation. Its computed effects, such as vibrations or surface temperature, are added to the force vector and sent to the haptics output display. Mapping with the haptic interface characteristics is important, and may lead, for example, to the substitution of contact forces by vibrations (as in the case of the CyberTouch glove). In conclusion, the haptics rendering pipeline has a much less standardized architecture compared to its graphics counterpart and reflects the rapidly evolving state of today's haptics technology.

4.2 PC GRAPHICS ARCHITECTURE

Personal computers form the largest computational base today, with hundreds of millions of units installed throughout the world. Most important for the real-time requirements of VR simulations are the PC CPU speed and the rendering capability of the resident graphics card. Within the past decade CPU speeds went from the 66 MHz of the Intel 486 CPU to the Pentium 4 with speeds in the multi-GHz range. Similarly, older PC graphics accelerators, such as the SPEA Fire, capable of only 7000 Gouraud-shaded polygons/sec, were replaced by graphics accelerators such as the ATI Fire GL2, which renders 27 million Gouraud-shaded triangles/sec.

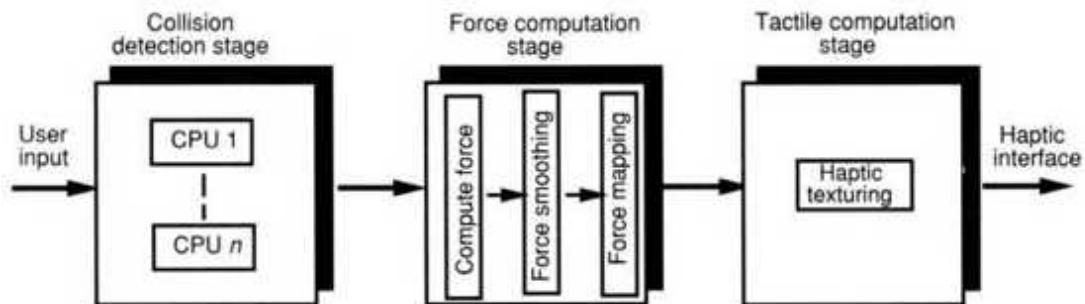


Fig. 4.9 The stages of the haptics rendering pipeline. Adapted from Popescu [2001].Reprinted by permission.

These facts, combined with a significant cost advantage, make the PC an excellent VR engine candidate. Until the late 1990s, however, the PC architecture had a communication bottleneck that hindered real-time graphics performance. This bottleneck was the slow peripheral component interface (PCI) bus, to which the graphics card and all other I/O devices were connected. The PCI bus bandwidth of only 132 MB/sec was inadequate to handle the traffic generated by all the devices connected to it. Specifically, memory transfers from the system RAM to graphics memory had to compete with hard drive, local area network (LAN), and other I/O traffic. The PCI bandwidth problem is compounded by the VR simulation's increasing need for image textures, which improve graphics realism, but require large memory data transfers.

Modern PC-based VR systems, such as the one illustrated in Figure 4.10 [Intel Co., 1999], solve the PCI bandwidth problem through the introduction of the Intel Accelerated Graphics Port (AGP) bus. The AGP bus operates at much higher bandwidth than the PCI bus and transfers textures and other graphics data directly from system RAM to the video memory (frame buffer) on the graphics card. The transfer is mediated by a memory controller chip (such as the Intel 820/850 chipset), which also allows simultaneous CPU-to-RAM traffic. The AGP transfer rate on current PCs is over 2 GB/sec using the AGP 8x bus [Intel Co., 2002]. At these high rates it is possible to reserve part of the RAM to serve as secondary video memory, such that less memory needs to be located on the graphics card. Furthermore, the AGP bus sideband address lines allow the graphics controller to send new read requests to the RAM while simultaneously receiving data from previous requests. This reduces the overall system latency. A second chip on the Intel chipset is an I/O controller hub, which allows traffic to the hard disk, serial ports, USB and game ports, parallel ports, and programmable I/O (PIO) ports. In this way, the PCI bus can better handle the rest of the devices connected to it.

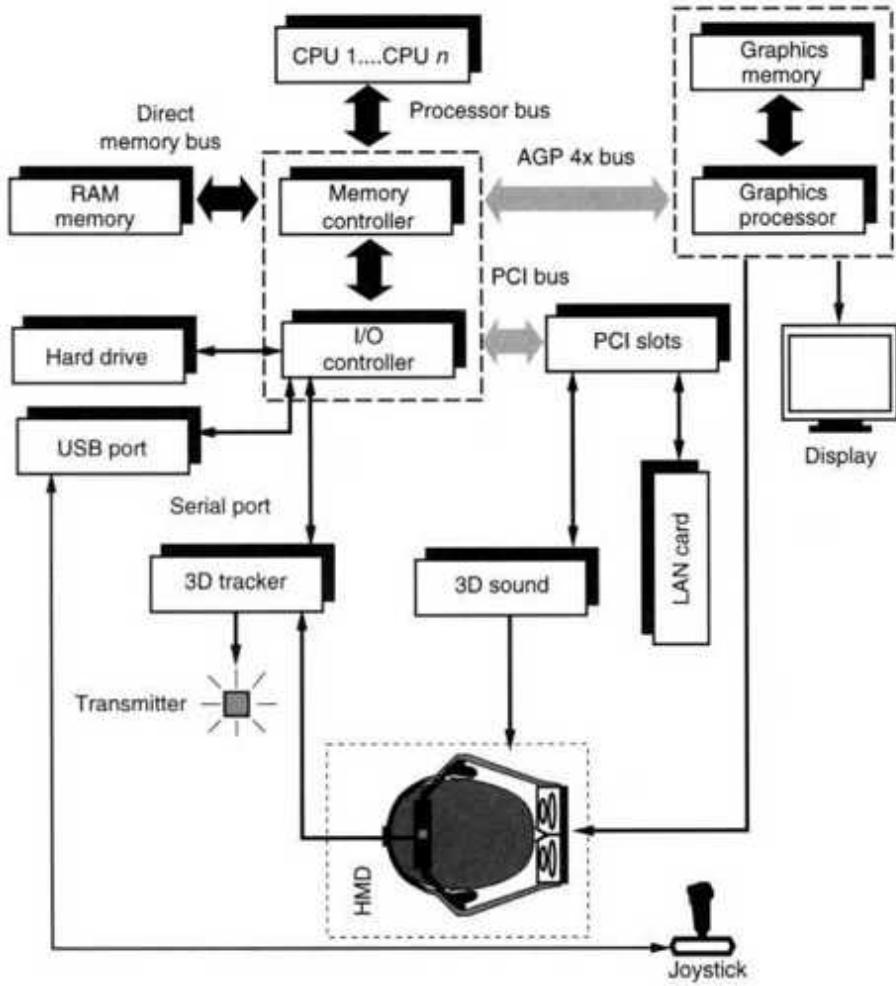


Fig. 4.10 PC VR engine. Adapted from Intel Co. [1999]. Reprinted by permission.

In order for the PC to be a VR engine, it needs to be integrated with the specialized UO devices used in the simulation. Figure 4.10 shows the user providing input through a (head) 3D tracker and a joystick. The user is receiving video feedback on an HMD, haptic feedback on the joystick, and audio feedback on headphones connected to a 3D sound card. The tracker is connected to a serial port and the joystick communicates through the USB port (as discussed in Chapter 3). The HMD connects to the graphics card RGB output port and the 3D sound card plugs directly into the PCI bus.

Integrating UO devices with a PC is not a simple task when these devices are shared among several PCs. Figure 4.11 illustrates such a case, namely

the basic cell of the VR Teaching Laboratory used in conjunction with this book. As can be seen, four PCs share one tracker source, by assigning one receiver for each VR engine. Other input devices are not shared (sensing glove, mouse, and force feedback joystick). Output to the user is provided through a combination of stereoscopic monitor and wired active glasses, through the joystick, and through 2D sound. Integrating the system means assigning the available serial ports to the tracker and the sensing glove, setting the tracker to work with four receivers (through its configuration switches), connecting the active glasses to the synchronization plug of the PC graphics card, setting the monitor to work in stereo mode, and connecting each PC to the LAN. The monitor configuration is done using the menu provided with the graphics accelerator driver, as detailed in the Laboratory Manual for this book. The LAN communication is necessary so that each PC receives tracker readings corresponding to its assigned sensor. Table 4.1 lists the specific hardware used by the VR Teaching Laboratory, its assigned communication ports, and its percentage of the total system cost (which is approximately \$5000 per PC-based VR engine).

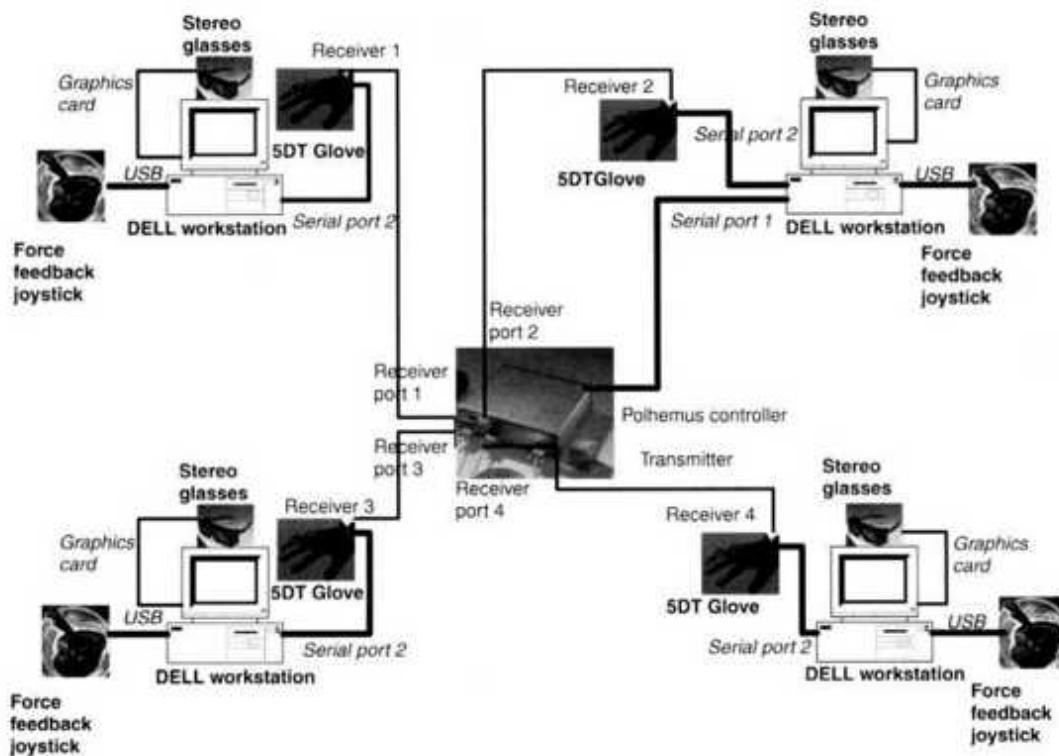


Fig. 4.11 PC-based VR engine used in the VR Teaching Laboratory. © Rutgers University. Reprinted by permission.

4.2.1 PC Graphics Accelerators

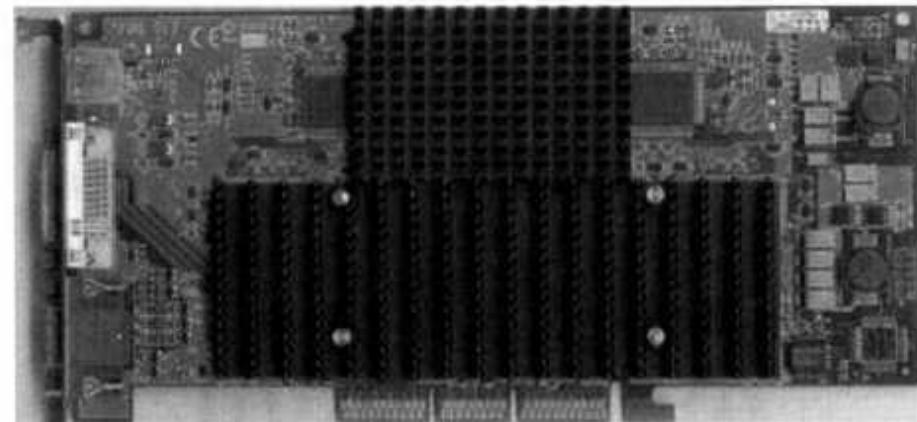
Graphics accelerator cards allow the real-time rendering of VR scenes on a PC. Within the scope of this book we are interested in cards that adhere to the OpenGL graphics standard, perform stereo rendering, and provide the synchronization signal necessary for use of active glasses. These are typically cards produced by third-party vendors, designed to retrofit mid-to-high-end PCs. Our discussion here focuses on two such cards, the ATI Fire GL2 [ATI Technologies, 2000] and the Elsa Gloria III, which uses NVIDIA Quadro2 Pro technology [NVIDIA, 2000]. We also look at the xBox architecture, a Pentium-based game platform that incorporates some interesting graphics technology.

4.2.1.1 The ATI Fire GL2. This midrange graphics accelerator, shown in Figure 4.12a [Scheffel and v. d. Weyden, 2000], is an example of traditional dual-processor architecture. Such architecture uses a geometry chip (IBM GT1000) to perform the geometry-stage pipeline computations and a rasterizer chip (IBM RC1000) for pixel-related computations and display. The chips are masked in Figure 4.12a by two very large coolers (metal heat sinks), which dissipate the chip-generated heat flux through convection and conduction. Also shown in the figure are the plug-in pins for the AGP slot on the PC motherboard (bottom) and the three output connectors on the left side (active glasses synchronization, digital display, and analog RGB video).

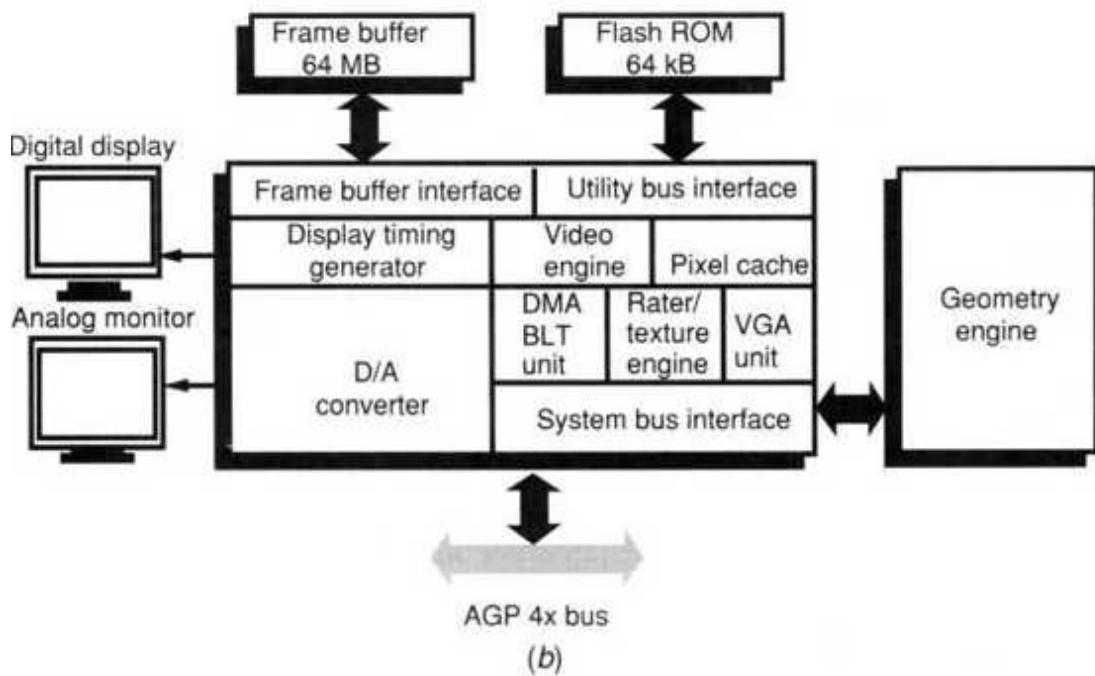
TABLE 4.1. Relative Cost of the PC VR System Used in the VR Teaching Laboratory (2002 Prices)

Product	Port	Percent of Budget
PC, 1.7 GHz, with ATI Fire GL2 graphics card	NA	48
Polhemus 3D tracker, Fastrack, four receivers	Com1	37
5DT sensing glove, five-sensor version, wired	Com2	10
Stereo Glasses, StereoEyes wired	Fire GL2	3
Force feedback joystick, Microsoft SideWinder	USB	2
Java and Java 3D	NA	0
VRML browser	NA	0

Figure 4.12b shows that the AGP data are received by the rasterizer system bus interface and sent to the geometry engine chip. Here the 190-MHz GT1000 performs 3D transformation and lighting computations (up to 16 light sources) at a rate of 27 million nontextured, Gouraud-shaded polygons/sec. The results are sent to the 120MHz IBM RC 1000 chip, which has the texture engine and pixel cache. Pixel data are transferred to the frame buffer memory at a fill rate of 410 Mpixels/sec, or 200 Mtexels/sec, based on a 256-bit-wide double-data rate (DDR) memory interface. The frame buffer contents are then sent to the digital output if a flat-panel digital display is used or to a 300-MHz D/A converter to obtain the RGB signal needed by analog monitors. Since the D/A converter operates at constant speed, the higher the image resolution (more pixels), the lower is the resulting screen refresh rate. For example, whereas the refresh rate is 100 Hz for a 1280 x 1024 monitor resolution, it drops to 60 Hz when the screen resolution grows to 1792 x 1344 [ATI Technologies, 2000].



(a)



(b)

Fig. 4.12 The Fire GL2 graphics board. (a) General appearance. From Scheffel and v. d. Weyden [2000]. © Tom's Guides Publishing AG. Reprinted by permission. (b) Block diagram. Adapted from ATI Technologies [2000]. © 2000 ATI Technologies Inc. Reprinted by permission.

4.2.1.2 The Elsa Gloria III. This is another midrange PC graphics card. As shown in Figure 4.13, it implements a newer design, which merges the geometry and rasterizing pipeline stages on a single 256-bit geometry processor unit (GPU). This rather large NVIDIA 250-MHz GE Force 2 chip is shown covered by a cooling fan, while other board components have

passive (heat sink) cooling. The placement of the AGP bus pins and of the digital and analog display connectors is similar to that of the previously discussed Fire GL2. Merging the two rendering chips used by the Fire GL2 into a single one offers a number of performance and cost advantages. The GE Force 2 GPU is capable of 31 million Gouraud-shaded polygons/sec, which is about the same as the shading capability of the Fire GL2. However, the GPU pixel fill rate is 1000 million pixels/sec and its texture fill rate is 2000 Million texels/sec. These rates are considerably larger than those of the Fire GL2, despite a frame buffer size that is the same for both cards (64 MB DDR). What is different is the faster Gloria III memory access, due to a faster memory clock (200 MHz vs. 120 MHz for the Fire GL2) and a memory bus speed of 6.4 GB/sec. Even at these fast fill rates the screen refresh rate is resolutiondependent. At the usual resolution of 1280 x 1024 the refresh rate is 120 Hz, whereas at the maximum resolution of 2048 x 1536 it drops to 75 Hz.

4.2.1.3 The xBox. This game platform, shown in Figure 4.14a, is similar to the Elsa Gloria III in its use of an NVIDIA GPU. The xBox has a Pentium III (733 MHz) CPU that communicates with the GPU over a 1-GBps front-side bus, as illustrated in Figure 4.14b [Abrash, 2000]. The GE Force 3-type GPU has a 300-MHz internal clock and renders up to 125 million Gouraud-shaded textured triangles/sec. This rate drops, however, to (only) 8 million triangles/sec when eight local light sources are simulated. Its fill rate of 1000 million pixels/sec is more than sufficient for the 640 x 480 TV resolution, allowing full-image antialiasing at a refresh rate of 60 Hz. An interesting feature of the xBox GPU is the programmable pipeline, which is a radical departure from the traditional fixed-function hardware previously described. The programmability of the rasterizing stage of the pipeline is realized through a pixel shader that complements the (traditional) vertex shader. The positive impact of the pixel shader on VR modeling tasks is discussed in more detail in Chapter 5.

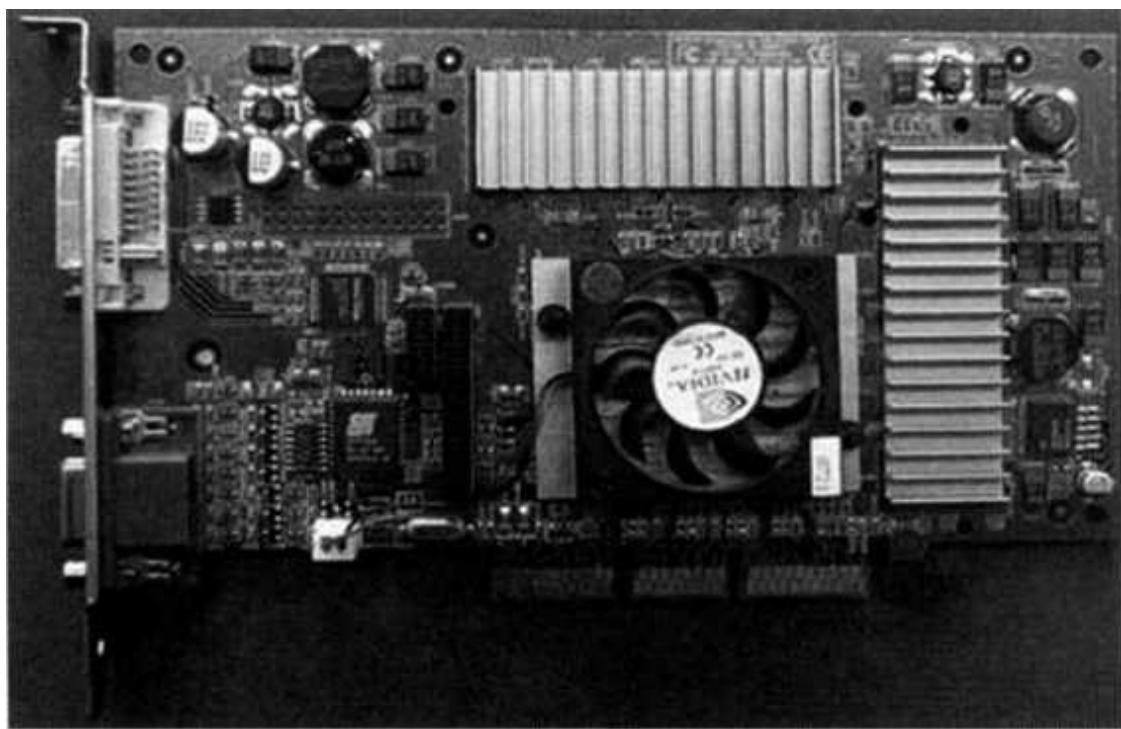
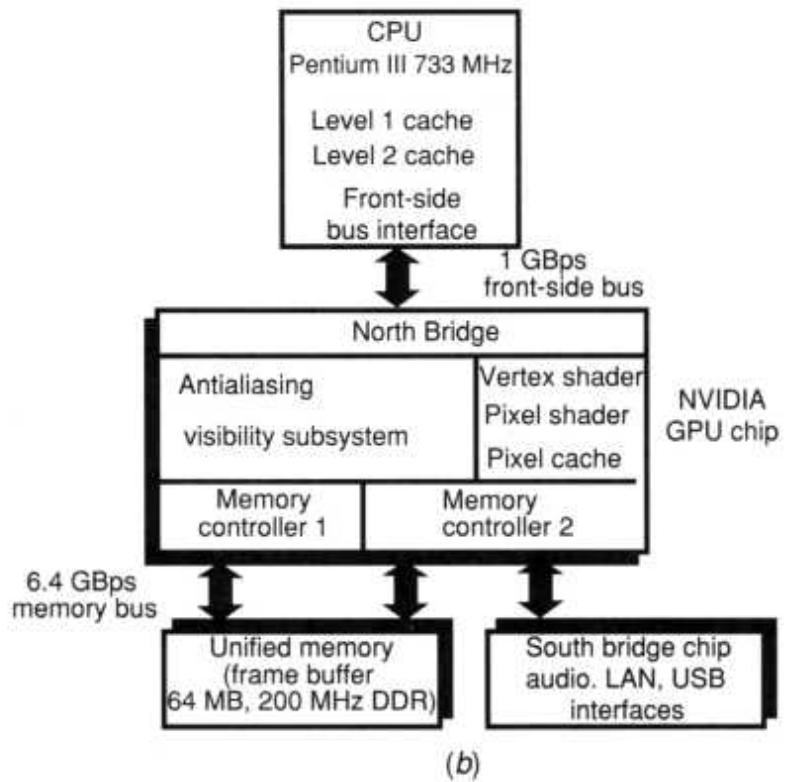


Fig. 4.13 The Elsa Gloria III graphics board. From Scheffel [2001a]. @ Tom's Guides Publishing, AG. Reprinted by permission.



(a)



(b)

Fig. 4.14 The Microsoft xBox game console: (a) general appearance. (b) block diagram. Adapted from Abrash [2000]. Used by permission from Microsoft. Microsoft, Xbox and the Xbox logos are either registered trademarks or trademarks of Microsoft Co. in the U.S. and/or in other countries. Reprinted by permission.

The xBox has a unified memory architecture (UMA), allowing the CPU, the GPU, as well as the xBox hard disk and DVD to share a single memory space. The GPU acts as the memory controller through a TwinBank memory architecture [NVIDIA, 2001]. This design utilizes two independent 64-bit memory controllers residing on the GPU, each accessing 128-bit data chunks for each DDR memory clock period. The resulting high data rate, coupled with fast frame buffer memory (200 MHz) and memory bus 6.4 GBps, satisfies the memory-intensive texturing operations associated with video-game graphics. A south bridge 1/0 hub manages the traffic to several USB ports, the network interface, as well as an NVIDIA Audio Processor. This media control processor (MCP) has a pair of high-performance multimedia DSPs, which render 3D audio and realistic sound effects.

4.2.2 Graphics Benchmarks

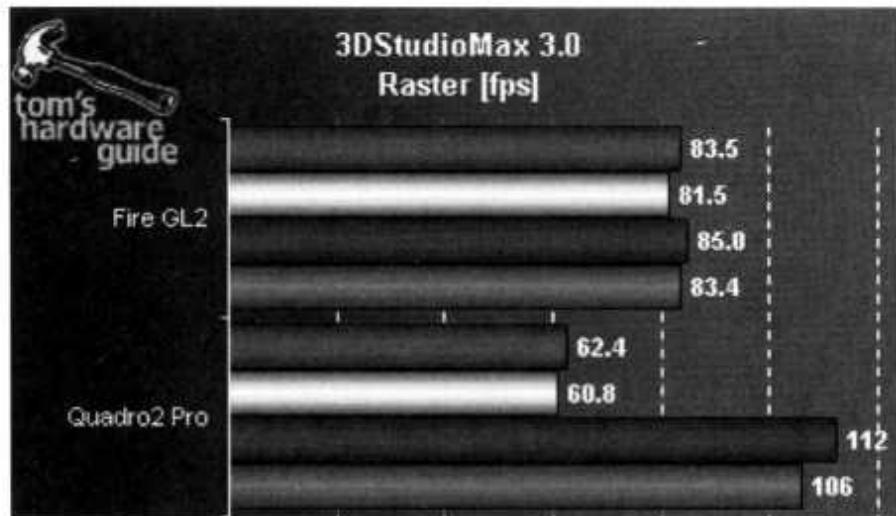
The foregoing discussion of professional-grade PC graphics cards is by necessity limited. Thus graphics accelerators such as the Fujitsu Celsius and the NEC TE4E have been left out. The very powerful 3Dlabs Wildcat II card is discussed later in this chapter. Choosing among the many options based on the highest polygon shading rates or pixel fill rates is misleading. A more productive way to rank graphics cards is to see how they perform in terms of frames/sec (fps) when running a given application.

Assume the user needs to choose between the Fire GL2 and the Elsa Gloria III (i.e., Quadro2 Pro). Although their price is similar (about \$1000), the Gloria III card has a (slightly) better shading rate and vastly faster fill rate, so it would be the winner. Figure 4.15 shows the fps count of the two cards when running 3D Studio Max 3.0 Raster and 3D Studio Max 3.0 Geom2 [Scheffel and v. d. Weyden, 2000].

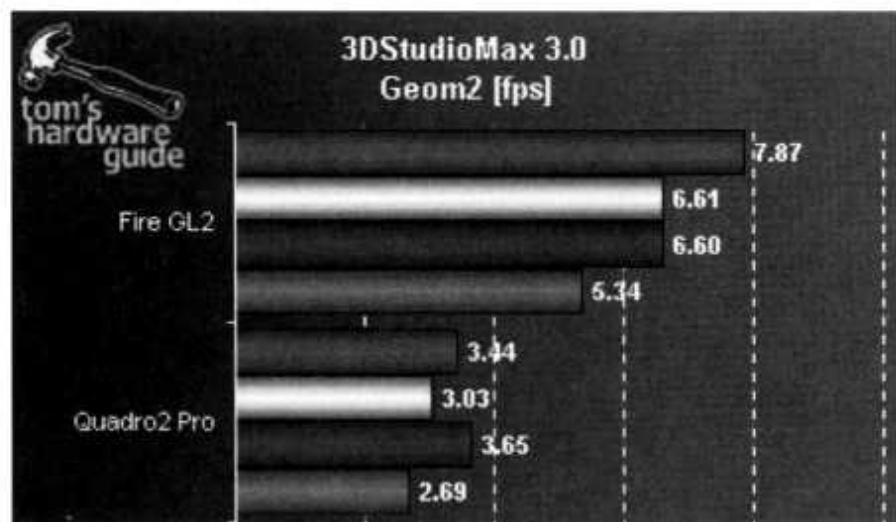
Each horizontal bar corresponds to progressively faster CPUs feeding the graphics pipeline (from 1 GHz for the bottom bar to 1.5 GHz at the top one). The size and resolution of the output display are maintained constant throughout the tests. As expected, the Quadro2 Pro is the winner in the first test (Figure 4.15a) because this is a rasterization-intensive application. However, the Fire GL2 is the winner in the second test, which is geometry-stage-intensive. Furthermore, CPU speed helps the geometry stage and the

Fire GL2 fps count for a 1.5-GHz Pentium 4 is almost three times that of the Quadro2 Pro running on a Pentium III 1-GHz PC.

This example illustrates the need to have a uniform way to judge graphics hardware and not rely exclusively on manufacturer specification sheets. Choosing among a myriad of potential application programs to determine fps counts is also impractical. What is needed is an accepted group of standard application programs specifically chosen to determine graphics hardware performance. Furthermore, an independent organization needs to develop a reproducible testing methodology, run such benchmark tests, and publish updated performance tables. Software vendors, notably Sense8 Co., developed benchmarks to see how their toolkits fared on various graphics platforms. Unfortunately, such benchmark tables were not updated as regularly as they should have been and were abandoned after the disappearance of the parent company, as happened with Sense8 Indy3D.



(a)



(b)

Fig. 4.15 3D Studio Max frame rates of Fire GL2 and Quadro2 Pro: (a) Raster; (b) Geom2. From Scheffel and v. d. Weyden [2000]. © Tom's Guides Publishing, AG. Reprinted by permission.

The Standard Performance Evaluation Co. (SPEC), a nonprofit organization, has been developing CPU and memory benchmarks for many years. In the mid 1990s they also developed the "viewperf" benchmark, a C program that measures 3D rendering performance for systems running under OpenGL [SPEC, 2002]. This has become the de facto standard in comparing graphics cards for VR applications. Viewperf version 6.1.2 is a set of six

virtual worlds or application viewsets. Each application viewset has a subset of tests associated with it as follows:

- Advanced Visualizer (AWadvs-04), an animation application containing 11 tests.
- Design Review (DRV-07), a 3D computer model review package containing five tests.
- Data Explorer (DX-06), a visualization application with 10 tests.
- Lightscape (Light-04), a radiosity visualization application with four tests.
- Mechanical CAD (MedMCAD-01), an immediate-mode MCAD design application with 12 tests.
- Pro/Designer (ProCDRS-03), a ProCDRS industrial design application with 10 tests.

TABLE 4.2. Viewperf Score Comparison of the Graphics Accelerators Presented in This Chapter ^a

System/Graphics	AWadvs-04	DRV-07	DX-06	Light-04	MedMCAD	ProCDRS
Dell 2.2 GHz Fire GL2	122	29	43	10	38	39
Compaq 2.2 GHz Quadro2 Pro	159	35	46	10	40	38
Dell 1.5 GHz Wildcat II 5110	200	34	45	10	51	54
Sun Blade 1000 900 MHz Expert3D	55	9	13	3	12	19
SGI Onyx2 IR R10K, 195 MHz	33	6	10	2	8	10

^aScores are in fps; they were extracted from the *viewperf 6.1.2* summary table at www.specbench.org/gpc/opc.data/summary.html for Fire GL2, Quadro2 Pro, and Wildcat II. Sun Microsystems provided scores for the Blade 1000. The SGI Onyx2 IR was tested in the authors' laboratory. Scores are rounded to the nearest integer.

Each viewset is run with a number of options, for example, to turn on ambient light, enable antialiasing, enable the Z-buffering, etc. The compound score represents a geometric mean of component tests, as given in the following equation:

$$\text{viewset geometric mean} = (\text{test}_1 \text{ Weight}_1) \times (\text{test}_2 \text{ Weight}_2) \times \dots \times (\text{test}_n \text{ weight}_n) \quad (4.3)$$

where weight_i is the weight associated with test_i of that viewset.

Weights have values between 0% and 100% and capture the frequency with which the operation associated with a particular test occurs in a given application. For example, the DRV test is weighted 75% for walkthrough rendering of surfaces, 13% when the model is textured, 4% for transparency, 4% for wireframe, depth-buffered rendering, and 4% for viewer position/orientation. Table 4.2 is a benchmark comparison of the Fire GL2 versus the Quadro2 Pro (Gloria III) and versus other architectures discussed in this chapter.

4.3 WORKSTATION-BASED ARCHITECTURES

The second largest computing base after the PC in deployed units is the workstation. Its advantage over the PC is greater computing power due to the ability to use superscalar (multiprocessor) architectures, larger disk space, and faster communication modalities. Thus one would expect major workstation manufacturers, such as Sun Microsystems, Hewlett-Packard, and Silicon Graphics Inc. (SGI), to develop some kind of VR capability. The primary market for workstations is still general computing rather than VR. Therefore manufacturers such as Sun Microsystems preferred to retrofit existing models with high-end graphics accelerators and adapt OpenGL to their Unix operating system. Since Unix is by design a multitasking environment, workstations are well suited to the real-time nature of VR. For example, one CPU can run user I/O and networking simultaneously with the application stage of the graphics pipeline running on another CPU. SGI, which created the OpenGL standard and has a much longer tradition in real-time graphics, went a step further. It created highly parallel graphics

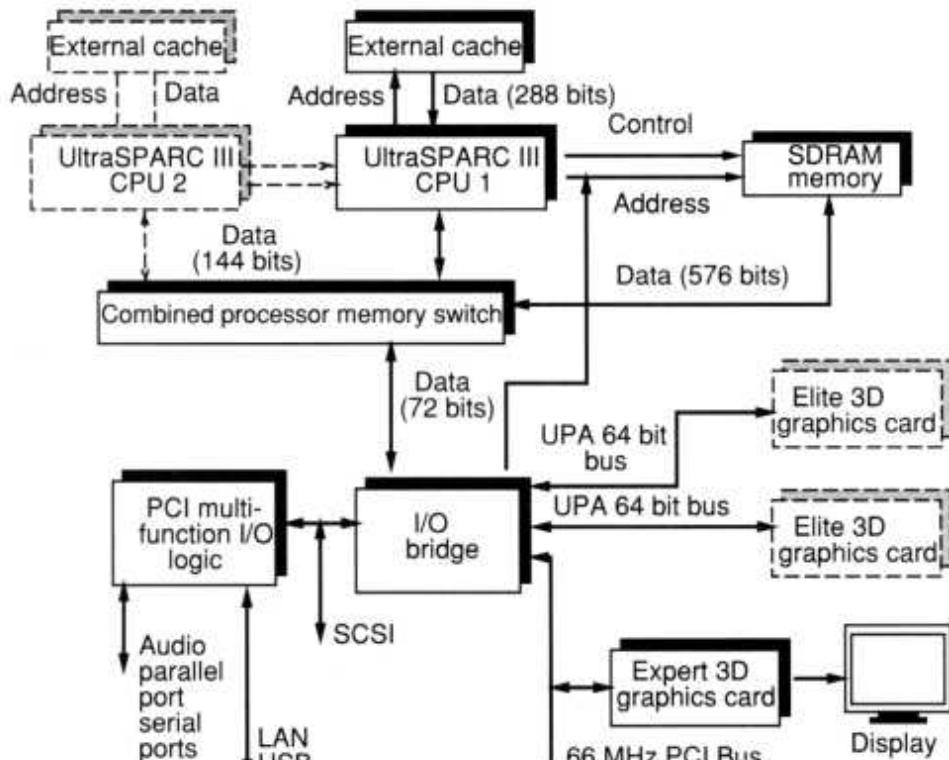
pipeline architectures to supplement parallelism at the CPU and Unix operating system levels.

4.3.1 The Sun Blade 1000 Architecture

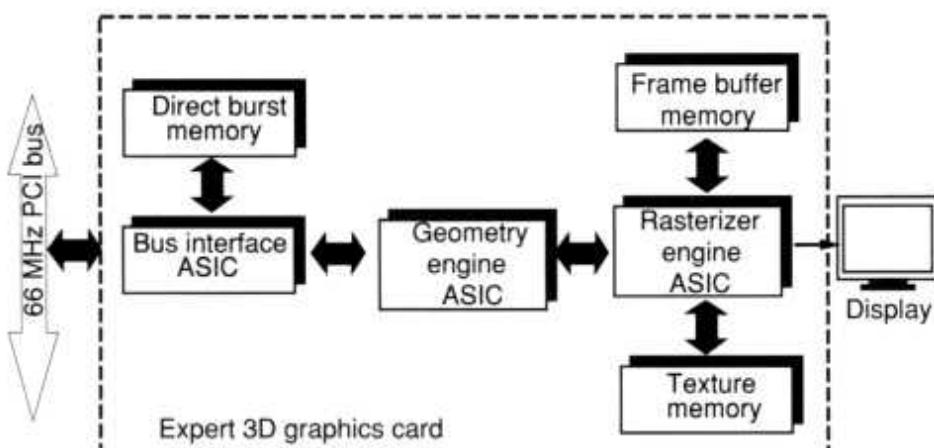
In 1992 Sun Microsystems introduced a virtual holographic workstation [Sun Microsystems, 1993]. This was a Sparcstation II with a GT graphics accelerator, rendering only 100,000 polygons/sec. This system has evolved to the current Sun Blade 1000 workstation [Sun Microsystems, 2000] with Expert3D graphics rendering 6 million triangles/sec. As illustrated in Figure 4.16a, the Blade 1000 has two 900-MHz UltraSpark III 64-bit processors, each of which is twice as fast as the previous-generation UltraSpark II CPUs. Their performance and low latency are helped by the presence of primary and secondary caches, which reduce the need to access the slower main memory. The data bus is wide (576 bits) and fully independent of the control/address bus. This is realized with the aid of a novel six-processor combined processor memory switch through which the data are routed. This design allows multiple memory calls to be serviced simultaneously and out of order, such that one operation need not wait for another to be completed.

The graphics pipeline of the Blade 1000 has some interesting features, representing a transition from the traditional Sun Ultra Port Architecture (UPA) graphics bus to the more standard PCI bus. There still are two 64-bit UPA busses, which are routed to older Elite 3D graphics cards. Using a separate bus for each card allows faster throughput than would be possible if a single bus were shared. Alternately the workstation can be retrofitted with Sun's high-end Expert3D graphics card shown in Figure 4.16b. This card communicates with the PCI bus through a bus interface application-specific integrated circuit (ASIC), which uses an 8-MB on-board direct burst memory. The 3D transformations, shading (up to 24 light sources), and geometry clipping operations are done by a geometry engine ASIC. This chip feeds the results into a rasterizer engine ASIC for the last stage of the pipeline. Pixel operations are accelerated by a 64-MB texture memory and stored in a 64-MB frame buffer at a fill rate of 143 million pixels/sec. This arrangement is superior to the single unified memory approach, in that texture memory can be accessed at the same time the frame buffer memory is

written. The resulting image has high resolution, namely 1920 x 1200 pixels in monoscopic mode and 1280 x 1024 pixels in stereoscopic mode (at 112-Hz screen refresh rates). The digital signal is converted to analog and output to RGB monitors, while a sync signal is provided for active glasses. Unfortunately, viewperf benchmark scores are lower for this card than for the PC graphics cards previously discussed, as seen in Table 4.2. This may be due to the Expert3D's (much) lower pixel fill rate, the slower geometry engine, and the slower CPU on the Sun Blade host. Another factor is, of course, the much slower PCI bus compared to the AGP busses currently used by PC graphics architectures.



(a)



(b)

Fig. 4.16 The Sun Blade 1000 workstation: (a) system architecture; (b) Expert3D graphics card detail. Adapted from Sun Microsystems [2000]. Reprinted by permission.

4.3.2 The SGI Infinite Reality Architecture

Prior to the current advances in PC graphics hardware, SGI dominated the VR market. In 1992 the company introduced the Reality Engine graphics architecture, which could render 240,000 Gouraud-shaded, textured polygons/sec [SGI, 1992]. This evolved into the highly parallel Infinite Reality graphics system illustrated in Figure 4.17 [Montrym et al., 1997], which can render 860,000 shaded and textured triangles/sec [Humphreys and Hanrahan, 1999]. Communication with the main system bus is done by a host interface processor (HIP) ASIC. Its cache holds the display list component of the scene database in order to minimize buss communication and speed up computations. Display lists are generated during scene traversal (discussed in Chapter 5) in order to optimize rendering speed. Alternatively, display lists can also be stored in system RAM, from where they can be fetched by the HIP in a fast direct memory access (DMA).

The data loaded in the graphics pipeline by the HIP are parceled by a geometry distributor ASIC to one of four geometry engines. This multiple-instructions, multipledata (MIMD) arrangement assures task parallelism by routing data to the least-busy geometry engine. Each geometry engine in turn consists of three floating-point cores (one each for the X, Y, and Z vertex coordinates). Thus vertex geometry stage computations (transforms, lighting, clipping) are also done in parallel in a single-instruction, multiple-data arrangement (SIMD). A geometry-raster FIFO buffer then reassembles the instructions in the correct order and places the data on a vertex bus. This assures communication with four raster memory boards. Each board has its own memory and fragment generator, such that the image is parceled and rendered in parallel. Finally, the pixels are sent to the display hardware, which performs digital-to-analog conversion prior to display on one high-resolution or up to four lower resolution displays. Although the Infinite Reality architecture is highly scalable and offers significant parallelism, it cannot compete with newer PC graphics accelerators in terms of either performance or cost. As illustrated in Table 4.2, an Onyx2 R10,000 CPU (195 MHz) with Infinite Reality graphics acceleration scored lower than all other cards presented here. These tests were done in the authors' laboratory and are consistent with results posted online [Vuurweek, 2002]. The importance of utilizing display list caching on the Infinite Reality architecture is illustrated in Figure 4.18 [Humphreys and Hanrahan, 1999].

This graph shows results of tests done at Stanford University in conjunction with research on tiled displays. The test consisted of rendering layers of 100 independent triangles each with and without display list draw caching. Thus the scene complexity depended on the number of layers, varying from 20,000 to 100,000 triangles. Each layer has its own associated display list, which can be preloaded in the HIP 16-MB cache. As discussed previously, when the scene complexity grows, the scene refresh rate drops and the time spent to render a given frame grows. This is true whether display lists are cached or not. However, the rate at which the frame rendering time grows with scene complexity depends on display list caching, which results in a sixfold faster rendering compared to noncached display lists. At 100,000 triangles spread over 1000 layers, the Infinite Reality spent 2 sec for each frame, which is entirely too long for VR purposes. However, when display list draw caching was implemented, the frame rendering time was 0.36 sec, or approximately three frames/sec.

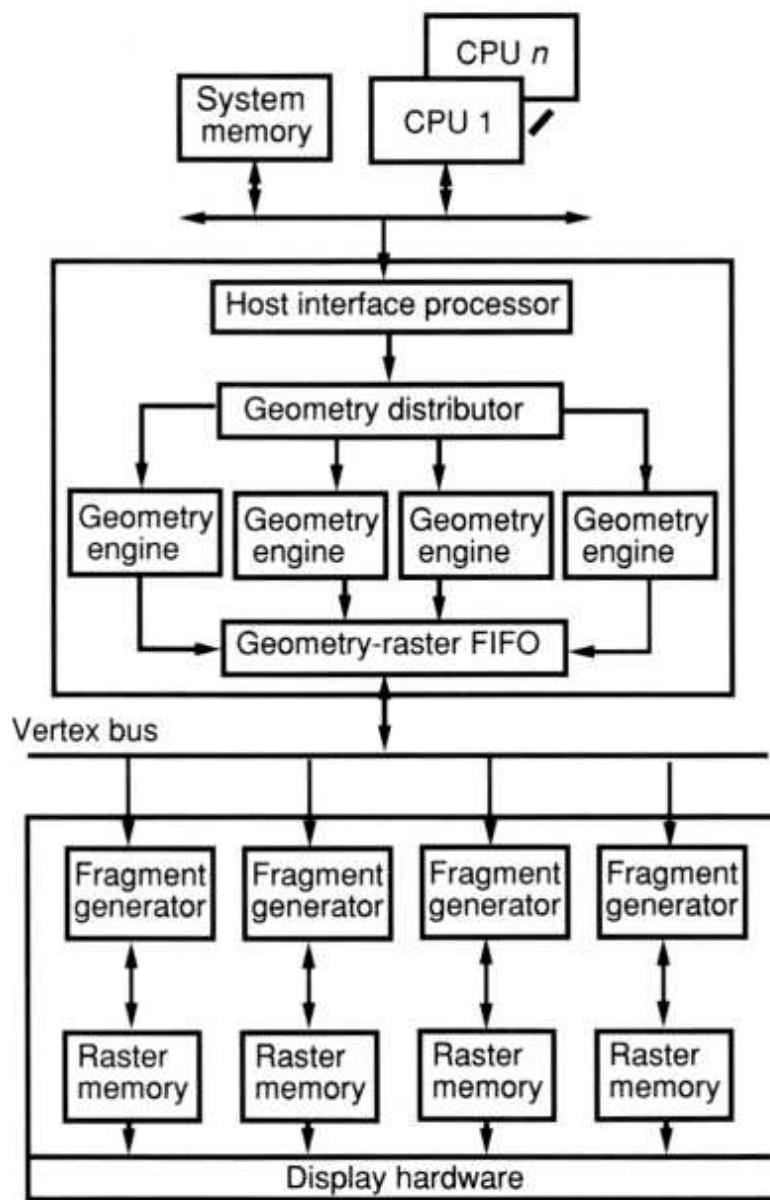


Fig. 4.17 The SGI Infinite Reality architecture. Adapted from Montrym et al. [1997]. Reprinted by permission.

4.4 DISTRIBUTED VR ARCHITECTURES

One advantage workstations had over PCs until recently was the Unix operating system support for multiple displays (fed by multiple graphics cards). Although the Gloria III (Quadro2 Pro) and the Fire GL2 have two graphics output ports (analog and digital), only one can be used at a given

time. Furthermore, until the introduction of Windows 2000, the PC operating system did not support two graphics cards working simultaneously on the motherboard. Thus a multimonitor display of the type shown in Figure 4.19 would necessitate the use of five PCs, each feeding one monitor [3Dlabs Inc., 2001a]. Such multiview displays are becoming more frequent today for single users running applications ranging from military command and control, to mechanical design, to oil exploration. Large-volume displays, such as the CAVE or the tiled display walls discussed in Chapter 3, also require multiple views to be rendered by the VR engine. In order to do so, the VR engine needs to have several graphics pipelines working cooperatively to provide the visual feedback needed by the simulation.

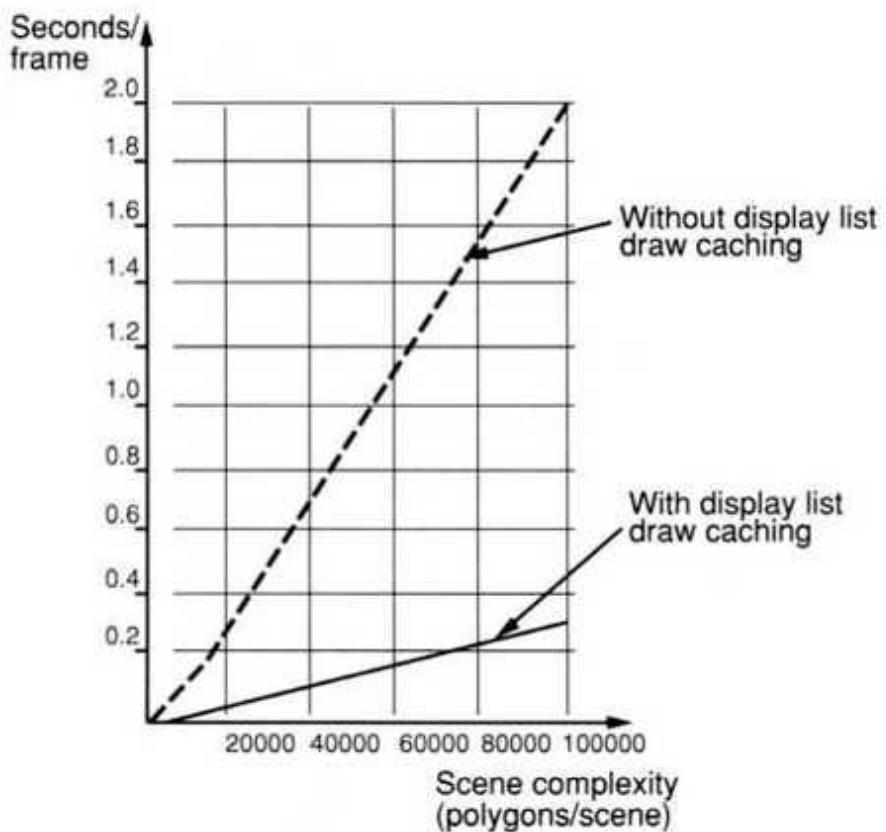


Fig. 4.18 Influence of display list caching on Infinite Reality rendering performance. Adapted from Humphreys and Hanrahan [1999]. ©1999 IEEE. Reprinted by permission.

Definition A distributed VR engine is one that uses two or more rendering pipelines. Such pipelines can perform graphics or haptics computations, and

be located on a single computer, on several colocated computers, or on multiple remote computers integrated in a single simulation.

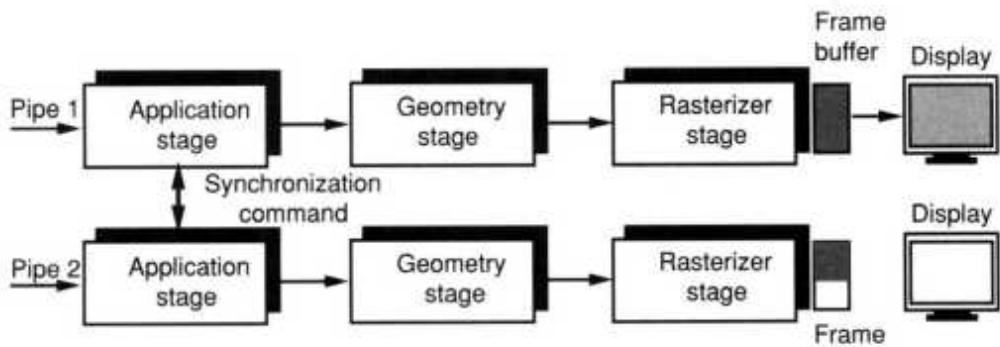
4.4.1 Multipipeline Synchronization

Whether multiple graphics pipelines are part of a single computer or of multiple cooperating ones, the output images need to be synchronized. This is especially true for tiled displays, where lack of synchronization can lead to disturbing visual artifacts of the type shown in Figure 3.14a. If the side-by-side displays are CRTs (as is the case of the system shown in Figure 4.19), lack of synchronization of the vertical sweep will result in flickering due to mutually induced magnetic fields. Synchronization is also important in order to reduce overall system latencies, create consistent frame refresh rates, and thus reduce the effects of simulation sickness (discussed in Chapter 7). Therefore, before describing distributed VR architectures, it is necessary to take a closer look at the multichannel synchronization of their graphics feedback.

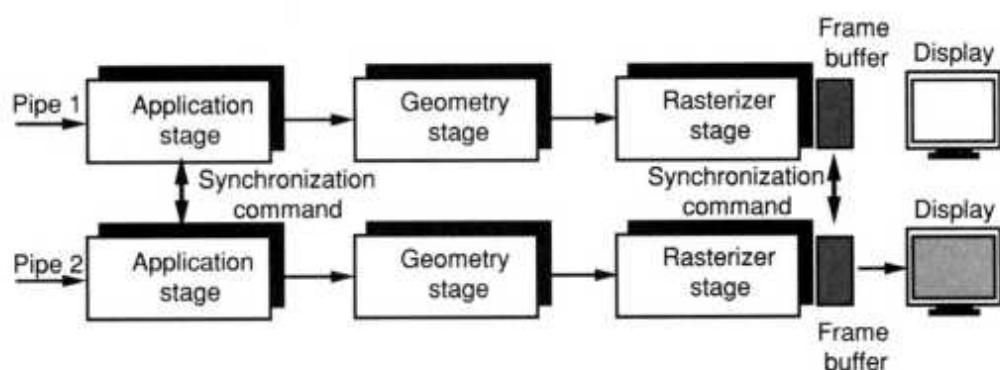


Fig. 4.19 A multimonitor VR application. From 3Dlabs Inc. [2001a]. © 3Dlabs Inc. Reprinted by permission.

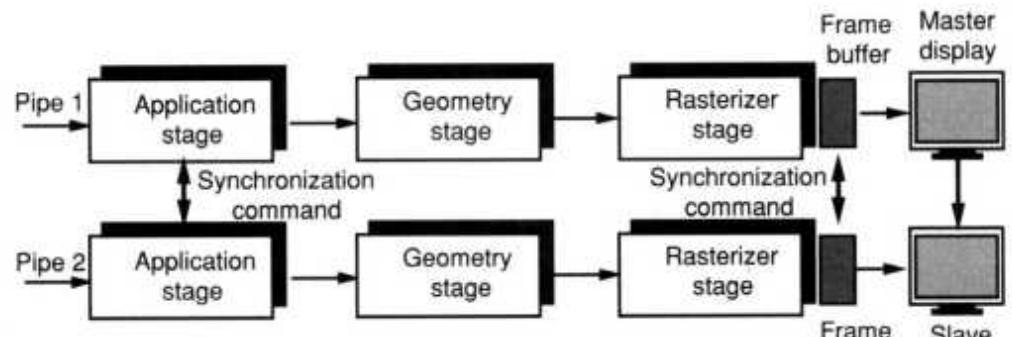
There are several approaches to multipipeline synchronization [Quantum3D, 2001], as illustrated in Figure 4.20. One approach, called software synchronization, asks the application stages on the parallel pipelines to start processing a new frame at the same time. This approach is not adequate, however, since it does not take into account the possible lack of symmetry in the loads processed by each pipeline. If pipeline 1 in Figure 4.20a has the lighter load (fewer polygons, less overlap, fewer textures, etc.), then it will finish ahead of pipeline 2, and its frame buffer will be filled and then displayed (or swapped) by the CRT.



(a)



(b)



(c)

Fig. 4.20 Multipipeline synchronization approaches: (a) software; (b) software and frame buffer swap; (c) software, frame buffer, and video.

Another approach, illustrated in Figure 4.20b, is to do both software and frame buffer swap synchronization of the two pipelines. In this case pipeline 1 (which finished first because of its lighter load) will wait for pipeline 2 to

fill its frame buffer, and then the system will issue a swap command to both buffers. Note that a buffer swap command does not guarantee an actual swap, since this can take place only during the vertical blank period of the display. This is the time interval when the electronic gun of the CRT is moved back from the bottom of the screen to the top of the screen in order to trace a new frame. Since the two displays are not synchronized in this approach, one or both buffers will have to wait for their respective display to allow the swap. In the worst case, one image (for example, the output of pipeline 1) will end up being 1/72 sec out of sync with the other one.

The third and optimal approach is to add video synchronization of the two (or more) monitors, as illustrated in Figure 4.20c. Video synchronization means that one of the displays becomes the master while the other is the slave. Interconnection between the graphics cards of the master and that of the slave(s) assures that the vertical and horizontal line retracing on the slaves follows the master. This is done by synchronizing the slave's internal video logic circuitry to that of the master.

Recall that in Chapter 2 we indicated that DC magnetic trackers benefit from synchronization with the CRT monitors and the building electrical mains in order to reduce interference and improve accuracy. This means that an external synchronization signal needs to be provided to both the tracker interface and the monitors. Such an external signal, called genlock, is also useful when graphics rendering is done in the field-sequential stereo mode. As discussed in Chapter 3, this entails alternating left-eye and right-eye images on a single monitor and wearing active glasses synchronized with the monitor frames. If the stereo image is tiled, then all side-by-side CRTs need to display the same eye's image before they all switch to the other eye's image. Otherwise the resulting flickering and blurring will induce simulation sickness. As detailed in Figure 4.21 [3Dlabs Inc., 2001a], an external genlock signal cable is plugged into the graphics card of the master PC. This PC is then wired to the slave 1 PC using a multiview cable and so on, in a daisy chain arrangement. Buffer swapping synchronization is done through handshaking over the multiview cable. When slave 2 has its frame buffer filled and is ready to swap the frame, it raises a "done" signal on the multiview cable connecting it to slave 1. If slave 1 had its frame buffer full

and ready for swapping, it then immediately raises its "done" signal. If not, it finishes the rendering and then notifies the master PC. Assuming the master PC had its frame buffer filled already, it raises the "release" signal to slave 1 and slave 2. In this way all three PCs swap at the same time. Swapping synchronization is possible because the genlock signal resets the internal video circuitry of all three PCs.

4.4.1.1 Graphics and Haptics Pipelines Synchronization. Recall that the meaning of rendering is extended in this book to include both the visual and haptic sensorial modalities. As such we need to look at ways to synchronize the two heterogeneous pipelines, as illustrated in Figure 4.22a. Synchronization in this case is done at the application level, where the CPU needs to do both database traversal (for the graphics pipeline) and collision detection for the haptics rendering pipeline. Two approaches exist: (1) force computation is done on the host computer (usually by a second CPU) or, (2) forces are computed by the processor embedded in the haptic interface controller. In the first method, only force vector data are transmitted to the interface controller processor, while in the second method collision detection information is provided to the interface controller. The force-rendering phase and mechanical texturing are always hardware-accelerated, which means they are done by the dedicated interface controller.

Decoupling the graphics and haptics rendering pipelines is necessary since they have significantly different output rates. The PHANToM controller, for example, performs force rendering at 1000 Hz, while the PC graphics pipeline rendering the corresponding images may have an throughput of 30-40 fps. More details are given in Chapter 6. Another example, shown in Figure 4.22b, is an experimental Rutgers Ankle Interface communicating with a PC host over an RS232 line [Girone et al., 2001]. The force rendering is done by an embedded Pentium on the Rutgers Ankle controller (shown on the right of the figure) at a rate of 300 Hz. The graphics component is rendered by a Fire GL2 card on a dual-processor 450-MHz Pentium II PC at a rate of 32 fps. Synchronization is done through message passing over a 38,400-bps RS232 line, which transmits approximately 150 new force targets every second.

4.4.2 Colocated Rendering Pipelines

A system with colocated rendering pipelines may consist of a single computer with a multipipeline graphics accelerator, side-by-side computers (each assigned a different rendering pipeline), or a combination of these alternatives. Such systems aim to balance cost and performance (or simulation quality) and will be discussed next.

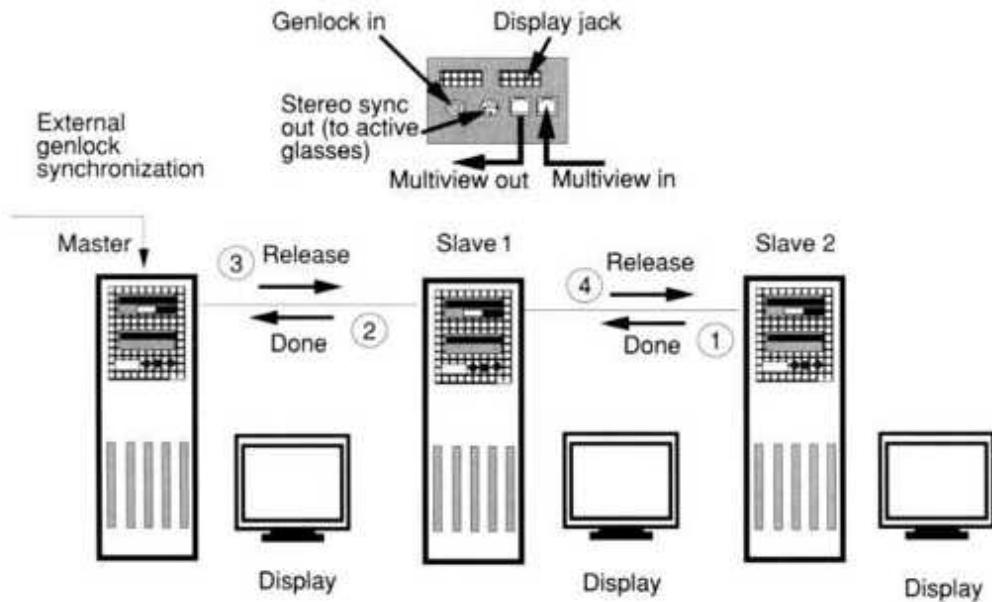
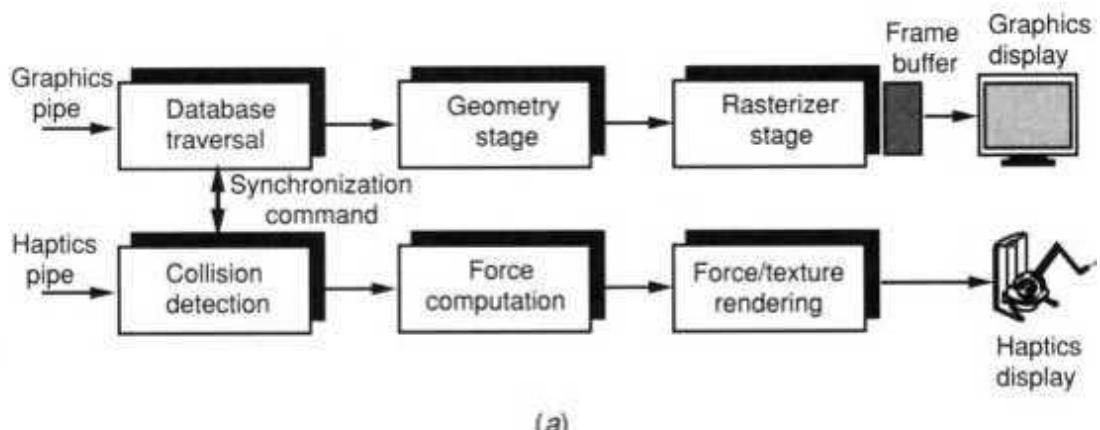


Fig. 4.21 Genlock and multiview synchronization. Adapted from 3Dlabs Inc. [2001a]. © 2001 3Dlabs Inc. Reprinted by permission.



(a)



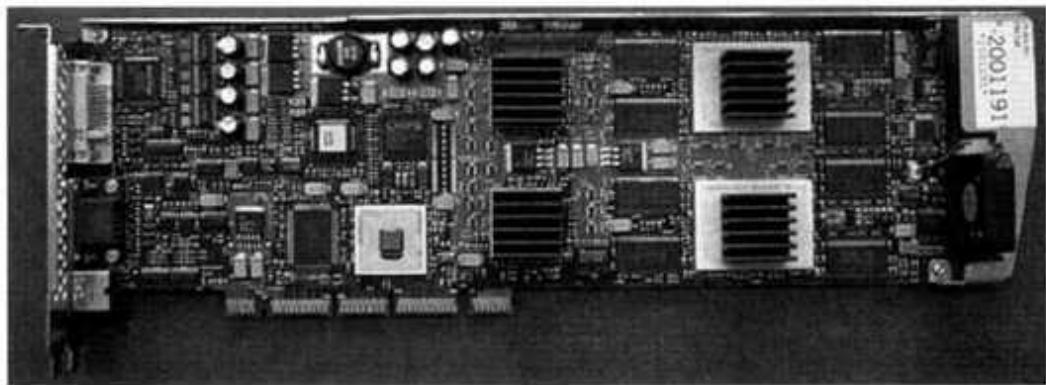
(b)

Fig. 4.22 Graphics-haptics pipeline synchronization: (a) block diagram; (b) the Rutgers Ankle prototype. © Rutgers University. Reprinted by permission.

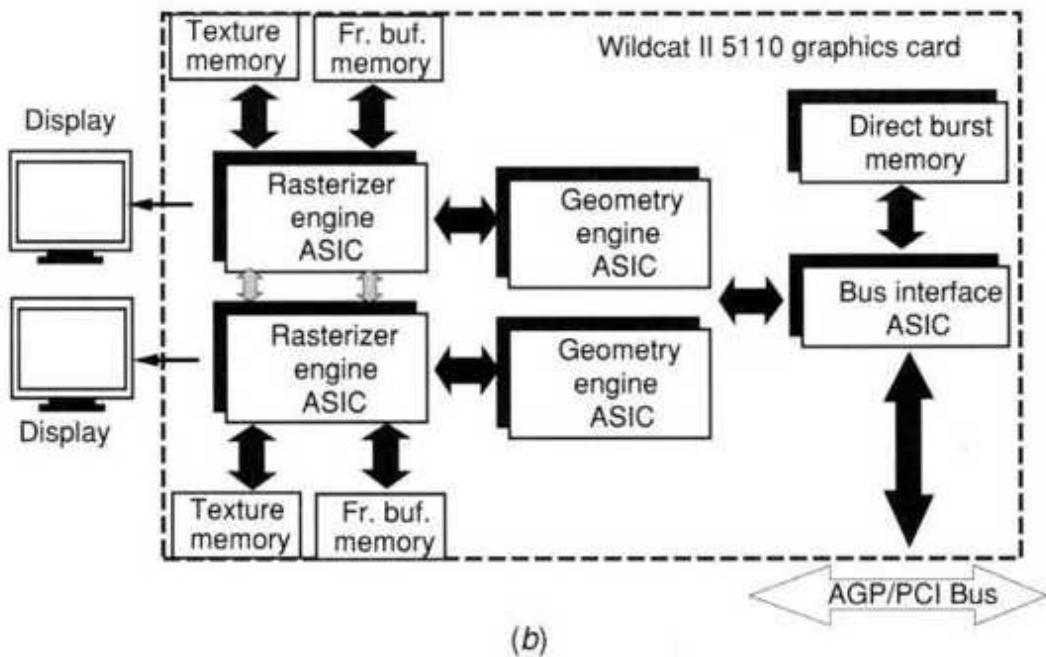
4.4.2.1 Multipipe Graphics Cards. These provide a less expensive alternative to the master-slave PC system illustrated in Figure 4.21. One solution to reduce costs is to place three single-pipe graphics accelerators in

a PC. Each single-pipe card then shares the AGP/PCI bus bandwidth and the same CPU and main memory, and connects to its own monitor. An even less expensive possibility is to use a single multipipe graphics card, such as the Wildcat 115110 shown in Figure 4.23a [3Dlabs Inc., 200 lb]. Unlike the Fire GL2 or the Gloria III (Quadro2 Pro) cards, the Wildcat 115110 can display on two CRTs simultaneously and independently. This is possible owing to the presence of two geometry engine ASICs and two rasterizer ASICs. As shown in Figure 4.23b, the geometry and rasterizer ASICs form two independent graphics pipelines. Each pipeline has its own texture memory and its own frame buffer, similar to the architecture used in the Sun Expert3D card previously discussed. Another similarity to the Sun architecture is the use of a bus interface ASIC feeding the two pipelines as well as the resident direct burst memory. One of the card's outputs is digital and one is RGB (analog). The dual-monitor option is supported only for analog displays. Thus an adaptor is necessary in order to convert the digital port output to analog RGB prior to display. The Wildcat 115110 thus allows the easy synchronization of two side-byside displays without the genlock and multiview wiring previously discussed. If three or four monitors are needed by the application, then a quad-pipe Wildcat accelerator can be used, otherwise external genlock/multiview cabling is necessary.

An interesting feature of the Wildcat graphics accelerator is its "parascale" (parallel and scalable) architecture. This allows the card to be configured for single monitor output, such that the two graphics pipelines split the load associated with rendering a given frame rather than working on separate images. This arrangement results in very high throughput and very high viewperf scores. The throughput grows linearly with the number of pipelines. For example, whereas a single graphics pipeline can render 5 million 25-pixel, smooth shaded triangles/sec, this more than doubles (12 million such triangles/sec) when the scene is split on two graphics pipelines. Indeed, Table 4.2 shows the Wildcat 115110 to be the fastest of all the PC- and workstation-based systems described here.



(a)



(b)

Fig. 4.23 The Wildcat II 5110 board: (a) outside appearance. From Scheffel [2001b]. © Tom's Guides Publishing, AG. Reprinted by permission. (b) dual-pipeline architecture. Fr. Buf., Frame buffer. From 3Dlabs Inc. [2001b]. © 3Dlabs Inc. Reprinted by permission.

4.4.2.2 PC Clusters. Large tiled displays, composed of tens of projectors, need an equal number of graphics pipelines to drive them if high resolution is desired. Using a single PC to render graphics for such a large tiled display is impractical owing to several factors. First, the PC has a limited number of AGP slots available, which prevents the installation of more than a few graphics cards. Indeed some cards, such as the Wildcat 115 110,

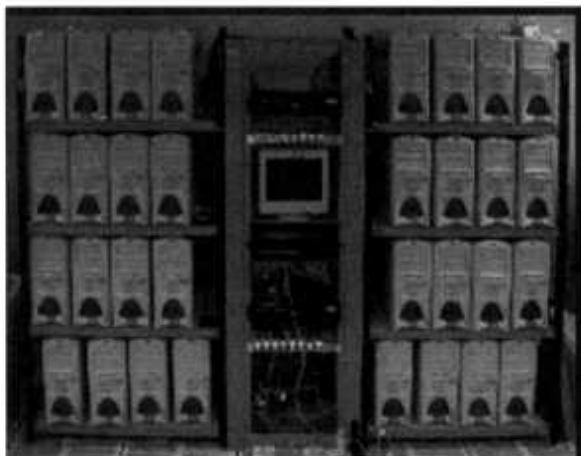
occupy two slots, due to their large cooling structure, making the problem even worse. Finally, even if sufficient slots were available on the motherboard, the more cards time-share a given AGP bus, the worse is the throughput, due to bus congestion. There are two architectural solutions to the rendering needs of large tiled displays. One solution is to rely on large, multipipeline workstations, such as the SGI Onyx 2/Infinite Reality, and allocate each pipeline to one or to several displays. This is the approach taken in systems like the PanoWall (Fig. 3.18), which rely on special hardware to control image blending. The drawbacks in this approach are high cost and reduced image resolution (since the fixed-size frame buffer of an SGI Infinite Reality graphics pipeline is divided into smaller display tiles).

A better alternative to the use of a single high-end graphics workstation is to use a cluster of PC rendering servers, each driving a single projector. In this way the display resolution is maintained (each composite tile being rendered at full frame buffer resolution), and the system cost is reduced by at least an order of magnitude compared to the old approach. The reduction in system cost comes from the use of low-cost consumer PCs (and graphics cards) each having processing capabilities equal to (or exceeding) that of a single pipeline high-end graphics workstation. Further reduction in cost is due to the cluster ability to perform image tiling in software without reliance on expensive dedicated hardware.

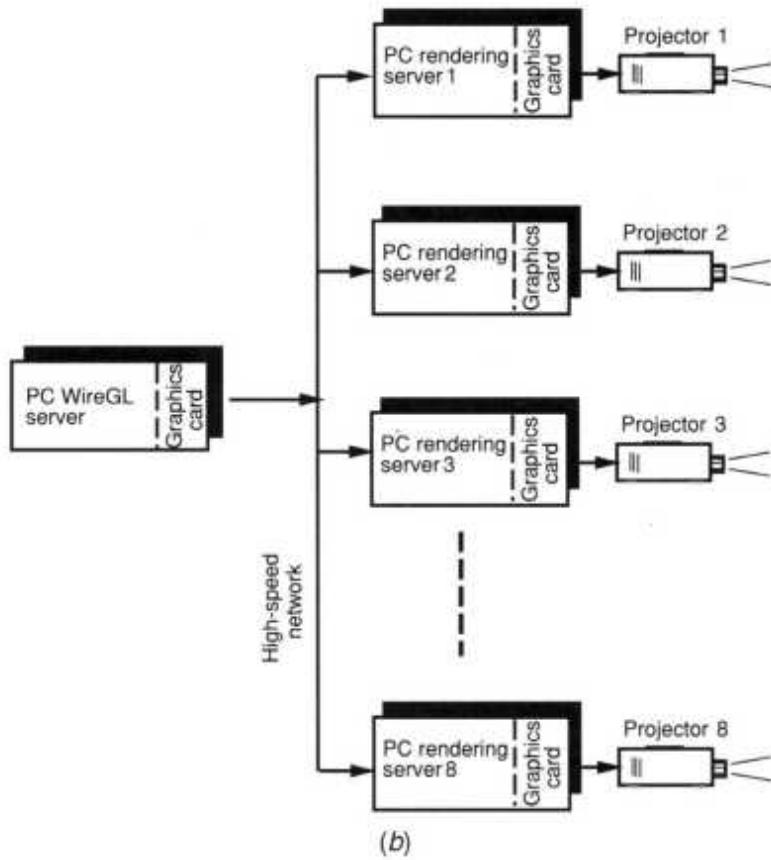
A PC cluster would have to be networked on a high-speed LAN to allow synchronization of the output graphics by a control server. The control server takes the virtual scene geometry and parcels it for the cluster in order to form a single logical screen. The fixed LAN throughput becomes the main limiting factor in the size of the cluster it can accommodate. The more PCs are added, the slower is the composite display refresh rate. It then becomes important to minimize LAN traffic, and the associated synchronization overhead, in order to reduce the impact scalability has on cluster graphics frame refresh rates.

An example of a PC cluster is the Chromium system depicted in Figure 4.24a [Hanrahan, 2001]. It consists of 32 rendering servers and four control

servers, each an 800-MHz Pentium III with an NVIDIA GeForce 2 GTS graphics card. The PC cluster is interconnected over a 64-bit, 66-MHz LAN Myrinet, which has a point-to-point observed bandwidth of 100 MB/sec. Each control server oversees eight rendering servers, as illustrated in Figure 4.24b [Humphreys et al., 2000]. These output 1024 x 768-resolution image tiles, which are then projected on the composite display screen.



(a)



(b)

Fig. 4.24 The Chromium PC cluster for tiled visual displays. (a) outside appearance. From Hanrahan [2001]. Photo courtesy of Prof. P. Hanrahan. (b) Networking architecture. Adapted from Humphreys et al. [2000]. © 2000 IEEE. Reprinted by permission.

Key to the scalability of the Chromium cluster is a WireGL software driver, which runs on each control server. It communicates with the rendering servers based on a TCP/IP protocol and shared memory. The WireGL intercepts OpenGL commands issued by the VR simulation and parcels them to the corresponding rendering PC while keeping track of the state of each node graphics. In order to minimize LAN traffic WireGL takes a two-pronged approach. First, OpenGL commands issued by the VR simulation application are encoded before being sent over to the servers. Eight bits is sufficient to encode the set of possible OpenGL commands, which are then decoded at the receiving PC based on a look-up table. Furthermore, commands and vertex data are sent together in a combined packet with a known structure. This is done by packing the OpenGL-encoded commands and data backward in the network buffer. Thus when the packet is decoded, the PC first gets the operation, then the data to which it applies, and knows how to interpret those data. This alleviates the need for costly separate command and data communications.

The second way to reduce LAN traffic is to send data only to the PC that needs it. WireGL, upon receiving geometry data from the VR simulation application, sorts it so that each server gets the geometry that overlaps its portion of the output display. Since each rendering server has its own outgoing network buffer, the more PCs in a cluster, the more buffers there are to send, and the larger is the LAN traffic. This is true if all buffers were sent, or broadcast, every time. Instead, WireGL keeps track of the state of each PC rendering server. Buffers are then transmitted only if there is a difference between the rendering server state and the network buffer data. The effects on scalability are significant, as illustrated in Figure 4.25 [Humphreys et al., 2000]. The graph illustrates the relative frame refresh rate for a scene consisting of 183,600 triangles being rendered by the cluster. The reference frame refresh rate was obtained when only one projector and one PC rendering server were used. In this situation the overhead associated with WireGL state checking results in a 5% penalty compared to the broadcast mode (no state checking). However, transmitting in broadcast mode results in a significant reduction in the frame refresh rate as soon as more projectors are added. Thus, for a 32-projector array (8 x 4), there is a 95% degradation in throughput due to LAN bottleneck. By

contrast, WireGL state checking assures that the 32-projector array still has 70% of the frame refresh rate of a single projector connected over the same LAN.

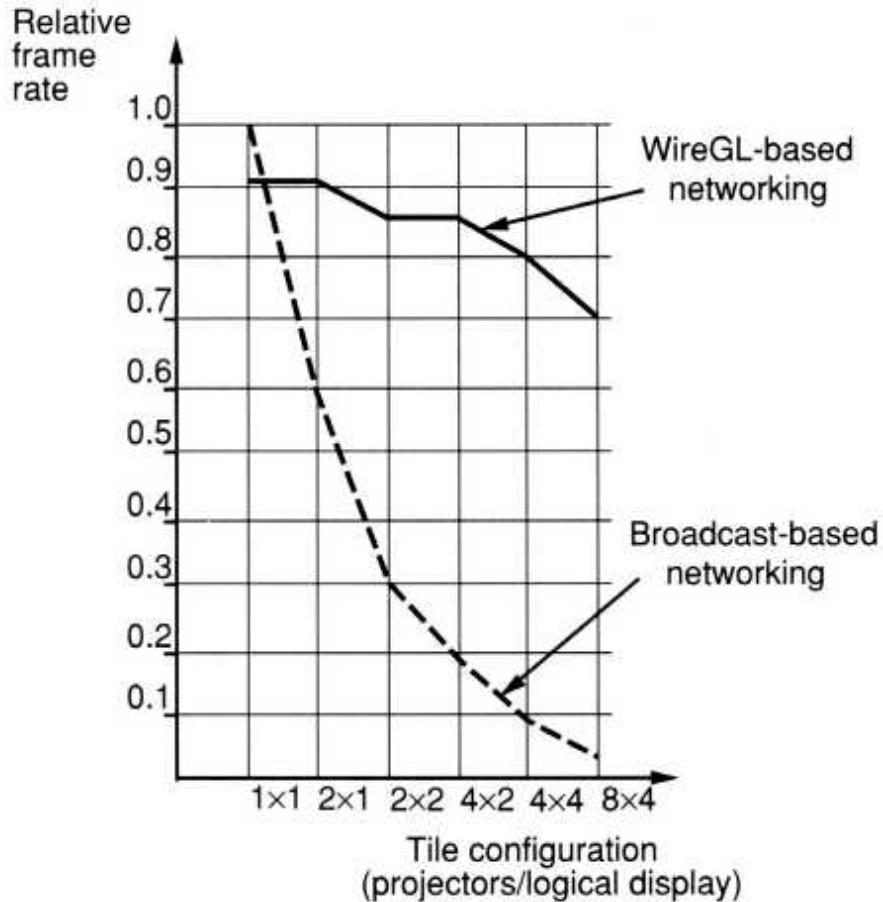


Fig. 4.25 Frame refresh rate comparison between WireGL and broadcast communication as a function of the number of cluster PCs. Adapted from Humphreys et al. [2000]. © 2000 IEEE. Reprinted by permission.

In the Chromium PC cluster example, the rendering and display were done in parallel. An alternate use of PC clusters is the parallel rendering of an image for a single display [Rudrajit et al., 2000]. In this approach a PC client sorts the virtual scene into patches that are then assigned to one of many PC servers for rendering. The output from the PC server cluster is then sent to a display PC frame buffer, where the tiles are merged into a single image prior to display. Researchers at Princeton University tested this sort-first rendering approach on a cluster of 18 Pentium III PCs connected over a

Mytrinet LAN. Despite the fact that each PC server had very low end graphics cards (\$250 Hercules 3D Prophet), there was an eightfold speedup in rendering for a scene of more than 355,000 polygons. The researchers found that the scalability of the system was limited by the computational overhead. This was associated with the image division into tiles (on the client PC), the tile overlap computations (on the server cluster), and the display PC D/A converter bandwidth (above 25 frames/sec). The amount of overhead grew linearly with the number of PCs added to the cluster.

4.4.3 Distributed Virtual Environments

The architectures presented so far distribute the load among multiple graphics cards of a single machine or divide the rendering task among several PC servers communicating over a LAN. Network distribution has the additional advantage of facilitating remote computer access and participation in the simulation. This in turn makes possible multiuser VR simulations where two or more remote users share a distributed virtual environment.

Definition A virtual environment is said to be distributed if it resides on two (or more) networked computers, which share the total simulation computational load.

The networked computers, called nodes, allow multiple remote users to interact with each other and with the objects in the shared virtual world [Williams, 1998]. The users' interaction within a distributed virtual world may be collaborative or cooperative."

Definition Two (or more) users are said to collaborate in a distributed virtual environment if they take turns performing a given simulation task such that only one user interacts with a given virtual object at a time. Conversely, users cooperate in a simulation if they simultaneously interact with a given virtual object.

The type of network that supports distributed virtual environments is determined by a number of factors. These include the user's actions, the multimodality of the interaction (graphics, voice, haptics), as well as the

number of simultaneous users in a given shared virtual environment. Other factors used in the design of networked virtual environments are simulation survivability in case of communication malfunctions, the acceptable level of latencies with which changes in the virtual world state arrive at remote nodes, communication reliability and scalability, etc.

TABLE 4.3. Characteristics of Networks Used by Distributed Virtual Environments

Type of Network Connection ^a	Bandwidth (bps)	Network Type ^b
Gigabit Ethernet	1 G	LAN
FDDI	100 M	LAN
Fast Ethernet	100 M	LAN
Ethernet	10 M	LAN
ATM	≥ 1 G	WAN
ADSL ^c	8 M/1 M	WAN
T1	1.5 M	WAN
ISDN	64 K	WAN
Modem	≤ 56 K	WAN

^aFDDI, Fiber-distributed data interface; ATM, asynchronous transfer mode. ADSL, asymmetric digital subscriber line; ISDN, integrated service digital network.

^bLAN, Local area network; WAN, wide area network.

^cThe numbers for ADSL refer to download/upload bandwidths.

In this chapter we are interested in the architecture aspects of networks, looking at their physical and logical connections. Physical connections refer to the wiring between computers, which constitutes the physical network, while logical connections refer to the way messages are passed over a network.

Two important aspects of virtual environment distribution are the network type and its bandwidth (in bits per second, or bps). Networks can be classified as LANs and wide-area networks (WANs). LANs generally use the 10-Mbps Ethernet (or its faster variants), whereas WANs rely on much slower T1 connections (1.5 Mbps), asymmetric digital subscriber lines (ADSLs) (8 Mbps/1Mbps), or modems (up to 56 Kbps). Table 4.3 summarizes the characteristics of networks used by distributed virtual environments.

4.4.3.1 Two-User Shared Virtual Environments. These are the simplest shared environments and were the first to be developed. As illustrated in Figure 4.26 [Pere et al., 1996], each user has a PC with graphics accelerator and possibly other interfaces connected to it (3D trackers, haptic interfaces, etc.). Communication between the users is done over a LAN, through simple unicast packets sent using a TCP/IP protocol. Unicast communication routes packets from the sender to a single recipient, and TCP/IP ensures reliability by guaranteeing that the message is delivered. In this basic shared virtual environment (VE) example, each PC has a copy of the virtual world. State coherence is maintained by simply transmitting any changes that occur locally, due to the user's actions, to the remote PC. In this way the two users have exact replicas of the virtual world running on their twin machines. Some form of buffering needs to exist to allow the local user to interact with the simulation even if the remote PC or the LAN breaks down. Once communication is reestablished the buffer can be read to recover the shared VE state coherence.

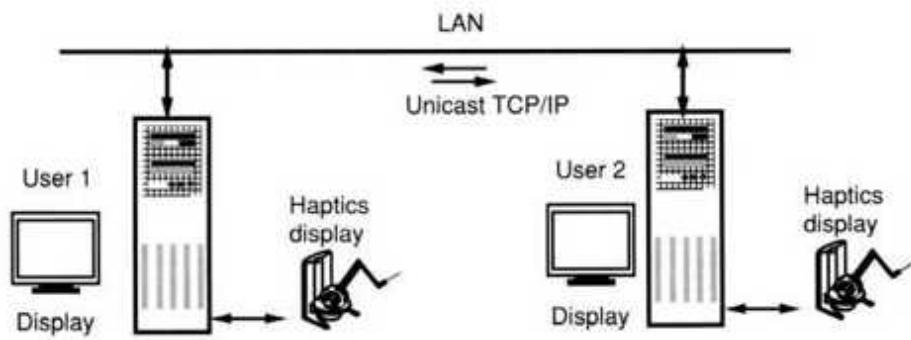


Fig. 4.26 Two-user shared virtual environment. Adapted from Pere et al. [1996]. @1996 ASME. Reprinted by permission.

4.4.3.2 Network Topologies for Multiuser-Shared Virtual Environments. These allow three or more participants to interact in a given virtual world. In this case the network architecture has to be able to handle dissimilar processing capacities and types of interfaces on the various remote computers. One approach is to have a central server to which the PC clients connect, as illustrated in Figure 4.27a [Singhal and Zyda, 1999]. In such a case the server workstation coordinates most of the simulation activities. It has the responsibility of maintaining the state of all of the virtual objects in

the simulation. Client processes manage the local simulations, interact with I/O devices, and perform graphics rendering on the local PCs. When a user acts on the shared VE, his or her actions are transmitted to the server as unicast packets, where they may be compressed and retransmitted to other users. The server performs such compression, as well as burst transmission smoothing, in order to reduce the network load. Furthermore, the server can decide to which user to relay the message it receives, based on the area of the virtual world each user interacts with. Clearly, a cooperative interaction should be relayed to all participating users if they are interacting in the same portion of the virtual world. However, when a user does a walkthrough in a virtual building, his or her actions are usually irrelevant to users outside that building. If the server has sufficient processing power, then the physical connection as well as the size and frequency of packet transmissions (for an average user) determine how many users can share in the simulation. For example, assume the simulation involves a battlefield populated by vehicles, aircraft, and soldier avatars, such as the Distributed Interactive Simulation (DIS) protocol, with packets of 144 bytes [Institute of Electrical and Electronics Engineers, 1995]. Singhal and Zyda [1999] estimated there could be between one and six users of DIS if the connection to the central server is done over slow dial-up modems. This number grows to between 39 and 163 users connected to the server through ADSL and to between 263 and 1085 users if cable modems are the physical connection. Although other applications will have different packet size and frequency requirements, the influence of the physical connection on the number of users interacting in a shared virtual environment is clear. In general, if a given message, say from user 1, needs to be transmitted to all the other users connected to the same server in unicast mode, then the network traffic grows with the square of the number of users N , or $O(N^2)$ [Capin et al., 1999]. Thus, unicast communication is bandwidthintensive, albeit more dependable than multicast communication discussed later.

At the high capacity of LANs or cable modems, it is very probable that the bottleneck preventing more users from participating will be the central server rather than network bandwidth. One solution to this problem is to replace the central server with several interconnected servers, as illustrated in Figure 4.27b. In this way each server maintains an identical copy of the

virtual world and is responsible for the communication needs of its clients. Messages are still sent in TCP/IP unicast mode, and may lead to increased latencies when interactions involve clients connected to different servers. For example, if client 1,1 interacts with client 2,n, then its message is first sent to server 1, then relayed to server 2, before finally arriving at client 2,n. Communication between servers needs therefore to be minimized in order to avoid network congestion and allow real-time cooperative simulations. This means that a minimum number of messages needs to be sent (for example, only changes in virtual object position or creation/deletion of virtual objects). Additional problems arise when servers are not matched in computational power. In that case, the faster server will send many messages (corresponding to VR state changes) and the slow server will spend a lot of time reading these messages instead of coordinating the local simulation. As a result, latencies in the slow server grow even more.

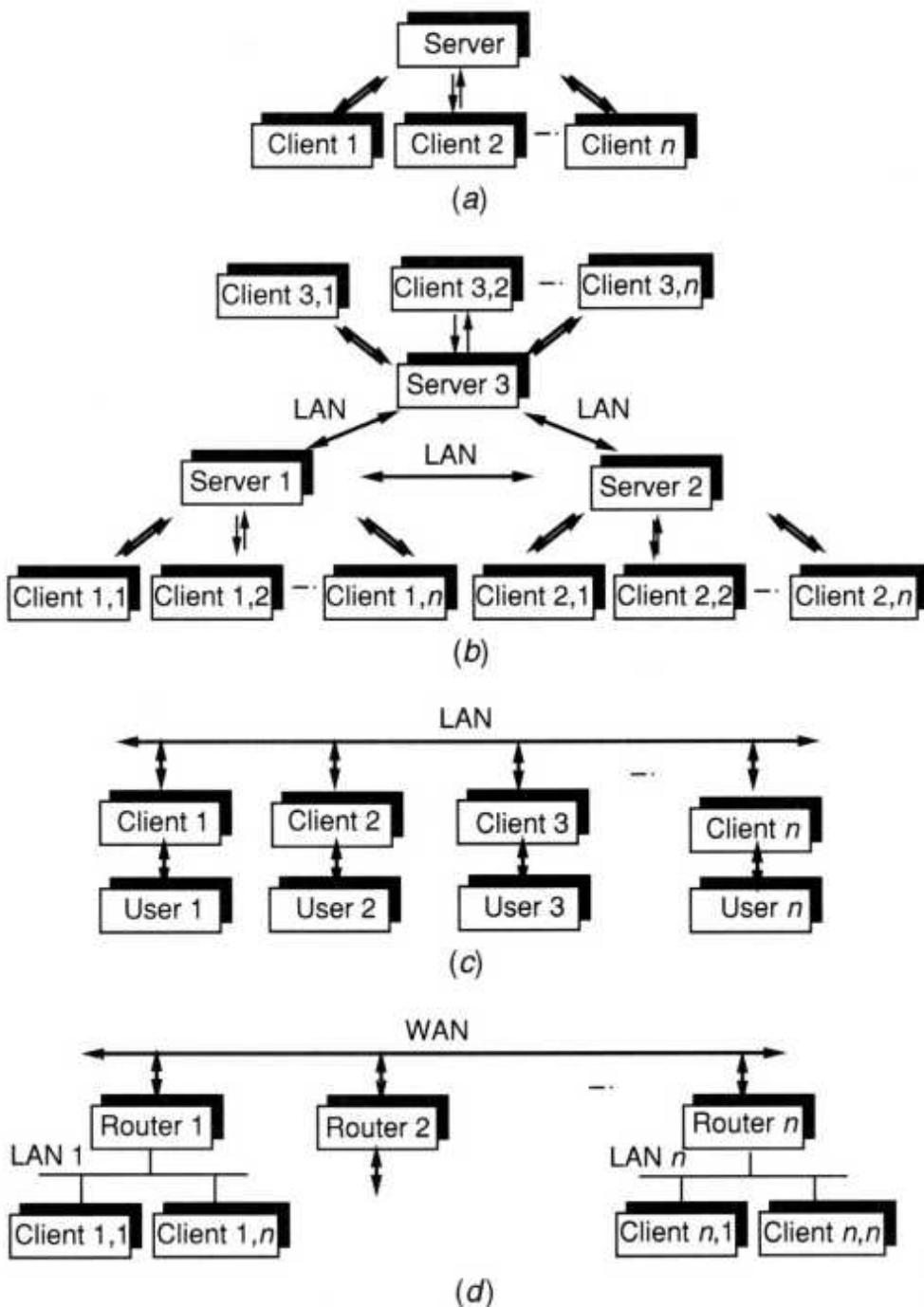


Fig. 4.27 Logical connections for networked multiuser virtual environments.
 (a) Single server. (b) Ring network with multiple servers. (c) Pier-to-pier LAN. Adapted from Singhal and Zyda [1999]. © Pearson Education Inc. Reprinted by permission. (d) Hybrid pier-to-pier WAN with bridging routers.

Adapted from Capin et al. [1999]. © John Wiley & Sons. Reprinted by permission.

A more recent approach to connecting a large number of users (clients) over a LAN is the pier-to-pier network illustrated in Figure 4.27c. Here the scalability of the networked VE is no longer limited by the server, which is eliminated. Its functions are distributed among a number of client peers communicating in a multicast UDP mode. In multicast communication, messages from one client are sent directly to any other client, without having to be relayed by the central server(s). If one user interacts with a portion of the virtual environment, then the resulting state change is sent over the LAN to a subset of peers. These peers serve other users for which the state change is relevant. All the peers on the LAN look at the message to determine if it pertains to them, using area of interest management (AOIM) software. This communication scheme works well in reducing the amount of message traffic to $O(N)$, and is faster due to the use of the UDP communication protocol.

Pier-to-pier multicast does have its drawbacks. First, if all users disconnect, there is nothing to maintain the networked virtual world. Thus a minimum of one client has to maintain the world, and new users joining the simulation have to first update their world state. This initial download is done by secure TCP/IP unicast, and may be lengthy. Furthermore, computer security becomes more of an issue because multicast networks are more vulnerable to computer viruses, as the "watch dog" server is eliminated. Finally, pier-to-pier does not work well over WANs, which may not support multicast communication everywhere.

Networks dedicated to multicast exist, for example, the Multicast Backbone, or MBone, developed at the Naval Postgraduate School, which allows audio and visual communication [Macedonia and Brutzman, 1994]. However, the MBone is not universally available, such that hybrid networks are necessary to allow pockets of multicast users to communicate with other such users over a WAN. In the example illustrated in Figure 4.27d, network routers, called proxy servers, are used to package multicast messages as unicast packets before transmitting them to other routers. Here the messages are unwrapped by the local proxy server and multicast to the local clients. Such a network, called the DiveBone, was successfully implemented

between Sweden and the United Kingdom [Greenhaigh et al., 2001] as part of the European COVEN project. It allowed 16 simultaneous remote users to interact in a single virtual world through standard Internet connections without the MBone. Furthermore, researchers modeled the network bandwidth required for N simultaneous users in multicast versus the same users having unicast-based communication over the hybrid network. For two or three users unicast was faster, as there was no overhead associated with packet wrapping/unwrapping prior to WAN transmission. However, as the number of users increased, the multicast traffic grew with $O(N)$, whereas unicast had a much larger bandwidth requirement of $O(N^2)$.

4.5 CONCLUSION

The real-time requirement of virtual reality simulations demands powerful graphics hardware and multiprocessor, or multicompiler, task distribution. This chapter analyzed the basic rendering pipeline for graphics and haptics and described various computing platforms that implement it. The discussion was dominated by PC-based systems, from single-pipe to multipipe and clusters, as they form the vast majority of today's VR engines. Of particular interest are architectures that support large-volume displays, or side-by-side tiled displays, where synchronization of the multiple graphics pipelines involved is crucial. Finally, this chapter discussed networked virtual environments that allow two or more remote users to interact with a shared virtual environment. Such distributed architectures are the foundation that allows multiuser VR applications to exist. The software tasks performed by the rendering pipeline are described in the next chapter, which focuses on VR modeling.

4.6 REVIEW QUESTIONS

1. What are the three stages of an OpenGL rendering pipe and how can their load be balanced?
2. What are the stages of haptics rendering?

3. For a given graphics board, what is the relationship between scene complexity and refresh rate? What happens with the refresh rate of a scene when a computer renders in stereo versus mono?
4. Why does a real computer fps output curve differ from the theoretical curve given in the text? Make a drawing and explain.
5. Describe the Fire GL2 graphics board. Make a drawing and explain.
6. Describe the Gloria III (Quadro2 Pro) graphics board. Make a drawing and explain.
7. Describe the xBox architecture.
8. Give an example of workstation-based 3D graphics architecture.
9. What are benchmarks and why are they necessary? Describe viewperf test scores.
10. Describe the graphics rendering modes you know. What are their characteristics?
11. Consider two computers that render 500,000 polygons/sec and 1 million polygons/sec, respectively. What is their frame rate for a scene containing 25,000 polygons? What is the maximum scene complexity allowed by each machine if the refresh rate needs to be 30 frames/sec?
12. What is genlock? What are various ways to synchronize graphics pipelines used in side-by-side displays? Which one works better (why)? Make a drawing and explain.
13. In what ways do PC clusters replace multipipe workstations in VR systems?
14. In a single-user distributed architecture, how is the graphics refresh rate decoupled from the haptics one, and why?

15. Draw the network topologies used in multiuser VR for unicast LAN. Explain how updates are transmitted from user to user in each case.

16. Repeat for multicast LAN and for WAN-networked VR.

REFERENCES

- 3Dlabs Inc., 2001a, "Using Wildcat's Genlock and Multiview Option in Visual Computing Applications," online at www.3dlabs.com/product/technology/Multiview.htm.
- 3Dlabs Inc., 2001b, "Wildcat: New Parascale Architecture," online at www.3dlabs.com/product/technology/Multiview.htm.
- Abrash, M., 2000, "A Powerful Game Platform Waiting in the Wings," online at www.ddj.com/documents/s=882/ddj008a/0008a.htm.
- ATI Technologies, 2000, "Fire GL2 User's Guide," ATI Technologies Inc., Marlborough, MA. Also online at support.ati.com/manual-pdf/FGL2-UG.pdf.
- Bui-Tuong, P., 1975, "Illumination for Computer Generated Pictures," CACM, Vol. 18(6), pp. 311-317.
- Burdea, G., 1993, "Virtual Reality Systems and Applications" [Short Course], in Electro '93 International Conference, Edison, NJ.
- Burdea, G., and P. Coiffet, 1993, La Réalité Virtuelle, Hermès, Paris.
- Capin, T., I. Pandzic, N. Magnenat-Thalmann, and D. Thalmann, 1999, Avatars in Networked Virtual Environments, Wiley, Chichester.
- Dinsmore, M., N. Langrana, and G. Burdea, 1994, "Issues Related to Real-Time Simulation of a Virtual Knee Palpation," in Proceedings of Virtual Reality and Medicine-The Cutting Edge, New York, pp. 16-20.

Foley, J., A. van Dam, S. Feiner, and J. Hughes, 1990, Computer Graphics. Principles and Practice, Addison-Wesley, Reading, MA.

Fuchs, H., 1992, "Superman Vision," Pixelation, No. 3.

Girone, M., G. Burdea, M. Bouzit, V. Popescu, and J. Deutsch, 2001, "A Stewart PlatformBased System for Ankle Telerehabilitation," Autonomous Robots, Vol. 10, pp. 203212.

Gouraud, H., 1971, "Continuous Shading of Curved Surfaces," IEEE Transactions on Computers, Vol. C-20(6), pp. 623-629.

Greenhaigh, C., A. Bullock, E. Frecon, D. Lloyd, and A. Steed, 2001, "Making Networked Virtual Environments Work," Presence, Vol. 10(2), pp. 142-159.

Hanrahan, P., 2001, "Multi Graphics-High Performance Distributed and Scalable Graphics on Clusters," University of Utah, February 8, 2001. Online at www.chpc.utah.edu/cc/cc2001/talks/hanrahan/hanrahan.pdf.

Humphreys, G., and P. Hanrahan, 1999, "A Distributed Graphics System for Large Tiled Displays," in Proceedings of IEEE Visualization '99, IEEE/ACM, pp. 215-223.

Humphreys, G., I. Buck, M. Eldridge, and P. Hanrahan, 2000, "Distributed Rendering for Scalable Displays," in Proceedings of Supercomputing 2000, CD-ROM, IEEE/ACM, Online at graphics.stanford.edu/papers/clust-render.

Institute of Electrical and Electronics Engineers, 1995, "Standard for Distributed Interactive Simulation-Application Protocols," IEEE Standard 1278.1-1995.

Intel, 1999, "Intel 820 Chipset. A Revolutionary Architecture for Mainstream Performance PCs in 2000," Intel, Santa Clara, CA. Also online at [//developer.intel.ru/design/chipsets/820/820white.htm](http://developer.intel.ru/design/chipsets/820/820white.htm).

Intel, 2002, "AGP 3.0 Interface Specification," Review 1.0, Intel Co., Santa Clara, CA. Also online at [//developer.intel.com/technology/agp30-final](http://developer.intel.com/technology/agp30-final)

10.pdf.

Macedonia, M., and D. Brutzman, 1994, "MBone Provides Audio and Video Across the Internet," IEEE Computer, Vol. 27(4), pp. 30-36. Also online at <ftp://taurus.cs.nps.navy.mil/pub/mbmg/mbone.html>.

Moller, T., and E. Haines, 1999, Real-Time Rendering, A. K. Peters, Natick, MA.

Montrym, J., D. Baum, D. Digman, and C. Migdal, 1997, "Infinite Reality: A Real-Time Graphics System," in Computer Graphics: SIGGRAPH'97 Proceedings, pp. 293-302.

NVIDIA, 2000, "Quadro2 Product Overview," NVIDIA, Santa Clara, CA.

NVIDIA, 2001, "NVIDIA nForce IGP TwinBank Memory Architecture," Technical Brief, NVIDIA, Santa Clara, CA. Also online at www.nvidia.com/docs/10/12/ATT/nForce_TwinBank_Memory_Architecture_Tech_Brief.pdf.

Pere, E., D. Gomez, G. Burdea, and N. Langrana, 1996, "PC-Based Virtual Reality System with Dextrous Force Feedback," in Proceedings of ASME Winter Annual Meeting, Atlanta, GA, DSC-Vol. 58, pp. 495-502.

Pimentel, K., and K. Teixeira, 1993, Virtual Reality: Through the New Looking Glass, Windcrest McGraw-Hill, New York.

Popescu, G., 2001, "Design and Performance Analysis of a Virtual Reality-Based Telerehabilitation System," Ph.D. Thesis, Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ.

Quantum3D, 2001, "Multi-Channel Synchronization in Visual Simulation Systems," White Paper, Quantum 3D, Inc., San Jose, CA.

Rudrajit, S., T. Funkhauser, K. Li, and J. P. Singh, 2000, "Sort-First Parallel Rendering with a Cluster of PCs," Technical Sketch presented at SIGGRAPH 2000, New Orleans, LA.

Scheffel, U., 2001a, "Elsa Gloria III-Best Bang for the Buck," online at www6.tomshardware.com/graphic/Olq2/010615/opengl-02.htm].

Scheffel, U., 2001b, "Open GL workstation Power-Graphics Cards for the Professional," online at www17.tomshard.conVgraphic/20010615/opengl-1O.html.

Scheffel, U., and U. v. d. Weyden, 2000, "OpenGL Turbobuster-Diamond's Fire GL2," online at www6.tomshardware.com/graphic/00g4/001213/index.html.

Scott, N., D. Olsen, and E. Gannett, 1998, "An Overview of the VISUALIZEfx Graphics Accelerator Hardware," Hewlett Packard Journal, May, pp. 28-34.

Sense8 Co., 2001, "Indy3D V3.0 Graphics Evaluation Tool Documentation," online at www.sense8.conVindy3d/Help/docs.htm.

Singhal, S., and M. Zyda, 1999, Networked Virtual Environments. Design and Implementation, Addison Wesley, Reading, MA.

Standard Performance Evaluation Corporation, 2002, "Viewperf 6.1.2," online at www.specbench.org/gpc/opc.static/overview.htm.

Sun Microsystems, 1993, "Sun Has High Hopes for Virtual Reality," SunWorld, 1993 (April), pp. 32-34.

Sun Microsystems, 2000, "The Sun Blade 1000 Workstation Architecture," Technical White Paper, Sun Microsystems Co., Palo Alto, CA. Also online at www.sun.ru/win/products/ workstations/blade 1 000/pits/sb 1000wp.pdf.

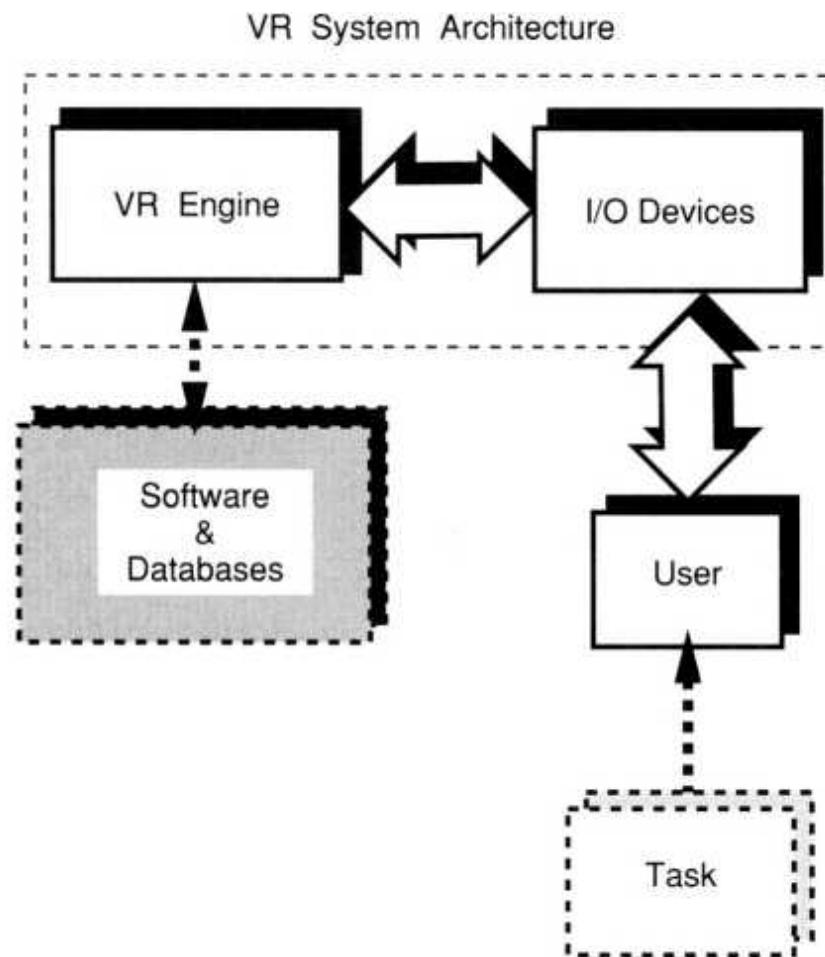
Viewpoint Co., 2002, "Digital Assets Catalog," online at www.viewpoint.com/vp/ catalog. j sp?style=premier.

Vuurwerk, 2002, "SGI Graphics Performance Comparison Tables," online at www.futuretech.vuurwerk.nl/gfxtables.htm].

Williams, J., 1998, "Using Java to Construct a Distributed Virtual Environment," Undergraduate thesis, Curtin University School of Electrical and Computer Engineering, Perth, Australia. Online at [//caffeinebuzz.com/thesis.html](http://caffeinebuzz.com/thesis.html).

CHAPTER 5

MODELING



The I/O devices and VR engines presented in the previous chapters can be seen as the enabling hardware for VR simulations. Another important aspect is the modeling of the virtual world. This means first the mapping of I/O devices with the simulation scene. For example, a sensing glove may control a virtual hand, or an HMD tracker may set the simulation view. VR interface mapping requires calibration to the user's characteristics as well as I/O communication coding. These tasks are usually solved by commercial toolkits, which will be discussed in the next chapter.

The next step following I/O device mapping is the development of object databases for populating the virtual world. This means modeling object shape, appearance, kinematic constraints, intelligent behavior, and physical characteristics (weight, inertia, hardness, etc.). Finally, in order to maintain a real-time interaction with the simulation, the model needs to be optimized during the model management step. Figure 5.1 illustrates the various aspects of VR modeling discussed in this chapter. We focus on the underlying mathematics and the relation to the rendering pipeline presented in Chapter 4.

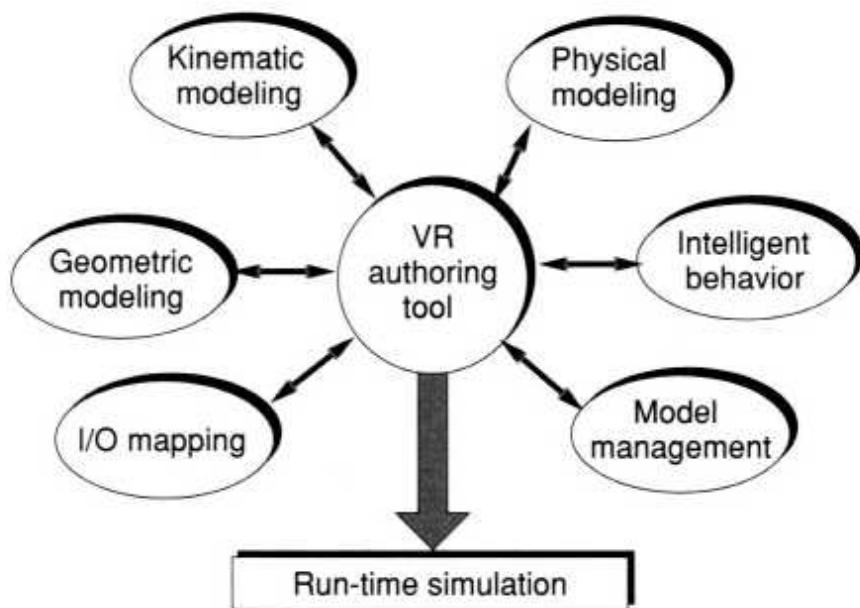


Fig. 5.1 The VR modeling cycle. Adapted from Burdea and Coiffet [1993]. © Editions Hermès. Reprinted by permission.

5.1 GEOMETRIC MODELING

Geometric modeling describes the shape of virtual objects (polygons, triangles, vertices, splines) as well as their appearance (surface texture, surface illumination, and color).

5.1.1 Virtual Object Shape

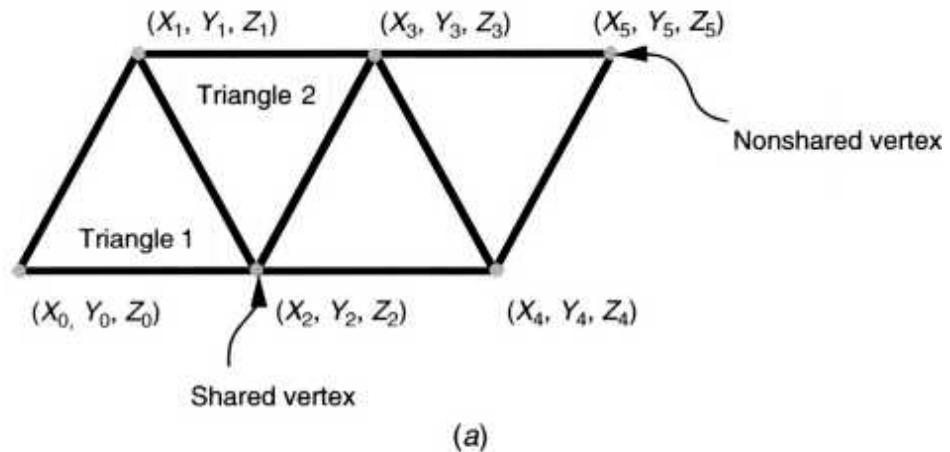
The shape of virtual objects is determined by their 3D surface, which can be described in several ways. The vast majority of virtual objects have their surface composed of triangle meshes. Triangle meshes are preferred because they use shared vertices and are faster to render. As illustrated in Figure 5.2a, the vertices of shared edges are the same for adjacent triangles. An example is the pair (X1, Y1, Z1) and (X2, Y2, Z2), which is shared between triangle 1 and triangle 2. This means that the model using triangle meshes will need less memory to be stored and will be loaded faster by the application stage of the rendering pipeline. Furthermore, as discussed in Chapter 4, certain architectures are optimized to process triangle meshes (through MIMD parallelism) and render them faster than surfaces described by independent polygons. Finally, triangle meshes are well suited for geometric transformations and level-of-detail optimization, as described later in this chapter.

For all their advantages, triangle meshes are not well suited for representing objects with highly curved and bumpy surfaces. The more intricate the detail needed in the curved surface, the more triangles are needed in order to model such objects. An alternate way of shape representation is to use parametric surfaces, as illustrated in Figure 5.2b. A parametric spline is represented by the points $x(t)$, $y(t)$, and $z(t)$, as given by

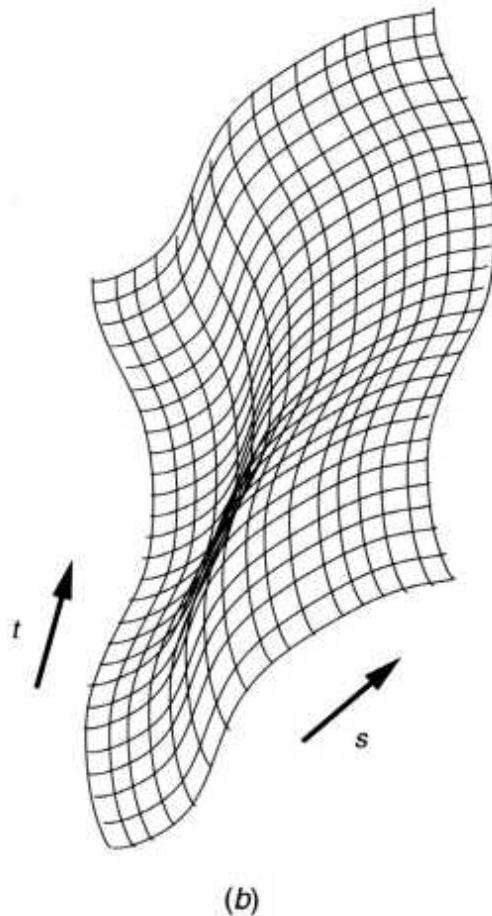
$$x(t) = ax \cdot t^3 + bx \cdot t^2 + cx \cdot t + da \quad (5.1)$$

$$y(t) = ay' \cdot t^3 + by' \cdot t^2 + cy' \cdot t + dy \quad (5.2)$$

$$z(t) = az \cdot t^3 + bz \cdot t^2 + cz \cdot t + dz \quad (5.3)$$



(a)



(b)

Fig. 5.2 The object shape description: (a) using a triangle mesh; (b) using a parametric surface. Adapted from Foley et al. [1994]. © Pearson Education Inc. Reprinted by permission.

where t is a parameter with values between 0 and 1, and a , b , and c are constant coefficients. Parametric surfaces are extensions of splines obtained

by introducing a second parameters also between 0 and 1. Thus points on a parametric surface are given by coordinates $x(s, t)$, $y(s, t)$, and $z(s, t)$ [Foley et al., 1994]. Since parametric surfaces are represented by functions of higher order than the linear functions describing a polygon, they use less storage and provide increased surface smoothness. Parametric surfaces are at a disadvantage when it comes to surface deformation. Whereas the shape of a polygonal surface can easily be changed by direct vertex manipulation, parametric surfaces do not allow that. Their shape can only be changed by changing the position of control points, which are located outside the surface itself.

There are several methods by which object surfaces can be constructed. These involve the use of a toolkit editor, importing CAD files, using a 3D digitizer, scanning the object with a 3D scanner, or purchasing models from existing databases. Regardless of the method used, the resulting surface is monolithic (or static) and does not allow any relative motion of its components.

5.1.1.1 Using a Toolkit Editor. All graphics programming languages allow 3D polygonal object shape to be created from scratch. Such capability is then exploited by the programming toolkit layer that creates an authoring tool. Text-based specification of vertex coordinates and connectivity is necessary to create the shape, and normals need to be specified if smooth shading is desired. The process is based on trial-and-error iterations and requires an amount of time inversely proportional to the programmer's skill. This approach is prohibitively expensive if the virtual scene has extremely complex models, requiring a significant amount of person-hours to develop.

5.1.1.2 Importing CAD Files. An alternate way to create 3D objects is to use a CAD program, such as AutoCAD or 3-D Studio. These programs are the de facto standard in mechanical and architectural design. Thus, preexisting models of mechanical assemblies or buildings can be reused to populate the virtual world. Alternatively, the user can interactively create the object's model, such as the house illustrated in Figure 5.3. Various moving objects within the house, such as the front door or a ceiling fan, need to be animated separately in the VR simulation. Therefore they have to be created and saved

under separate files (because there is no standard way to keep objects separate in the AutoCAD DXF file format). Once the AutoCAD object files are saved, they need to be converted into formats compatible with the particular VR toolkit used to run the real-time simulation.

5.1.1.3 Creating Surfaces with a 3D Digitizer. This approach is taken when a given virtual object used by the simulation is not part of an existing database. One solution is to use a 3D digitizer, such as the Immersion MicroScribe 3D mechanical arm (illustrated in Fig. 2.23) with a modeling toolkit such as form•Z [Autodesk Inc., 1998]. The process of digitizing a clay model of the object starts with calibration of the MicroScribe. This is done by selecting three points in space and mapping them with three points on the computer screen selected using a mouse. This process establishes the distance scaling as well as the orientation of the model in relation to the virtual space. The user needs to trace a mesh on the real object surface and select desired vertices by pressing the MicroScribe pedal. Symmetrical objects have the advantage that the manual digitization process needs to be done only for half of their surface. The other (symmetrical) half is obtained by mirroring the data already input in the computer and subsequently stitching the two halves. The form•Z allows an increase in object level of detail by automatically subdividing surface patches. This process creates finer contours without requiring additional vertices to be digitized by the user.

Another 3D digitizer is the HyperSpace produced by Mira Imaging Inc. Unlike the form-Z, the HyperSpace digitizer has a Polhemus stylus, which is used to get 3D point information. As illustrated in Figure 5.4a, the stylus has a miniature 3D tracker receiver and can be used with two emitters for increased accuracy. The user places the stylus on the model surface (such as a statue) to obtain its wireframe vertices. The vertex (x, y, z) coordinates are stored when the user presses the built-in pushbutton in the stylus to initiate the data acquisition. A built-in editor later allows line segments and polygons to be repositioned along any axis. Points within line segments and points connecting polygons may be fit to Bezier curves and reshaped. An example of a model produced in this way is the digitized statue of Venus de Milo shown in Figure 5.4b. The model was created with 4200 textured

polygons and the scene textured using the NuGraph toolkit [Okino Inc., 1997].

HyperSpace, form-Z, and similar 3D digitizers come with a set of translators that format the data in DXF, Wavefront, Symbolics Swivel 3D, and other popular file formats, as well as simple ASCII. This allows programming flexibility and easy rendering by many of the toolkits discussed in Chapter 6. A drawback of 3D digitizers is their limited work envelope, which makes them impractical to use on large objects. Furthermore, the digitization procedure is done one point at a time, which is timeconsuming. Finally, hand tremor and tracker noise limit accuracy to about 1 mm (at best).

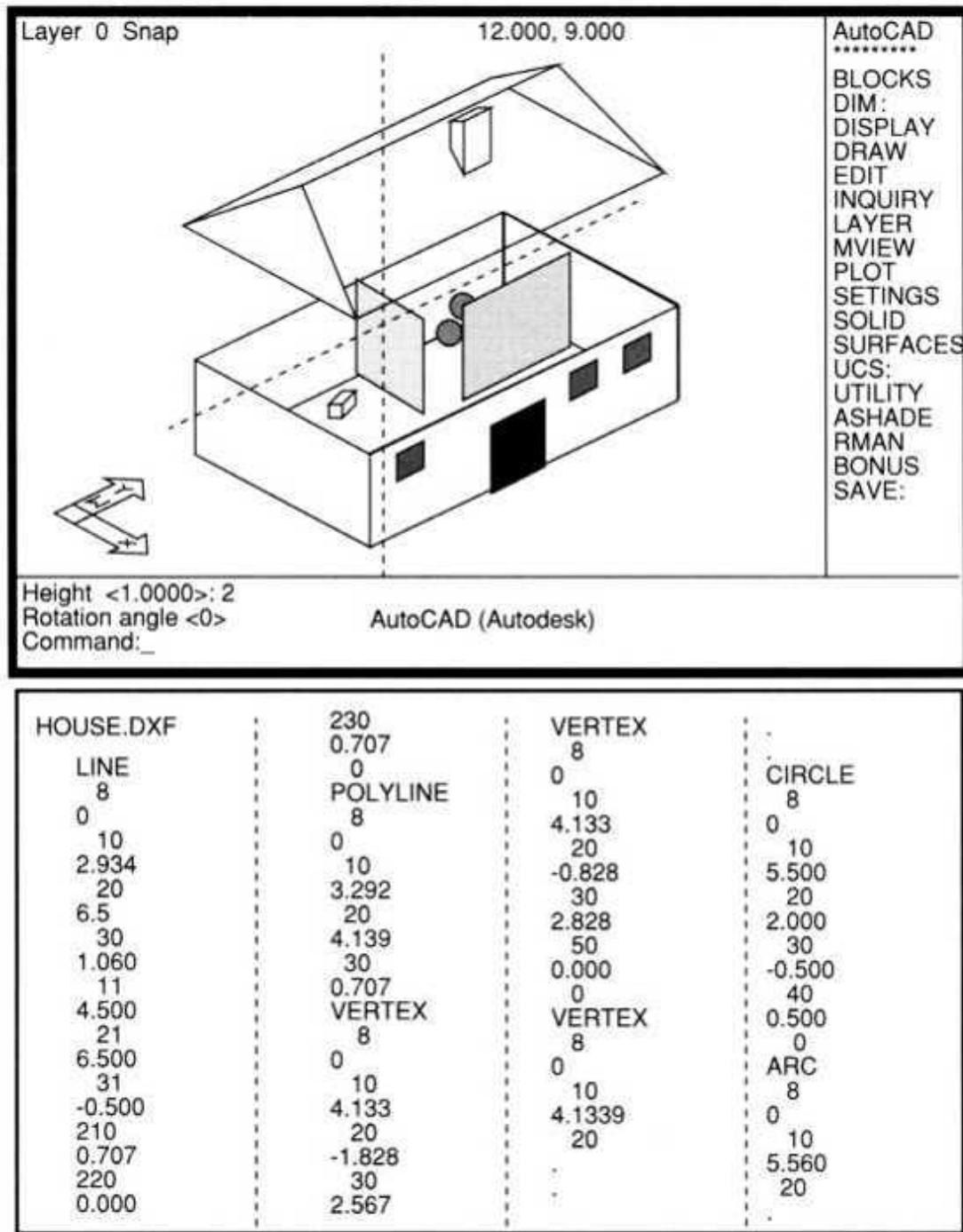


Fig. 5.3 A virtual house created with AutoCAD. Programming credit Daniel Gomez. From Burdea and Coiffet [1993]. © Editions Hermes. Reprinted by permission.

5.1.1.4 Creating Surfaces with a 3D Scanner. This represents a faster way to build 3D models, especially for large objects. These scanners have a digital

cameralaser assembly, which measures the distance to an object through triangulation. The distance is obtained based on the known geometry of the camera-laser assembly (distance and angle) and the image information of the reflected laser beam. The 3D scanner's advantages over the 3D digitizers previously discussed are noncontact measurement and the ability to get substantially more measurement points in a given amount of time.

An example of a handheld 3D scanner that works up to about 0.7 in away is the Polhemus FastScan, shown in Figure 5.5a [Polhemus, 1998]. It consists of a laser and two miniature monochrome cameras integrated in a rigid structure that holds a Fastrack receiver. The tracker source is placed close to the object being scanned, such that the position and orientation of the laser beam on the object surface are known. Since the scanning speed is limited (50 lines/sec), motion of the object being scanned will introduce errors. In such cases a second receiver is attached to the object that could inadvertently move during scanning (such as a human face) in order to compensate for these errors.

The accuracy of triangulation-based scanning is degraded rapidly with distance to the object surface. This is due to the digital camera's inability to distinguish between adjacent dots when the object is too distant. Furthermore, if the laser beam is not properly focused, it can spread to a cross-section diameter of 12 cm at 100 m away. This degrades the triangulation results even more. An alternative is to supplement laser data with radar time-of-flight data to determine range. This is the approach taken by the Cyra Technologies Cyrax 2500 laser scanner, used for modeling large and distant objects [Goldberg, 2001]. The scanner has a single-point accuracy of 6 mm at a range of 50 m and a maximum range of 100 m. Thus practically any size objects, from battleships to buildings, can be scanned.

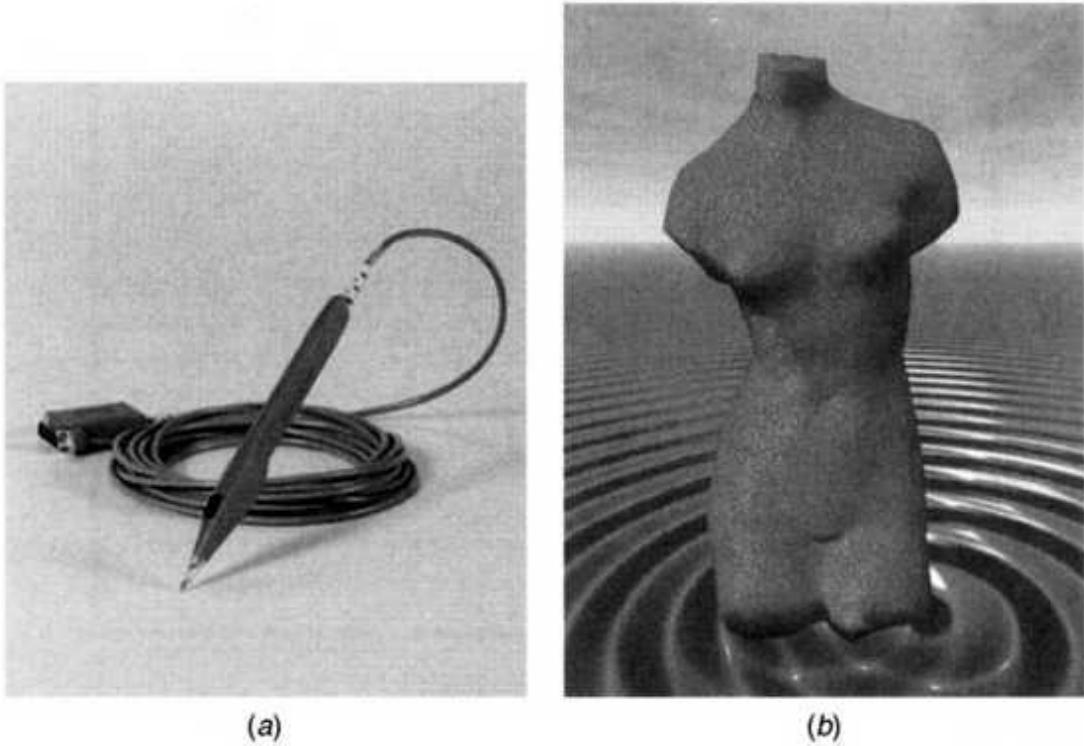


Fig. 5.4 Creating a model with a 3D digitizer: (a) the Polhemus stylus; Courtesy of Polhemus Co. (b) the Venus de Milo digitized model. From Okino Inc. [19971. Reprinted by permission.

Whether scanning is done at short or at long range, the data are in the form of a cloud of dots, each having (x, y, z, color) information. These data need to be first processed before they can be stored as a polygonal or spline-based model. Some scanner drivers perform this data conversion. Alternately, third-party conversion software can be utilized. Figure 5.6 depicts the stages of such processing performed by the Wrap software [Raindrop Geomagic, 1999]. The dense cloud of dots is first converted to a triangle mesh by interconnecting neighbor dots. The second stage is polygonal decimation, in order to reduce the model complexity by as much as 95%. Finally, the polygonal model is converted to a NURBS (nonuniform rational ,B-spline) parametric surface format for improved appearance. Some scanners use their camera to take photo images of the scanned object. Such images are then converted to textures and pasted over the object surface to increase realism.

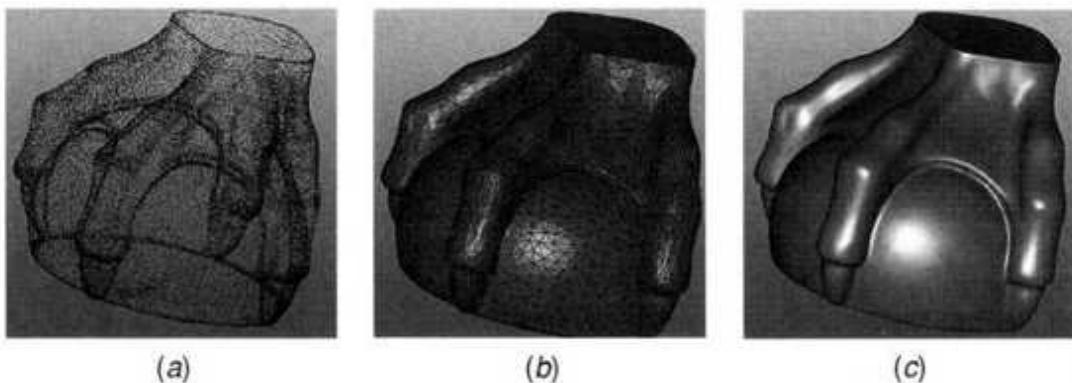


Fig. 5.6 The conversion of scanner data for a claw foot piece of furniture: (a) dot cloud; (b) decimated triangle mesh; (c) NURBS surface. From Raindrop Geomagic [1999]. Reprinted by permission. Geomagic Wrap is a trademark of Raindrop Geomagic Inc.

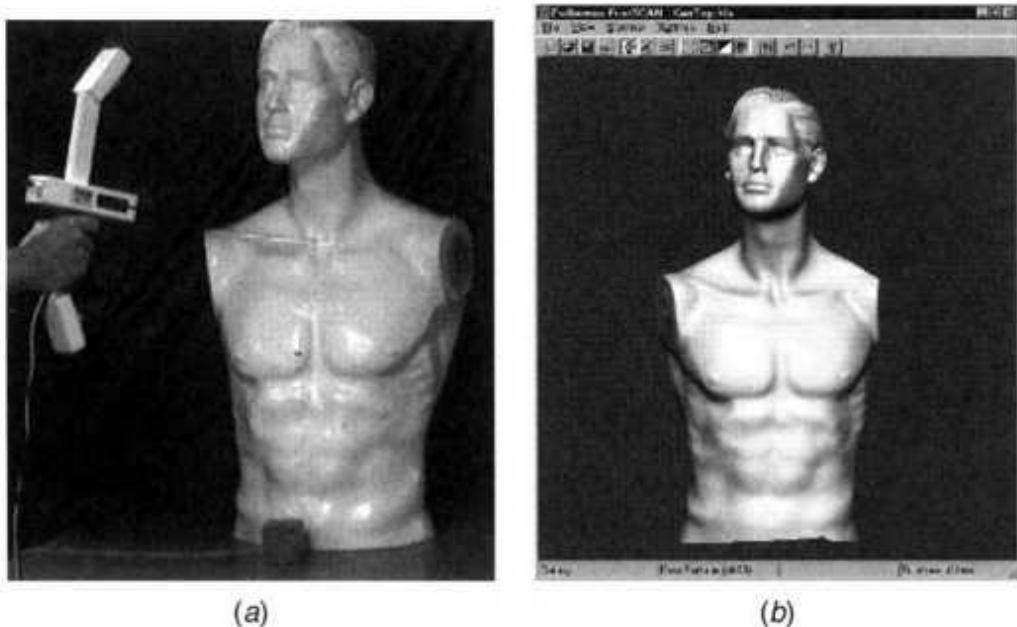


Fig. 5.5 The Polhemus FastScan 3D scanner: (a) original statue; (b) model after data processing. From Polhemus [1998]. Reprinted by permission.

5.1.1.5 Using Online 3D Object Databases. This involves purchasing already created 3D models from various commercial databases. Such models usually have three levels of detail, low (L), medium (M), and high (H), according to the number of polygons in the model. An example the reader is familiar with is the Viewpoint dinosaur models shown in Figure

4.5. The need to have several models of the same object relates to level-of-detail model management techniques discussed at the end of this chapter. Each model comes with two data files, one containing vertex coordinates and the other for vertex connectivity. It is then possible to use the local graphics library to transform these numbers into a wireframe or a shaded model of that object. Purchasing 3D models is fast, but can be expensive (the price of a model grows with its level of detail). Furthermore, the choice is limited to the database contents, which are by necessity finite.

Table 5.1 summarizes the various methods for modeling 3D object geometry discussed in this chapter.

5.1.2 Object Visual Appearance

Modeling the geometry of a virtual object is the necessary first step in creating a realistic-looking 3D scene. The next step is to illuminate the scene such that the object becomes visible. The appearance of an object will depend strongly on the type and placement of virtual light sources as well as on the object's surface reflectivity coefficient (as discussed in Chapter 4). Another factor determining scene complexity and realism is surface texture. Thus our discussion here focuses on scene illumination and surface texturing.

5.1.2.1 Scene Illumination. This determines the light intensities on the object surface. It can be classified into local illumination and global illumination.

TABLE 5.1. Methods for Modeling 3D Object Geometry

Method	Feature	Source
Toolkit editors	Tedious, requires skill	OpenGL, Starbase, PHIGS
CAD programs	Interactive, existing technology	AutoCAD (Autodesk), 3-D Studio, etc.
3D digitizers	Interactive, allows custom models	Autodesk Inc., Mira Imaging, Polhemus Inc., etc.
3D scanners	Fast multipoint acquisition, large objects	Polhemus Inc., Cyra Technologies, etc.
Commercial 3D databases	Vertex list, connectivity, static model, level of detail	Viewpoint Inc., etc.

Definition Local scene illumination treats the interactions between objects and light sources in isolation, neglecting the interdependences between objects. Global illumination models the interreflections between objects and shadows, resulting in a more realistic-looking scene.

A common local illumination method which shades objects based on light intensity interpolation is Gouraud shading [Gouraud, 1971]. This shading mode requires that the normal of each vertex of the polygonal mesh describing the surface of the virtual object be known. The vertex normal can be determined analytically or by averaging adjacent polygon normals, as shown in Figure 5.7a.

The averaged vertex normal, \bar{N}_v is given by

$$N, N_i \sim \text{in } -t \quad (5.4)$$

Vertex normals are used to determine vertex illumination following the models described in Chapter 4. Subsequently, edge intensities I_u and I_b are determined from vertex intensities I_1 , I_2 , and I_3 , as illustrated in Figure 5.7b. Finally, intensities at a point on each scan line inside a polygon are interpolated based on I_Q and I_b [Foley et al., 1990]:

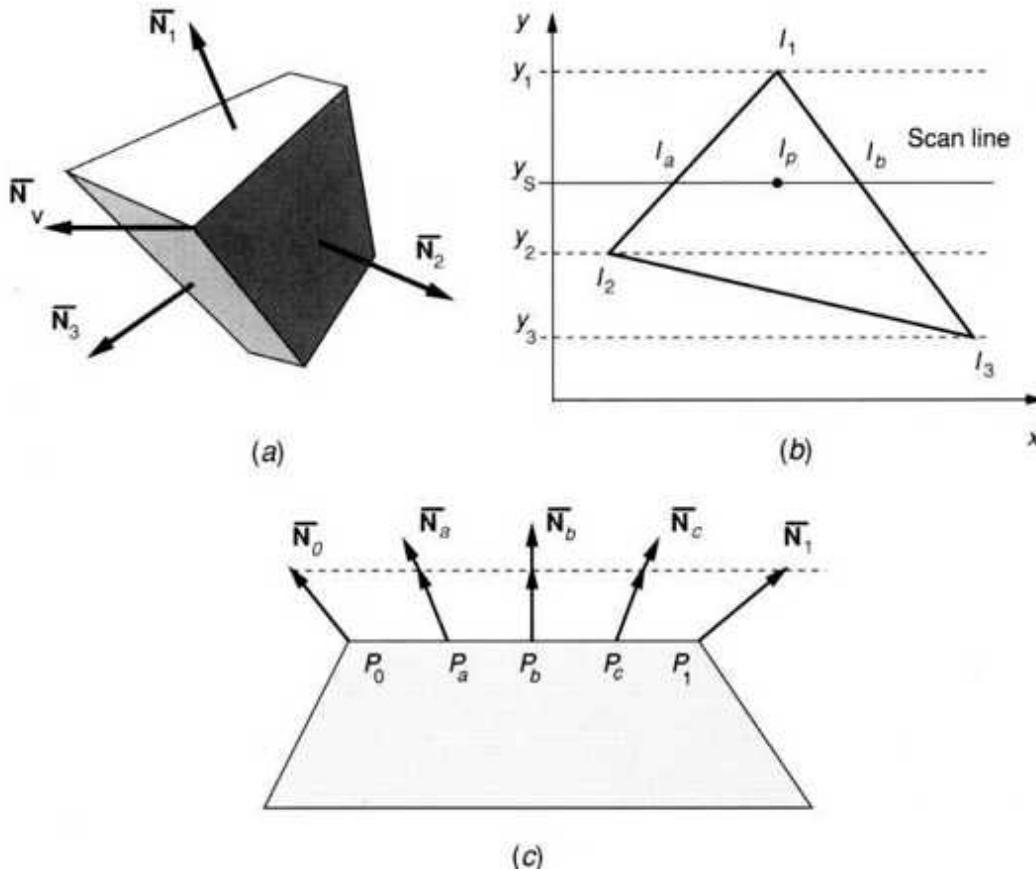


Fig. 5.7 Polygon shading modes: (a) vertex normals obtained from averaging polygon normals; (b) light intensity interpolation; Adapted from Foley et al. [1990]. © Pearson Education Inc. Reprinted by permission. (c) normal vector interpolation. Adapted from Bui-Tuong [1975]. © 1975 ACM Inc. Reprinted by permission.

$$I_a = I_t - (I_t - I_2) \frac{Y_t - Y_s}{Y_1 - Y_2} \quad (5.5)$$

$$I_b = I_t - (I_t - I_3) \frac{Y_1 - Y_s}{Y_1 - Y_3} \quad (5.6)$$

$$I_p = I_b - (I_b - I_a) \frac{x_b - x_s}{x_b - x_a} \quad (5.7)$$

The resulting object shape has improved smoothness, as previously seen in Figure 4.7c. However, the number of light intensity values I_i increases compared to flat shading, and the frame refresh rate decreases if the pipeline is transform-limited.

A complex local illumination method called Phong shading [Bui-Tuong, 1975] is based on normal-vector interpolation. This model first determines the edge normals based on vertex normals, as illustrated in Figure 5.7c. The normals inside the polygon are then obtained from interpolation of normals on the edges along a scan line. The surface normals are then normalized and the corresponding light intensity I_A calculated with the illumination models previously introduced. The greater accuracy in normal calculation results in better discrimination of the angle α , and subsequently a more accurate simulation of specular reflection. As seen in Figure 5.8b [McAllister, 1998], the shiny Utah Teapot surface has a much more realistic appearance than the same surface flat-shaded. The drawback of Phong shading is increased computation load, since it involves three additions, one division, and one square root for each pixel of the polygon being shaded. Until the late 1990s consumer-grade graphics boards did not have the computation power to allow real-time Phong shading, so it was rarely used in VR simulations. Advances in PC hardware, coupled with fast Phong shading algorithms [Bishop, 1986, Mortensen, 1995], are overcoming this limitation.

Although Phong shading does provide good realism for individual objects, it does not account for interdependences between objects, and thus does not provide sufficient scene detail. Global illumination models do account for object interreflections and shadows. As a result, these algorithms produce photorealistic scenes, as shown in Figure 5.9 [Soler and Sillion, 1998].

One approach taken in global illumination is to model object radiosity. This method considers that each object in the scene (including virtual light sources) emits an energy flow in the form of light. The amount of energy radiated by the objects per unit surface and unit time determines its radiosity. Modeling radiosity starts by dividing the scene into equal patches and computing patch vertex colors. Patches are then subdivided and the process repeated through several iterations. Thus radiosity modeling is extremely computation-intensive, and traditionally has been done offline, prior to rendering. This limited the type of VR simulations that use radiosity illumination to architectural walkthrough or the simulation of other static worlds. During a walkthrough the user simply navigates through the scene by changing the viewpoint. Since this process does not affect either the scene

geometry or object surface coefficients, the radiosity computation is not affected and need not be done at rendering time.

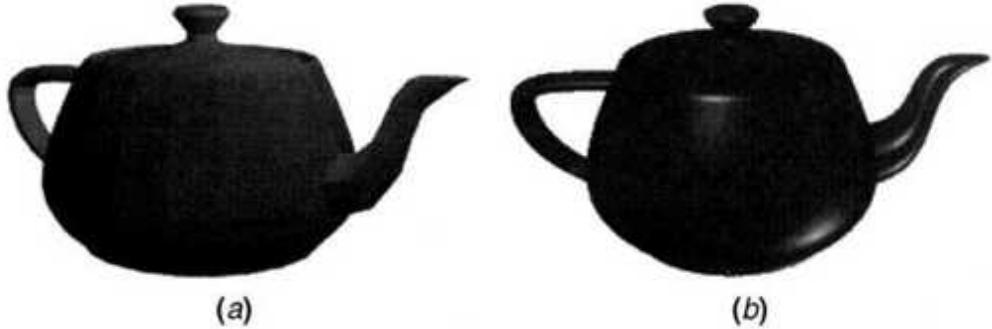


Fig. 5.8 The Utah Teapot appearance versus shading mode: (a) flat-shaded; (b) Phong-shaded. From McAllister [1998]. Reprinted by permission.

Researchers at Fraunhofer Institute for Computer Graphics in Germany have developed an approach that allows interactive VR scenes to have global radiosity-based illumination [Schoffel, 1997]. As illustrated in Figure 5.10, a second processor is allocated the radiosity computation load and communicates with the main graphics pipeline through shared memory. At simulation initialization, the same geometry is loaded in the application stage of the graphics pipeline and in the radiosity process. The iterative radiosity computations determine the patch vertex colors, which are then sent to be rendered. Once this is done, the scene is globally illuminated, and shading will not change as long as only the viewpoint is changed by the user. In other words, the user interacts only with the rendering pipeline in real time (in this case). If the user decides to change the scene geometry by grasping and moving objects, the event is sent to the radiosity process through shared-memory message passing. The coprocessor needs to recompute the global illumination, taking advantage of scene coherence. This means that the user's actions generally affect only a small subset of the scene shading, and thus computations can be optimized. Once the new radiosity data are obtained, they are sent to the graphics pipeline, again through shared memory. This allows interactivity, since the scene is no longer static, at the expense of a small delay in radiosity updates. Global illumination recomputation can also be triggered automatically rather than by

the user's actions. This is the case of intelligent object behavior or time-of-day change, which affects the light coming from a virtual sun, for example.



Fig. 5.9 Global radiosity-based illumination. From Soler and Sillion [1998]. Reprinted by permission.

5.1.2.2 Texture Mapping. This is a method of improving the image realism without the need for additional surface polygons. The location of an object vertex is stored at modeling time in object coordinate space (x , y , z). The modeler also stores the location in parametric space [Moller and Haines, 1999]. For parametric surfaces, such as splines, this is implicit through the pair (s, t) , as previously discussed. Then, during the geometry stage of the graphics pipeline the object is shaded, such that each pixel is given a color based on vertex color and interpolated light intensities (Gouraud shading) or interpolated normals (Phong shading). Catmull [1974], and later Blinn and

Newell [1976], realized that providing intricate details only through polygons was too time-consuming and computationally expensive.

Definition Texturing is a technique performed in the rasterizing stage of the graphics pipeline in order to modify the object model's surface properties such as color, specular reflection, or pixel normals.

As illustrated in Figure 5.11 [Foley et al., 1990], the texturing process uses a responder function to change object parametric coordinates to texture space coordinates. The texture coordinates are the (u, v) index in the texture array. The size of the texture array depends on the operating system requirements, and is usually square. The color of the corresponding texel is then retrieved by the rasterizer and used to change the shaded model pixel color. This process, called modulation, multiplies the surface color (output by the geometry engine) by the texel color.

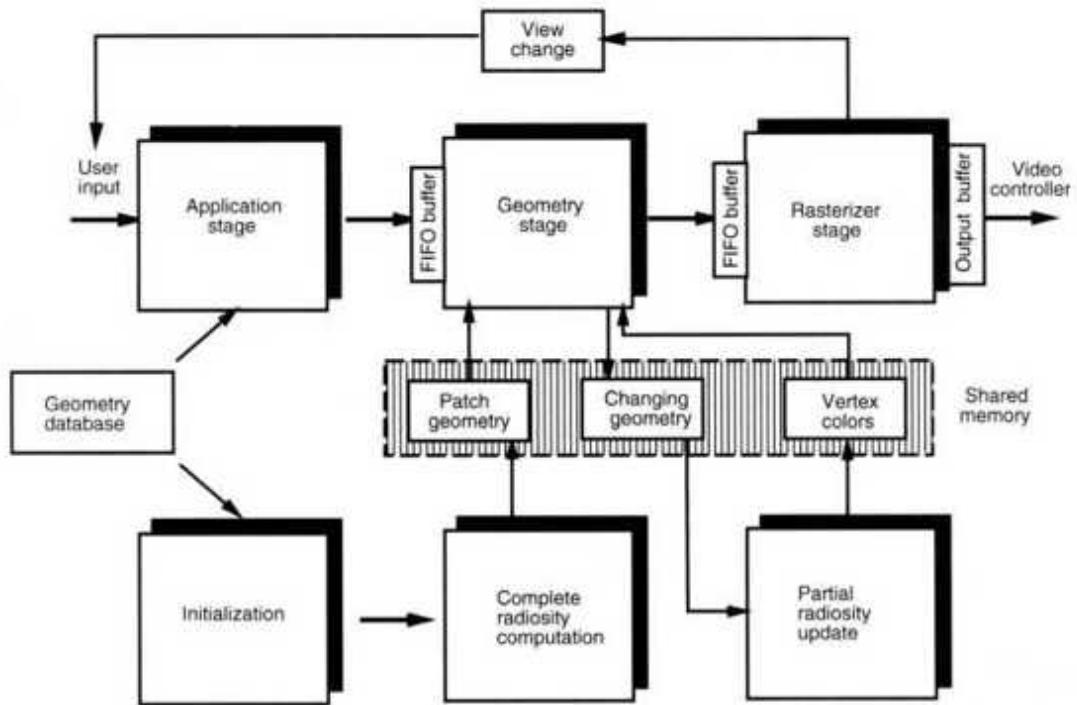


Fig. 5.10 Radiosity process synchronization with a graphics rendering pipeline. Adapted from Schoffel [1997]. © 1997 ACM Inc. Reprinted by permission.

Textures may be created in several ways. First, a paint program can be used to interactively create and store texture bitmaps. Another approach is to use a photograph of the desired texture and electronically scan it. Tools such as RealTexture [Computer Graphics Systems Development, 2000] help the process by compensating for foggy or misaligned photographs. An analytical function can also create textures on the fly, resulting in an economy of memory and bus traffic (from RAM to local texture memory on the graphics accelerator). Finally, there are online commercial texture databases, such as ImageCELS [Realworld Imagery Inc., 2002]. This texture library consists of thousands of textures for buildings, landscapes, trees, objects, people, and vehicles. The textures are seamless, allowing infinite replication (to cover large scenes) and scaling. Furthermore, ImageCEL textures can be modified into additional texture types by overlaying one on top of another. The texture library is provided in high-resolution TIFF files (32 bit), mostly being arrays of 512 x 512 texels.

Textures offer several advantages for the VR simulation. First, they increase the level of detail and scene realism. A case in point is Figure 5.9, where texturing was used extensively (for walls, paintings, floor, etc.). The second advantage that textures provide is better 3D spatial cues owing to perspective transformations. Last, but not least, textures substantially reduce the number of polygons in the scene, and can therefore increase the frame refresh rate. An example is Figure 5.12, which shows two trees. The first one requires 45,992 polygons to be rendered, while the second one is constructed with a single textured polygon! The criticism with single-polygon-textured trees (or other 2D models) is that the user cannot walk around them. This can be easily remedied by placing two similarly textured polygons bisecting at a 90° angle. This way the tree will appear three dimensional, and will cast a shadow (as the polygonal tree does) [Reed, 1990]. Use of three identical textured polygons trisecting at 60° is another solution. A scene using the Viewpoint tree and rendered by the Wildcat II card (the best performance in Table 4.2) will have 11 such trees displayed at 30 frames/sec. However, the same VR engine (332 Mpixels/sec) will render a whole forest of textured trees per scene, and still at 30 frames/sec.

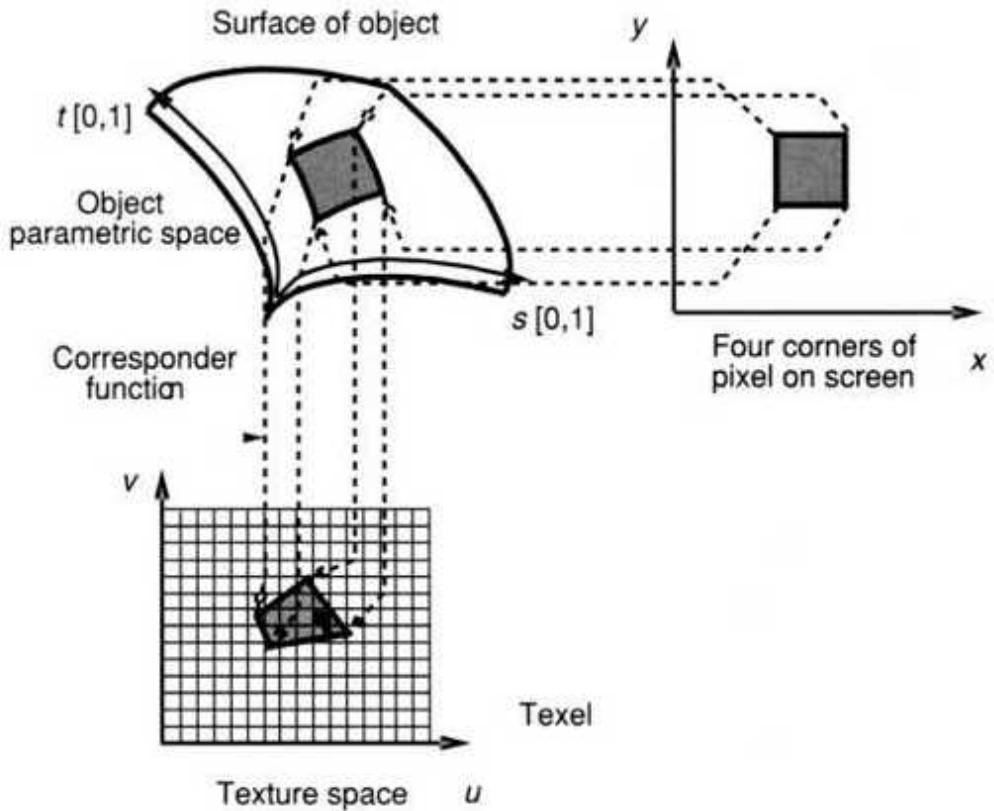
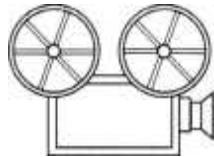


Fig. 5.11 Texture mapping (from screen pixels to texels.) Adapted from Foley et al. [1990]. © Pearson Education Inc. Reprinted by permission.

In the foregoing discussion we assumed that a single texture is mapped to a given polygon. In fact the process can be repeated to further improve the simulation realism. As illustrated in Figure 5.13a, this multitexturing process applies a second texture over the output of the first texturing process, and the result is then stored in the frame buffer. If necessary the blending cascade can have more than two stages. A useful example of multitexturing is bump mapping [Blinn, 1978]. A bump map is a monochrome texture that represents surface height information by a degree of gray. When such texel data modulate the object surface data, the normals at each pixel are changed. This results in much finer surface detail, without additional polygons. As seen in Figure 5.13b, the candle surface and the vase are multitextured with the bump map shown to the right of the image. The result is extremely realistic, especially for the melted wax detail.



VC 5.1

Another type of texture that can be used in blending cascades uses light maps. As shown in Figure 5.14a, a light map is a gray-scale, low-resolution texture that models the light produced by a given light source (in this case a spot light). The bright areas correspond to higher light intensities and darker areas map to lower light intensities. When these light maps are part of the blending cascade, they produce a high-quality visual effect, as seen in Figure 5.14b for an illuminated brick wall. Thus light maps produce higher quality lighting effects at lower polygonal counts than a visually similar scene shaded using vertex information. Unfortunately, since light maps are applied in the last stage of the rendering pipeline, they lack the flexibility associated with geometry-stage lighting. Thus a light map needs to be precomputed for each possible light source type and orientation. Newer graphics architectures, such as NVIDIA Quadro2 Pro, incorporate a shading rasterizer in their GPU, which allows dynamic use of light maps. This is done by providing each texel with normal vector information, contained in a special texture called a normal map. This information is then used in per-pixel dot-product mapping [NVIDIA, 2000]. Unlike vertex lighting approaches, the rendering pipeline performance using pixel normals is no longer affected by the number of light sources in the scene. Lighting is both high quality and dynamic regardless of how many light maps are added.

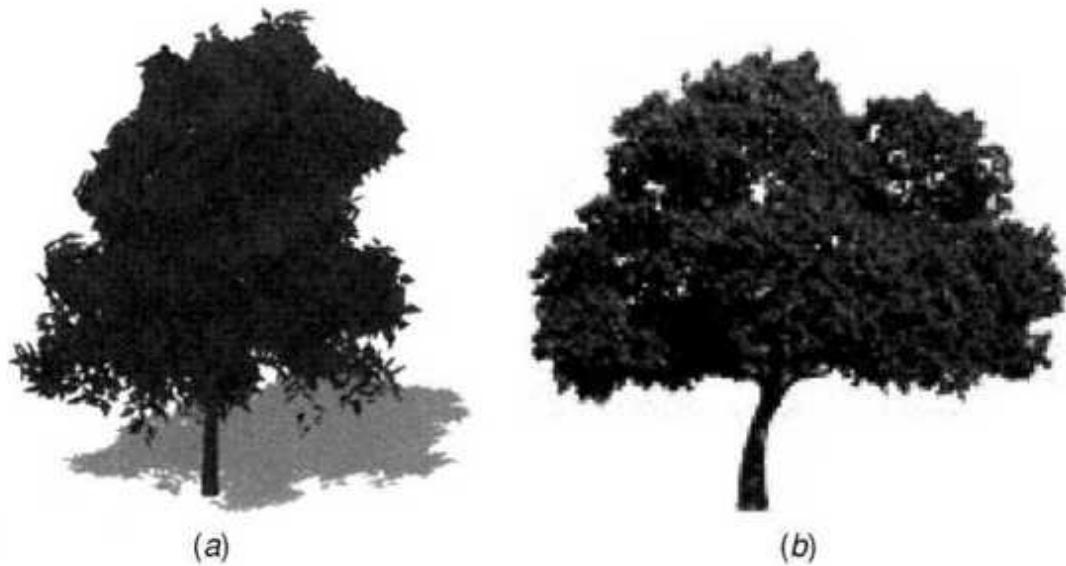
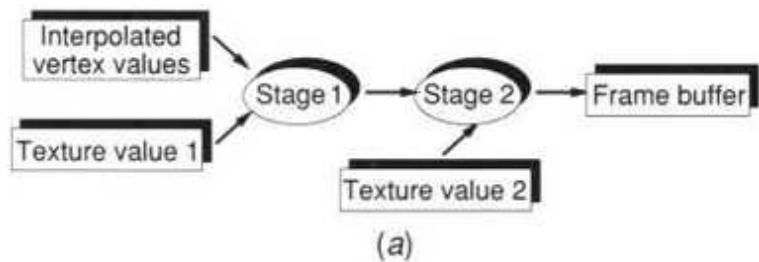
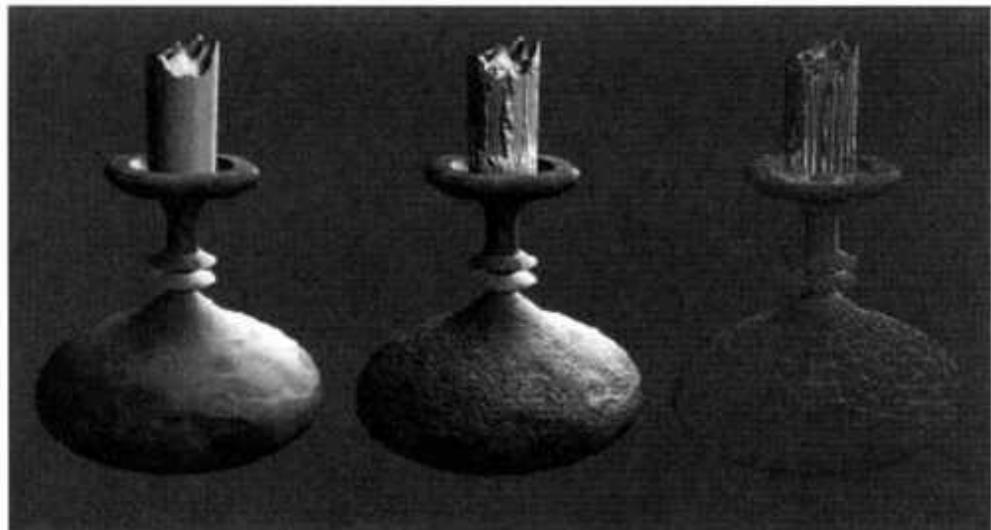


Fig. 5.12 Reduction in polygon count owing to surface texture. (a) Tree with 45,992 polygons. From Viewpoint Inc. [2002a]. Reprinted by permission. (b) Tree with one textured polygon (1246 x 1080 pixels). From Realworld Imagery Inc. [2002]. © 2002 Realworld Imagery Inc. Reprinted by permission.

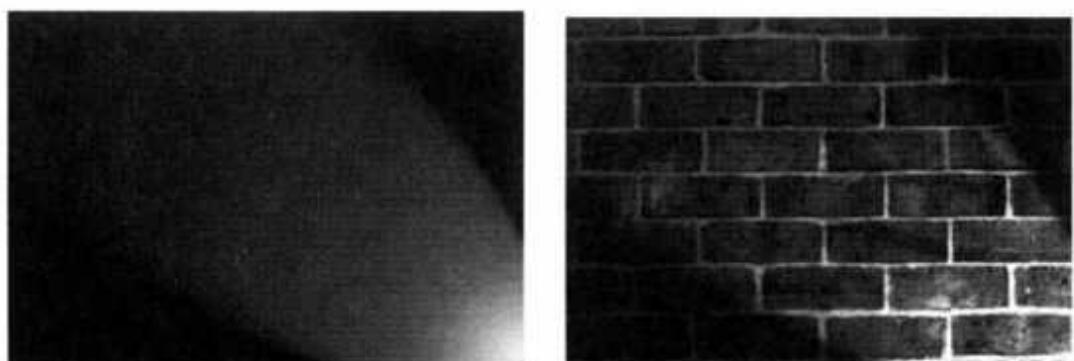


(a)



(b)

Fig. 5.13 Multitexturing. (a) Blending cascade principle. Adapted from Moller and Haines [1999]. Reprinted by permission. (b) Bump mapping example. From NVIDIA [2000]. © 2000 NVIDIA Co. Reprinted by permission.



(a)

(b)

Fig. 5.14 Multitexturing for lighting: (a) standard 2D light map texture; (b) light map texture overlaid on top of wall texture. From NVIDIA. [2000]. ©

2000 NVIDIA Co. Reprinted by permission.

5.2 KINEMATICS MODELING

The next step in virtual world modeling, following the setting of object shape and appearance, is kinematics modeling. This determines the location of 3D objects with respect to a world system of coordinates as well as their motion in the virtual world. Object kinematics is governed by parent-child hierarchical relations, with the motion of a parent object affecting that of its child. A further aspect of kinematics modeling is the way the world is viewed, namely the motion of a virtual camera. Finally the camera image needs to be transformed and projected in the 2D display window in order to provide visual feedback to the user.

5.2.1 Homogeneous Transformation Matrices

Denavit and Hartenberg [1955] and later Fu et al. [1987] and Foley et al. [1990] chose 4×4 homogeneous transformation matrices to express object translations, rotations, and scaling. A homogeneous transformation matrix is given by the generic equation

$$R_{3x3} \ P_{3x1} \ TA-B = 0 \ 0 \ 0 \ 1 \quad (5.8)$$

where R_{3x3} is the rotation submatrix expressing the orientation of system of coordinates B with respect to system of coordinates A, and P_{3x1} is the vector expressing the position of the origin of system B with respect to the origin of system of coordinates A.

Here the systems of coordinates A and B are orthonormal, having Cartesian triads of vectors (i, j, k) such that norm $\|i\| = \|j\| = \|k\| = 1$, and the dot product of any two is zero. Each column in the R_{3X3} submatrix represents the projection of the B system of coordinates along the triad of vectors (i, j, k) of the A system of coordinates.

The 4×4 homogeneous transformation matrices offer certain computational advantages. First, they treat both rotations and translations in a uniform way. Second, they can be compounded, such that complex

kinematics relationships can be modeled. Finally, they are easily invertible as in

$$RT^{-1} RT T P TBE-A = (TAB)^{-1} = 0 \ 0 \ 0 \ 1 \quad (5.9)$$

5.2.2 Object Position

We have seen that object surface modeling uses (x, y, z) vertex coordinates expressed in an object system of coordinates. This system of coordinates is attached to the object, usually at its center of gravity, and oriented along the object axes of symmetry. When the object moves in the virtual world, its system of coordinates moves with it. Therefore, the position and orientation of object vertices in the object system of coordinates remain invariant, regardless of the object position in the scene. This is true as long as the object surface does not deform or is not cut.

Figure 5.15a shows an example of a virtual prism which we will call object 1. It has an attached Cartesian system of coordinates (i_1, j_1, k_1) . The system of coordinates has its origin at the object's center of gravity and overlaps the prism's three axes of symmetry. The absolute position of object 1 is referenced to a fixed system of coordinates called the world system of coordinates (i_w, j_w, k_w) . The transformation between object 1's system of coordinates and the world system of coordinates is given by

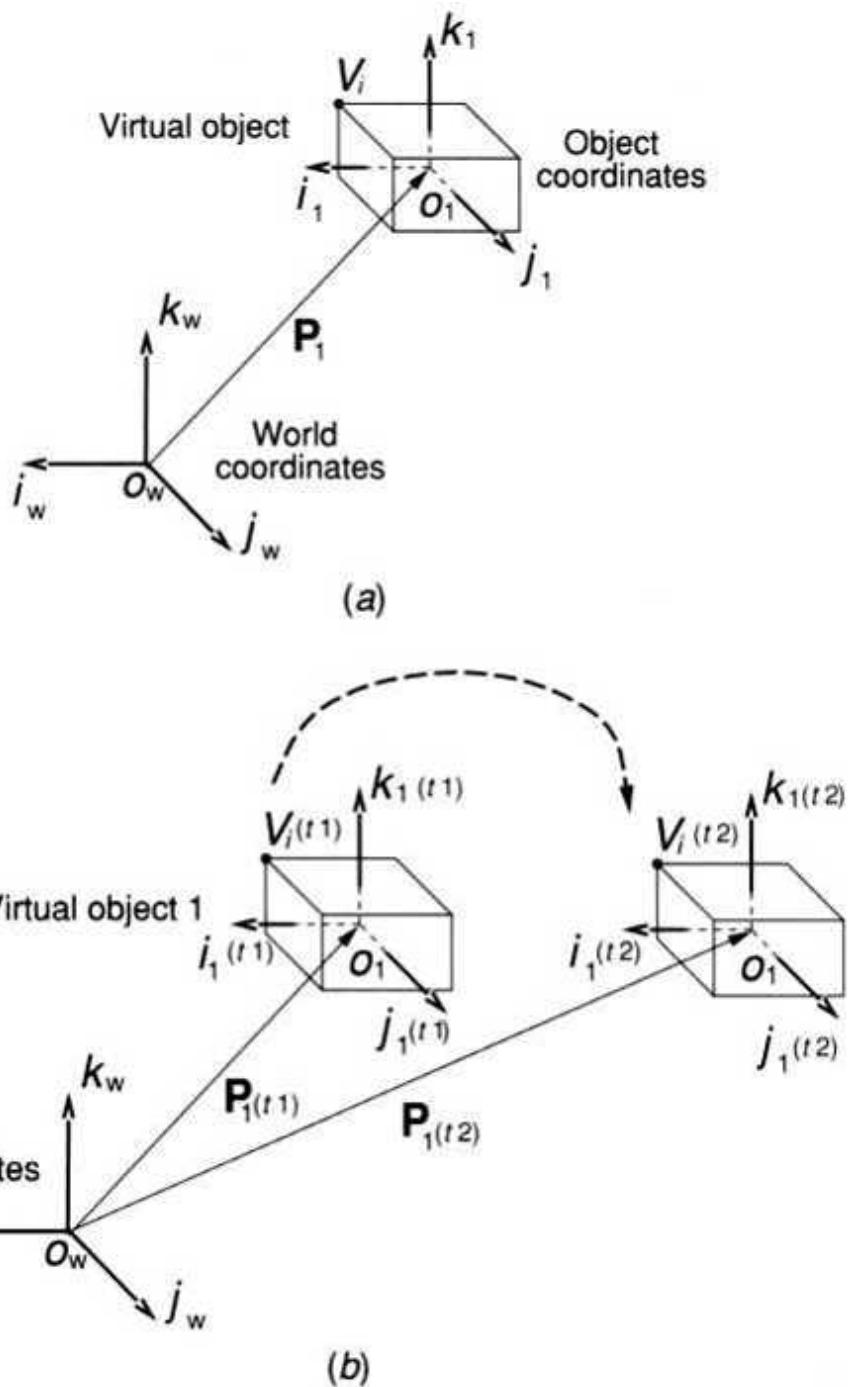


Fig. 5.15 Virtual object's position in the world system of coordinates: (a) static object; (b) translating object. Adapted from Burdea and Coiffet [1993]. © Editions Hermès. Reprinted by permission.

$$i_w \ I \ j_w \ k_w \ P_t \ T_{W-t} = 0 \ 0 \ 0 \ 1 \quad (5.10)$$

where iw_{-1} , jW_i , and kW_i are 3×1 vectors expressing the components of (i_1, j_1, k_1) in the world system of coordinates, and P_i is the position vector from O_w to 0_1 .

The matrix expressed in Equation (5.10) is fixed as long as the virtual prism is stationary. However, if object 1 moves, then the transformation becomes a function of time, as in

$$iw_{-}(t) \ jwF_i(t) \ kw_i(t) \ Pi(t) \ Tw_i(t) = 0 \ 0 \ 0 \ 1 \quad (5.11)$$

The position P_{if_i} of any vertex V_i in the system (ii, j, k_1) is known from the 3D model file of object 1. When the virtual prism moves, the changing position of vertex $V_i(t)$ in the world system of coordinates is then given by

$$Vi'w'(t) = Tw_i(t)Vi(1) \quad (5.12)$$

where $i = 1, \dots, n$ (the total number of vertices in the object file).

Figure 5.15b shows the particular case when object 1 is translated. In this case the only element changing in $Tw_i(t)$ is the position vector $P_i(t)$. Thus $Vi(w)(t)$ is given by the simplified equation

$$1 \ 0 \ 0 \ pi(t) \ Vilw1(t) = 0 \ 1 \ 0 \ p1j(t) \ Vi(1) \ 0 \ 0 \ 1 \ Plk(t) \ 0 \ 0 \ 0 \ 1 \quad (5.13)$$

Note that the $R3 \times 3$ rotation submatrix is an identity matrix because the object system and world system of coordinates are aligned in this example. By repeating the same transformation to all vertices, the object is translated such that it has a different position at time t_2 . Translating the object back to its original position is done by simply multiplying the result of Equation (5.13) by the inverse of that homogeneous transformation matrix.

When populating the virtual world it is sometimes necessary to scale objects. Scaling is in fact realized by a sequence of three operations: (1) the object is translated to the origin of the world system of coordinates, (2) it is scaled, and (3) it is translated back to its original position. Translations are done by multiplying with homogeneous transformation matrices of the type given in the previous equation, while scaling is given by

$$S_i \ 0 \ 0 \ 0 \ T_{scaling} \ 0 \ s_j \ 0 \ 0 = 0 \ 0 \ s_k \ 0 \ L \ 0 \ 0 \ 0 \ 1 \quad (5.14)$$

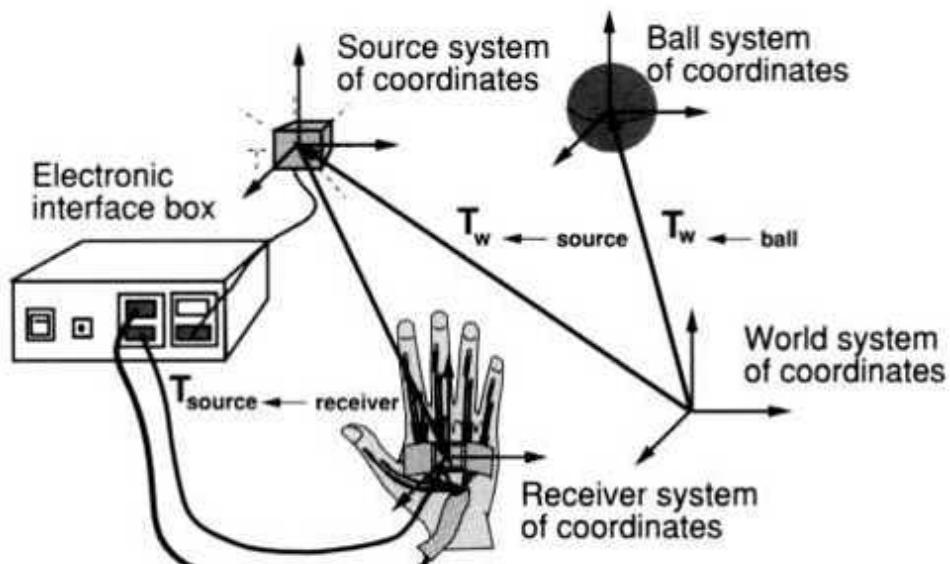
where s_i , s_j , and s_k are scaling factors about the three axes of the world system of coordinates. If scaling is uniform, then the three factors are equal.

5.2.3 Transformation Invariants

Many times one of the objects seen in the virtual scene is a 3D hand. Its position and orientation are mapped to the position and orientation of the 3D tracker attached to the sensing glove worn by the user. As illustrated in Figure 5.16a, the tracker receiver's 3D position relative to the source is given by the time-dependent homogeneous transformation matrix $T_{source-receiver}(t)$. Assuming that the source is fixed, then its position in the world system of coordinates is given by the fixed matrix $T_{w-source}$. Thus a virtual hand can be moved in the virtual world by multiplying all its vertices with a compound matrix as in

$$V_i W(t) = T_{w-source} T_{source-receiver}(t) V_i \text{hand} \quad (5.15)$$

where $V_i \text{hand}$ are the vertex coordinates in the hand system of coordinates, and $i = 1, , n$.



(a)

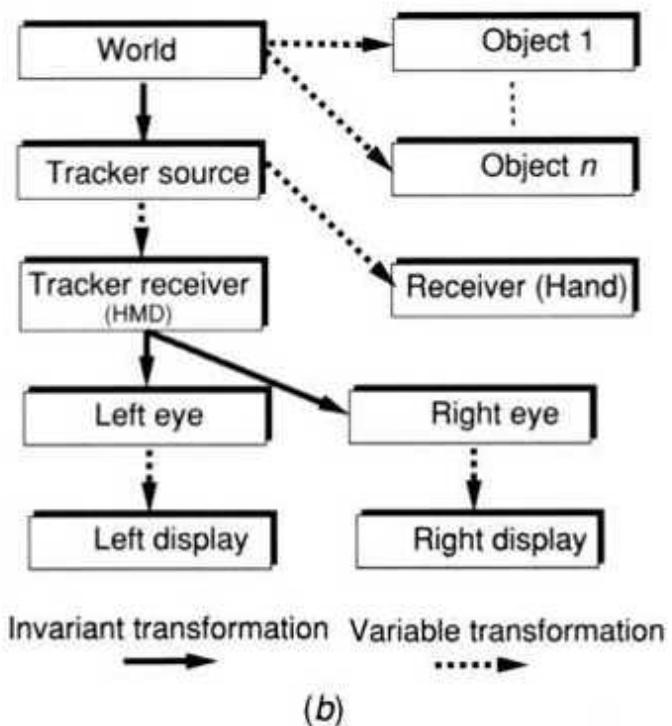


Fig. 5.16 Coordinate systems for a single user: (a) sensing glove and virtual ball; (b) viewing the world with an HMD. Adapted from Robinett and Hollaway [1992]. © 1992 ACM Inc. Reprinted by permission.

Figure 5.16a shows that the same world contains a virtual ball, and its 3D position is given by T_w -ball . If the virtual hand interacts with the ball, then

we need their relative position in the virtual world. Assuming that the hand is open and slaps the ball with the palm, then we need the transformation Treceiver*-ball(t) [Burdea et al., 1993]:

$$\text{Treceiver-ball (t)} = \text{Treceiver-w(t)} \text{Tw-ball (t)} \quad (5.16)$$

$$\text{Treceiver-ball (t)} = (\text{Tw-receiver (t)})^{-1} \text{Tw-ball (t)} \quad (5.17)$$

Once the ball is grasped, its position does not change with respect to the receiver (assuming a power grasp), and thus Treceiver±ball is invariant (until released). Invariant here means that the transformation matrix is not a function of time. It remains unchanged from frame to frame in the simulation [Robinett and Holloway, 1992]. In that case the ball's position in the virtual world is given by

$$\text{Tw.ball(t)} = \text{Tw -source Tsource -receiver (t)} \text{Treceiver -ball} \quad (5.18)$$

A virtual hand can be used for other actions besides slapping a ball. Objects can be grasped and moved to new locations. Flying in VR is also possible by having the computer understand the user's pointing gesture. Steering the view of the scene in the desired direction means changing the hand orientation. For travel within large virtual worlds it is useful to shrink the world until the final destination is within arm's reach. Then the world is expanded so as to arrive precisely at the desired destination. Robinett and Holloway [1992] analyzed these fundamental manually controlled actions for a single-user immersive system and tabulated them, as shown in Table 5.2. Interested readers may also consult Sturman [1992] for a detailed discussion on manual wholehand-input control of VR simulations.

If the user wears an HMD, then additional transformation matrices exist. Here the transformations between the HMD tracker receiver and right and left eyes are fixed. Similarly, the transformation between the room and the HMD tracker source is also fixed (as for the hand). Figure 5.16b is a graph where nodes represent systems of coordinates and edges stand for transformation matrices between two coordinate systems. The graph makes explicit the time-dependent and invariant transformations, which express the simulation kinematics.

5.2.4 Object Hierarchies

In the previous discussion we assumed that the object model is monolithic. This simplification implies that the virtual hand, for example, could not move its fingers independently, and no mention was made of the user's sensing glove readings. In order to allow such motion the hand 3D model has to be segmented. Such segmentation is the basis of object hierarchies within the virtual world.

TABLE 5.2. Fundamental Manual Control Actions in VR^a

Transformation	User	Object
Translate	Fly through the virtual world	Grasp and move the object
Rotate	Tilt the world	Grasp and turn the object
Scale	Expand or shrink the world	Scale the object

^a Adapted from Robinett and Holloway [1992]. © 1992 ACM Inc. Reprinted by permission.

Definition Object hierarchies define groups of objects which move together as a whole, but whose parts can also move independently.

A hierarchy implies at least two levels of virtual objects. The higher level objects are called the parent objects and the lower level is formed of child objects [Foley et al., 1990]. The motion of a parent object is replicated by that of all its children. However, a child object can move without affecting the parent object's position. Child objects are usually replicated several times (they have multiple instances) in the hierarchy.

The parent-child hierarchy is mathematically described by the same kind of homogeneous transformation matrices previously described. The position of a matrix in the matrix product equation is related to the hierarchical level it represents. For example, the matrix $T_{chi} \cdot Id \sim T_{grandchild}$ is to the right of the matrix $T_{parent} \cdot F_{child} \cdot Id$. Graphically the segmented model hierarchy can be represented by a tree graph. Its nodes are object segments and its branches represent relationships. A node in the tree graph is parent to all nodes in the lower branches emanating from it.

The power of hierarchical structures stems from their support by most graphics libraries through message passing. Geometric transforms applied to a given parent object are automatically passed on to all its children. At the top of the tree graph stands a global transformation that determines the view to the scene. If this matrix is changed, all objects in the virtual world will appear translated, rotated, or scaled. It is possible to change the global transformation matrix interactively with data from an I/input device such as a trackball and thus fly around the virtual world.

Figure 5.17a shows a model of a virtual football player contained in the Viewpoint database. The database model is not dynamic, so in order to allow limb motion, the model has to be segmented. The number of resultant hierarchy levels depends on how accurate the model kinematics needs to be. In our example, we are not interested in the motion of his head or hands. Thus the head and torso form the single upper body segment. Similarly, each lower arm and hand form a single lower arm segment. Since the motion of the upper body affects that of the arms, the upper body segment is parent to two upper arm segments, each being parent to a lower arm segment. Similarly, the lower body segment is parent to two upper leg segments, each of which is parent to a lower leg segment. The resulting tree-graph hierarchy of the virtual football player is illustrated in Figure 5.17b. The relative positions in the tree graph indicate inclusions. The upper body node is above the arm node, therefore the arm object is included in the upper body, and so on.

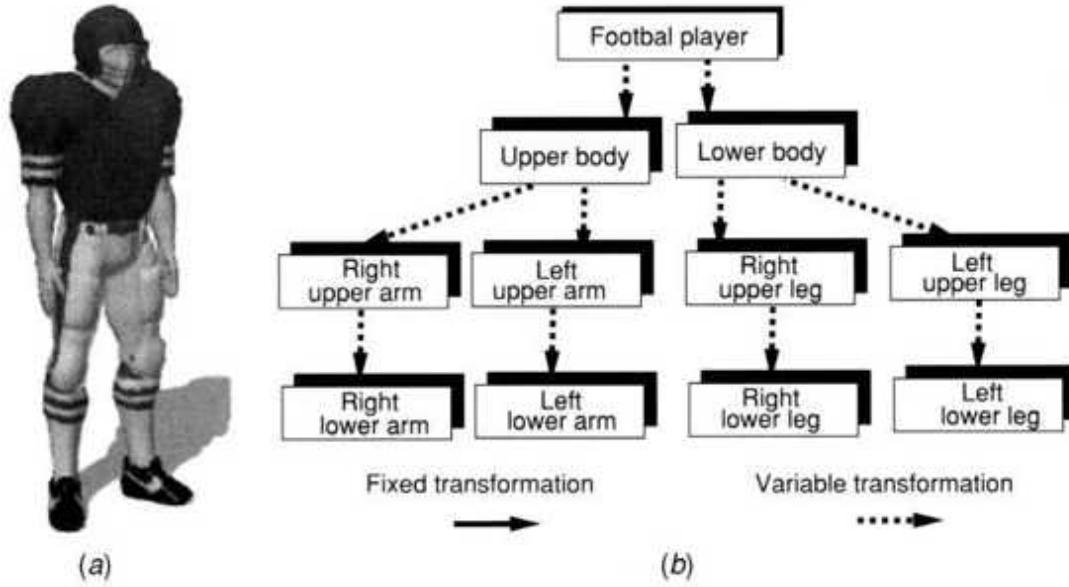


Fig. 5.17 The hierarchical structure of a football player avatar. (a) 3D monolithic model. From Viewpoint Inc. [2002b]. Reprinted by permission. (b) Hierarchical tree-graph structure of the segmented model. Adapted from Foley et al. [1990]. © Pearson Education Inc. Reprinted by permission.

Most important in many VR applications is the ability to animate a virtual hand. Its hierarchy has a palm parent and five finger children. Thus, when the palm moves (according to the tracker data), so do its children (fingers). To implement a grasping gesture, it is further necessary to subdivide each finger into substructures. The finger becomes parent to the first finger link, which becomes parent to the interphalangeal link, which in turn is parent to the last (distal) link, as illustrated in Figure 5.18. Sensing glove data can then be used to animate the virtual hand by changing the position of the finger segments. This is done by changing $T_1(t)$ between subsequent nodes in the hand tree-graph structure. It is thus possible to determine fingertip positions in the world system of coordinates, as in

$$T_w \leftarrow \text{fingertip}(t) = T_w \cdot \text{source } T_{\text{source}} \cdot \text{receiver}(t) \cdot T_{\text{receiver}} \leftarrow I(t) \cdot T_{1F2}(t) \cdot T_{2_3}(t) \cdot T_{3e} \cdot \text{fingertip} \quad (5.19)$$

This equation makes explicit the time dependences that exist, in order to show that the T_3 -fingertip is an invariant. This transformation depends only

on the finger dimensions, not on time. To show the virtual hand we need to include the global transformation matrix (parent) as in

$$T_{\text{global}} * \text{fingertip}(t) = T_{\text{global-w}}(t) T_w * \text{fingertip}(t) \quad (5.20)$$

Note that the global transformation $T_{\text{global-w}}(t)$ is time-dependent, since it is controlled by the user, who changes the view to the simulation interactively. Alternatively, this matrix can be changed by an algorithm that flies the virtual camera on a predefined path through the scene.

5.2.5 Viewing the Three-Dimensional World

The inverse of the matrix $T_{\text{global-w}}(t)$ in Equation (5.20) is the view transformation. It represents the position and orientation of the camera system of coordinates in the (fixed) world system of coordinates. The camera system of coordinates in OpenGL is left-handed, unlike the other Cartesian system of coordinates associated with model transforms [Moller and Haines, 1999]. The camera is oriented such that it is looking in the positive z direction, with the y axis pointing upward and the x axis to the right, as seen in Figure 5.18.

Mapping the virtual objects to camera coordinates (the so-called eye space) is the first task in the geometry stage of the rendering pipeline. This is followed by lighting (previously discussed), perspective projection, clipping, and screen mapping. These geometry substages are discussed next.

The graphics pipeline is not concerned with the entirety of the virtual world, rather it processes only what the camera sees. This represents a volume of space called a frustum, which is a portion of a pyramid aligned with the camera system of coordinates. As illustrated in Figure 5.19a, the tip of the frustum is at the origin of the camera system of coordinates, also called the center of projection. Lines from the center of projection to the 3D object vertices intersect the viewing plane to form the object's perspective projection. The size of the projection is inversely proportional to the distance from the object to the center of projection (distant objects appear small, closer ones large). The frustum pyramid is further defined by a near and a far rectangle. The rectangle dimensions are given by the parameters

$(1, r, b, t, n, f)$, which determine how large the viewing angle is and at what distance objects are seen.

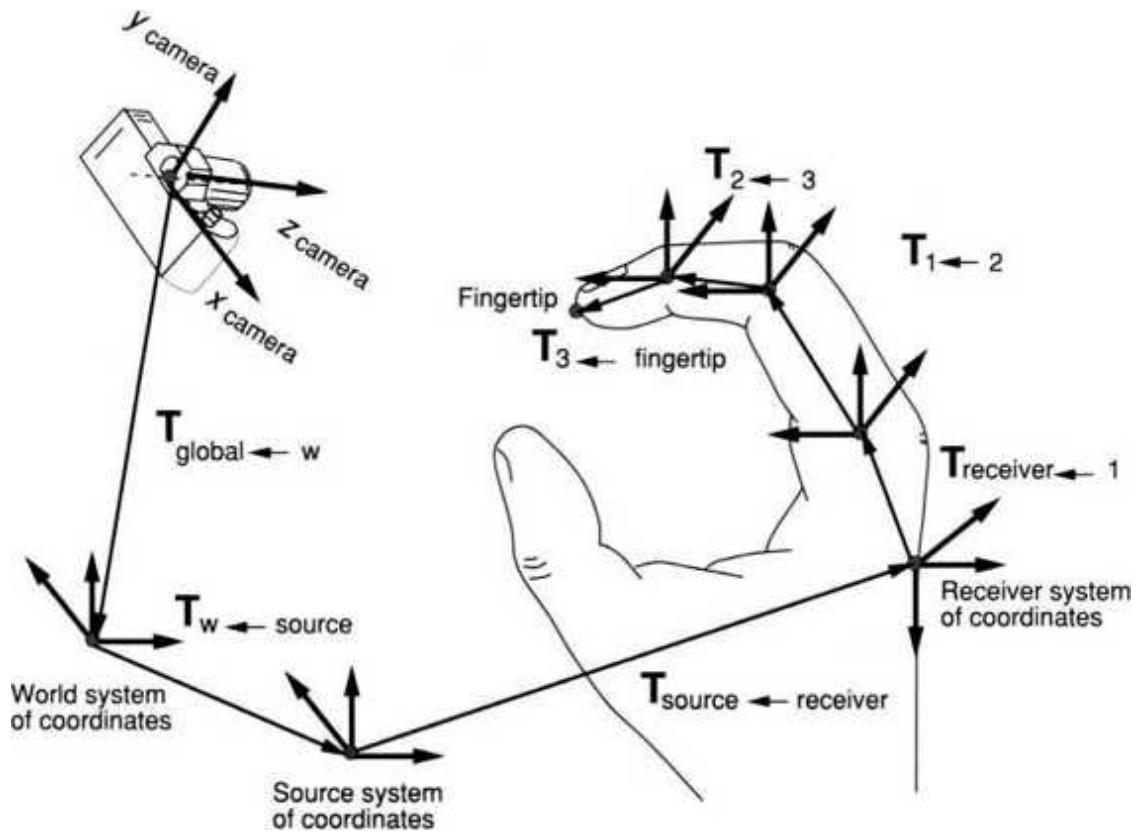


Fig. 5.18 The hierarchical structure of a virtual hand as seen by a virtual camera.

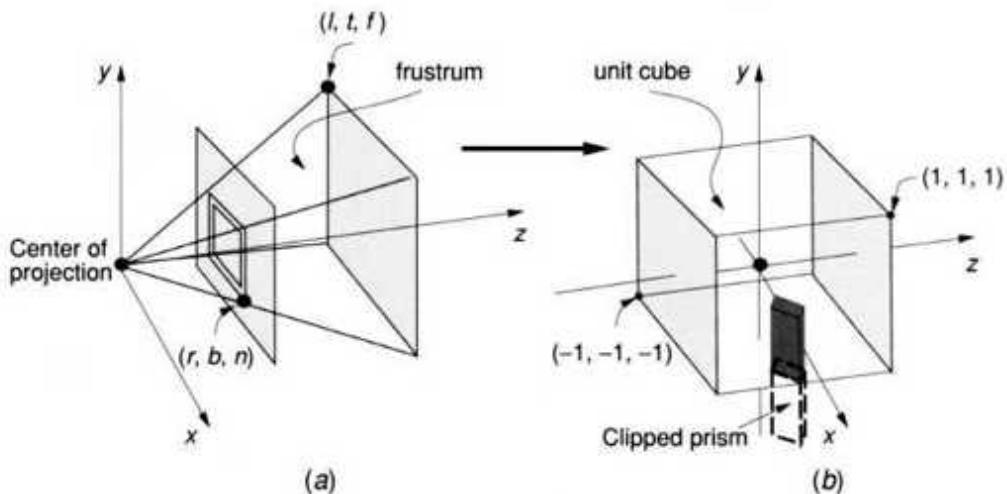


Fig. 5.19 Viewing the 3D World: (a) the frustum; (b) the canonical view model.

To optimize processing, the geometry stage of the graphics pipeline maps the frustum to a canonical view volume. This is a unit cube with corners at $(-1, -1, -1)$ and $(1, 1, 1)$, as illustrated in Figure 5.19b. The mapping is done by the following projection matrix [Moller and Haines, 1999]:

$$2n \ 0 \ r+1 \ 0 \ r-1 \ r-1 \ 0 \ 2n \ t+b \ 0 \ T_{\text{projection}} = t - b \ t - b \ 0 \ 0 \ _- f + n - 2f_n \ f - n \ f - n \ 0 \\ 0 \ -1 \ 0 \ (5.21)$$

Once the 3D objects are projected into the canonical cube, their coordinates are normalized, which helps in the Z-buffering process. Objects with higher z values are further away, and should not be rendered if occluded by objects with smaller z. Furthermore, objects will be clipped such that only the portion of an object inside the cube will be sent down the graphics pipeline.

The last stage of geometry processing is screen mapping. This stage translates and scales the (x, y) vertex coordinates of the objects inside the cube to the dimensions of a 2D display window. If the window is rectangular, the scaling will be nonuniform. The vertex z coordinate is not affected by screen mapping, as it is sent directly to the Z-buffer.

5.3 PHYSICAL MODELING

The next step in virtual world modeling, following geometry and kinematics, is the integration of the object's physical characteristics. These include weight, inertia, surface roughness, compliance (hard or soft), deformation mode (elastic or plastic), etc. These features, together with object behavior (discussed later in this chapter), bring more realism to the virtual world model. The computation load required by physical modeling is assigned to the haptics rendering pipeline, as detailed in Chapter 4. Therefore our discussion of physical modeling follows the stages of this pipeline, namely collision detection, force calculation, force smoothing, force mapping, and haptic texturing.

5.3.1 Collision Detection

The first stage of haptic rendering (and thus physical modeling) is collision detection, which determines whether two (or more) objects are in contact with each other. This can be considered a form of haptic clipping since only objects that collide are processed by the haptics rendering pipeline.

Collision detection can be classified into approximate and exact [Lin et al., 1996]. Approximate collision detection, also called bounding-box collision detection, uses 3D extents (bounding boxes), as illustrated in Figure 5.20a.

Definition A bounding box is a prism which encloses all the vertices of a given 3D object.

Bounding boxes are classified into oriented and axis-aligned. An oriented bounding box (OBB) is a prism aligned with the object's major axes and which changes orientation dynamically as the object rotates. An axis-aligned bounding box (AABB) is a bounding box aligned with the world system of coordinates. A special case of an AABB is a fixed-size cube, which can accommodate any orientation of the 3D object it encloses. Fixed-size bounding boxes are computationally faster than variable-size bounding boxes, but also tend to be larger than these.

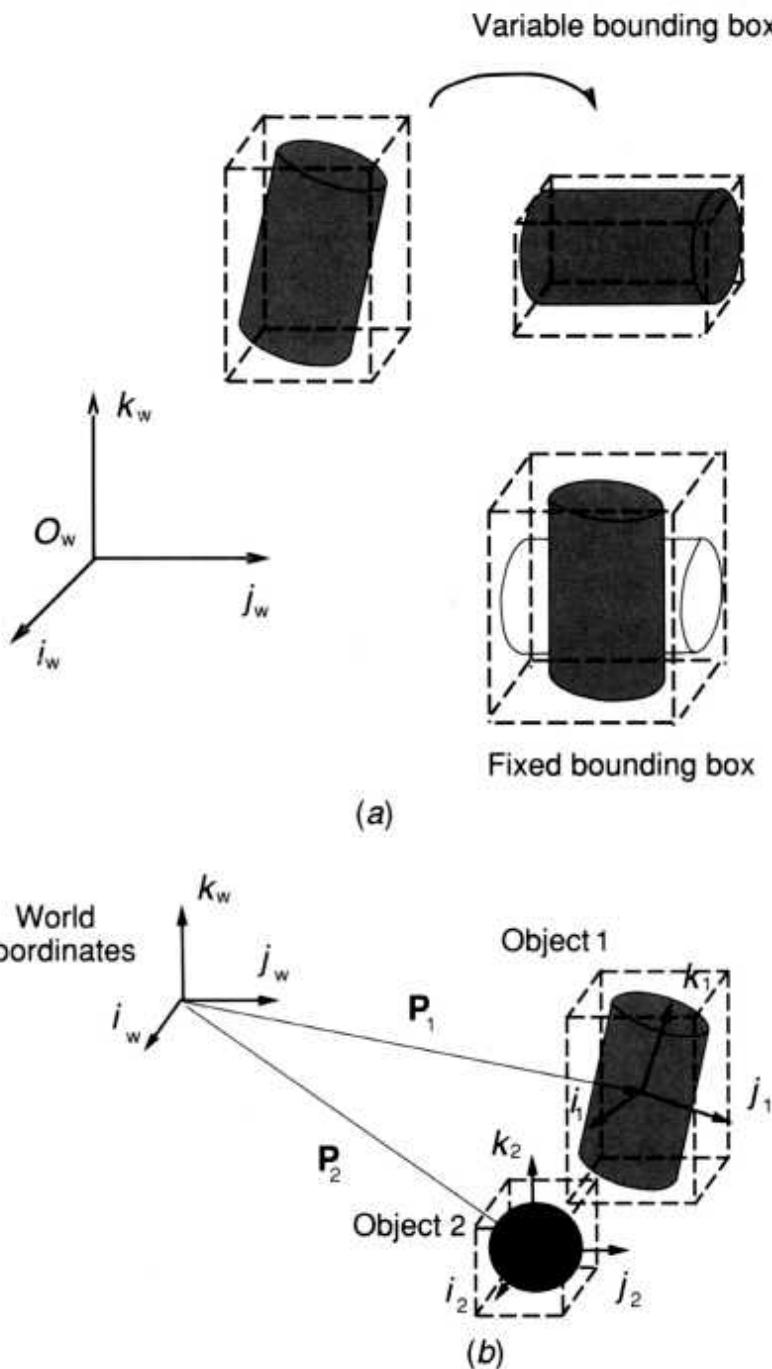


Fig. 5.20 Bounding-box collision detection: (a) types of axis-aligned bounding boxes; (b) checking for collision.

Bounding-box collision detection tests whether two (or more) bounding boxes overlap in 3D space. As illustrated in Figure 5.20b, if two objects are in close proximity to each other, their bounding boxes overlap (they collide).

Testing for collision means projecting their bounding boxes along the axes of the world system of coordinates and determining whether all the three projection intervals overlap. Since the AABBs are already aligned with the world system of coordinates, their projection is computationally faster than for OBBs. However OBBs are a tighter fit to the objects they enclose, so OBB-based collision detection is more accurate.

Bounding-box collision detection needs to be fast in order to detect all collisions. We recall that the first stage of the haptics rendering pipeline is done in software by the CPU. Thus the CPU has to perform collision detection at every frame, which is computationally intensive. The more objects exist, the more costly the collision detection is to the VR engine computational resources. In the worst-case scenario, there are fast-moving objects in a cluttered virtual world. In this case, the CPU may miss a collision entirely if the following condition applies:

$$1 \text{ d} < v - \text{fps} \quad (5.22)$$

where d is the largest distance between two bounding boxes along the trajectory of the fast-moving object, v is the relative velocity of the moving object, and fps is the rendering speed of the graphics pipeline (frames/sec).

A way to alleviate the problem of undetected collisions (when objects sail through each other) is to increase the rendering speed (larger fps), which means upgrading the pipeline hardware. Another approach is to take advantage of scene coherence when performing the bounding-box overlap testing. Scene coherence means that, at any given time, a moving object is located in the neighborhood of the position it had in the previous frame. In other words, the scene does not change too much from frame to frame. Scene coherence lends itself well to collision detection optimization algorithms. Specifically, all fixed-size bounding boxes in the virtual world can be projected on the three axes of the world system of coordinates. The three resulting interval lists can be sorted at the start of the simulation and overlap pairs created. At subsequent frames the sorted lists are traversed looking for overlaps. The scene coherence makes the interval lists change little from

frame to frame, such that collision detection cost is proportional to the number of objects in the world, $O(n)$.

Approximate collision detection is a necessary first step, designed to weed out objects that are far apart, such that their bounding boxes do not overlap. If the VR application needs to know exactly where the two objects are in contact, then an exact collision detection needs to be performed. The two types of collision detection are performed at each simulation cycle, as illustrated in Figure 5.21 [Cohen et al., 1995]. Exact collision detection is performed only on the object pairs that are potentially in contact (according to the AABB collision detection test).

The performance of exact collision detection algorithms is usually degraded in proportion to the complexity of the virtual scene. Lin [1993] developed a fast algorithm that has almost constant performance versus the number of object vertices. For most objects, including spherical ones, the runtime is roughly $O(nn)$, where n is the number of vertices in each object. The algorithm uses local features (such as vertices, edges, or facets) of convex polyhedra and is extendable to concave ones. The tests use Voronoi volumes defined by planes extending the facets adjacent to the one of interest. The internal Voronoi volume is defined by the prism having the facet as base and the apex at the centroid of the object. The collision test is then performed for the closest features of two objects, provided that they fall in the external Voronoi volume. Once a vertex is inside the internal Voronoi volume a collision has occurred. When any facet has more than five edges a preprocessing stage is added in order to divide its external Voronoi volume into subvolumes (each resting on a facet with fewer than five edges). This is needed to decouple the collision detection computation time from the number of vertices. Cohen and his colleagues at the University of North Carolina at Chapel Hill [Cohen et al., 1995] extended Lin's local collision detection algorithm to multibody collision detection for VR simulations. In this way the number of possible interactions is trimmed from $O(n^2)$ to $O(n + m)$, where n is the number of simultaneously moving objects and m is the number of objects which are very close to each other.

Another exact collision detection method was proposed by Schlaroff and Pentland [1991] at the MIT Media Laboratory. Instead of using a polygon representation for an object shape, they used implicit functions that algebraically define the object surface. These superquadrics of the form $f(x) = 0$ have an inside-outside feature which can detect whether a point is inside or outside of the volume bounded by the implicit function. All points $x = (x, y, z)$ for which $f(x) < 0$ are inside this volume, while points outside the volume correspond to $f(x) > 0$. Thus, when a vertex P satisfies $f(P) < 0$, collision has occurred. The resulting algorithm has a complexity of $O(m)$, where m is the number of collision tests (points).

5.3.2 Surface Deformation

Collision detection is followed by collision response, which depends on the characteristics of the virtual objects in contact and on the particular application being developed. If the objects in contact are nonrigid, then one form of collision response is surface deformation. Surface deformation changes the 3D object geometry interactively and thus needs to be coordinated with the graphics pipeline.

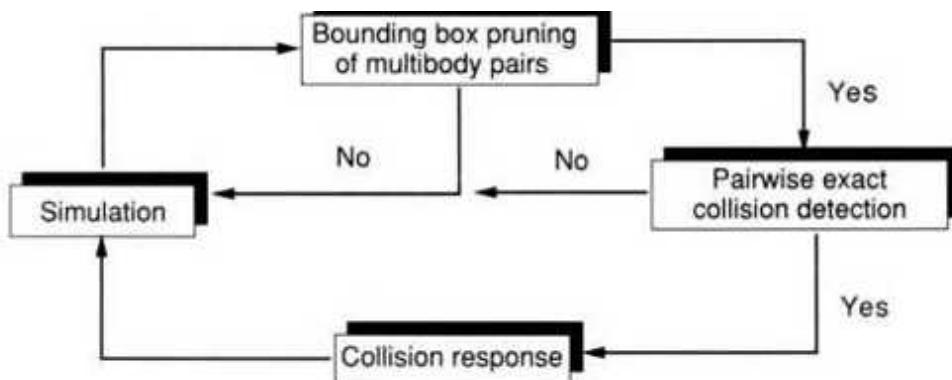


Fig. 5.21 Multibody collision detection algorithm. Adapted from Cohen et al. [1995]. © 1995 ACM Inc. Reprinted by permission.

If the object is modeled by parametric surfaces, then the deformation is done indirectly by modifying the position of control points surrounding the surface. The indirect surface modification is difficult because users are less familiar with splines. Furthermore, control points are difficult to manipulate when occluded by other objects. Hsu developed a direct free-form

deformation (DFFD) approach, which allows the user to select a point on the object's parametric surface, then move a pointer to where the new location of that point should be [Hsu et al., 1992]. The algorithm then calculates the necessary changes to the control point lattice that would induce the desired change in surface shape. The problem is underdetermined since there are many lattice configurations that would yield the same deformation location for the selected surface point. Thus a least-squares solution is used to select among possible control lattice candidates.

If the object is modeled using polygons, then surface deformation is done directly, through vertex manipulation. The user can interactively change the location of a vertex in the look-up table, thus redefining the shape of the polygons sharing it during image rendering. General graphics libraries such as Starbase [Hewlett-Packard, 1991] provide connectivity information in the look-up table containing the object's polygonal mesh. The user can thus modify a vertex and its neighbors according to some application-dependent deformation propagation law. An example of surface deformation when a virtual hand squeezes an elastic virtual ball is discussed later in this chapter.

An extreme case of surface deformation is surface cutting. Unlike the previous examples, this form of collision response results in surface topological changes. Figure 5.22 shows a top view of a (sharp) moving object contacting the surface polygonal mesh of a stationary object [Song and Reddy, 1995]. The vertex closest to the contact location is replaced by twin vertices that are initially colocated. To simulate the surface cutting, these twin vertices retract gradually. The process is repeated with new twin vertices being created along the cutting path.

5.3.3 Force Computation

When users interact with 3D object surfaces (whether these are deformable or not), they should feel reaction forces. These forces need to be computed by the haptics rendering pipeline and sent to the haptic display providing force feedback to the user.

Force computation takes into account the type of surface contact, the kind of surface deformation, as well as the object's physical and kinematic

characteristics. The simplest surface contact is a single-point contact, which is the model implemented by haptic interfaces such as the PHANToM. Haptic gloves involve multiple contacts with a single (grasped) object. Furthermore, rigid bodies that are immobile (such as walls) behave differently than elastic objects that are mobile (such as rubber balls). These in turn behave differently than plastically deformed objects (such as soda cans), and so on. Thus our discussion on force computation is structured in accord with the type of object being haptically simulated.

5.3.3.1 Elastic Virtual Objects. These objects have 3D surfaces that deform proportionally to the magnitude of the contact (or constraint) force. Conversely, as the contact force diminishes, the geometry of an elastic virtual object returns to its original shape, as illustrated in Figure 5.23 [Burdea et al., 1992]. Modeling such interactions is governed by Hooke's law:

$$J \cdot K_{object} \cdot d \text{ if } 0 < d < d_{max} \\ F_{constraint} = F_{max} \text{ if } d_{max} < d \quad (5.23)$$

where K_{object} is the object stiffness and d is the amount of surface deformation at the point of contact. For a given value of d , harder objects (with large K_{object}) will create larger feedback forces than softer objects. Eventually, the modeled force will reach the maximum force capability of the haptic interface, which corresponds to the surface deformation d_{max} . For larger deformations Hooke's law does not apply, since the haptic interface has saturated, and the force stays constant. For the same haptic interface, softer elastic objects will have a larger d_{max} . Owing to the object elasticity, feedback forces are present during both surface compression and relaxation (moving away from the object) as long as d is not zero.

The example illustrated in Figure 5.23 assumed that the elastic object is homogeneous, and thus has a constant K_{object} . There are cases when the elastic object is not homogeneous, having, for example, a softer exterior and a harder interior. In this case the force-surface deformation dependence can be approximated by the piecewise linear relationship given by

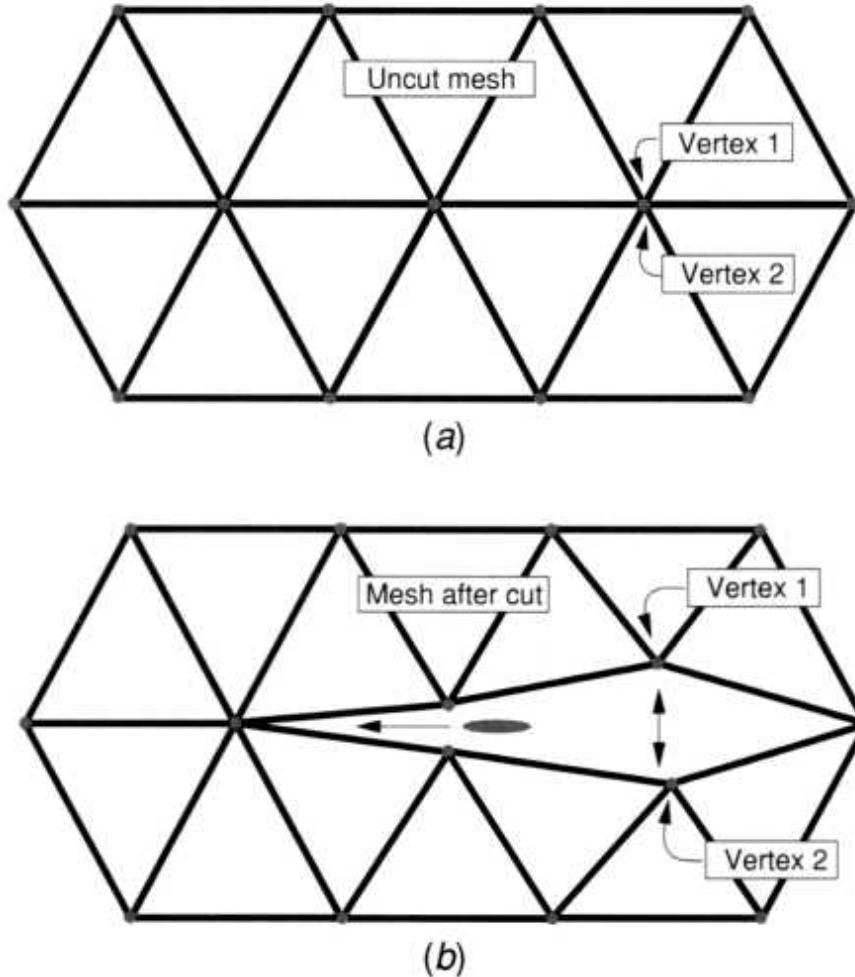


Fig. 5.22 Top view of a surface polygonal mesh being cut: (a) twin vertices before the cut; (b) vertices being retracted along the cutting path. Adapted from Song and Reddy [1995]. Reprinted by permission.

$$F_{\text{constraint}} = \begin{cases} K_1 \cdot d & \text{if } 0 \leq d \leq d_{\text{disc}} \\ K_1 \cdot d_{\text{disc}} + K_2 \cdot (d - d_{\text{disc}}) & \text{if } d_{\text{disc}} \leq d_{\text{max}} \\ F_{\text{max}} & \text{if } d_{\text{max}} < d \end{cases} \quad (5.24)$$

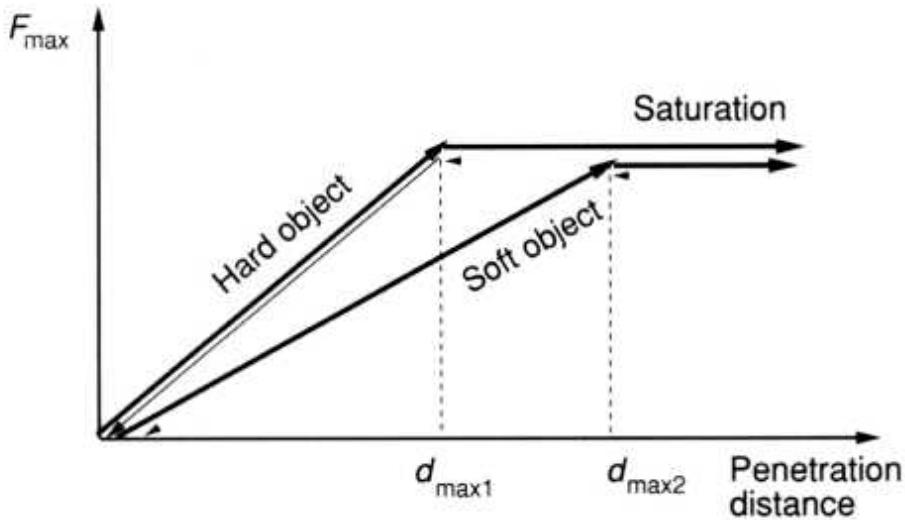


Fig. 5.23 Contact forces present during elastic object deformation. Adapted from Burdea et al. [1992]. Reprinted by permission.

where d_{ds} is the amount of surface deformation at which the influence of the harder interior starts to be felt. As seen in Figure 5.24, this marks a discontinuity in the slope of the feedback force curve. Subsequently, contact forces grow faster as long as the haptic interface is not saturated.

Another example of 3D objects with nonuniform hardness is virtual pushbuttons. These model the tactile feel of real pushbuttons commonly found in control panels and thus are very useful in haptically enhanced training simulations. Figure 5.25a illustrates a virtual hand pressing one of two pushbuttons included in a virtual control panel. At the beginning the user feels an increasing resistance, which models the compression of a return spring inside the button. When the displacement of the push button reaches a given value r_n the spring is disengaged and the force drops to a very small value F_{residual} , due to static friction. If the user continues to press against the pushbutton past the threshold n , the feedback force grows very quickly to the maximum value allowed by the haptic interface. This signifies that the button has reached its linear travel limit and is now pushing against the supporting back plane. The contact force profile shown in Figure 5.25b is expressed as

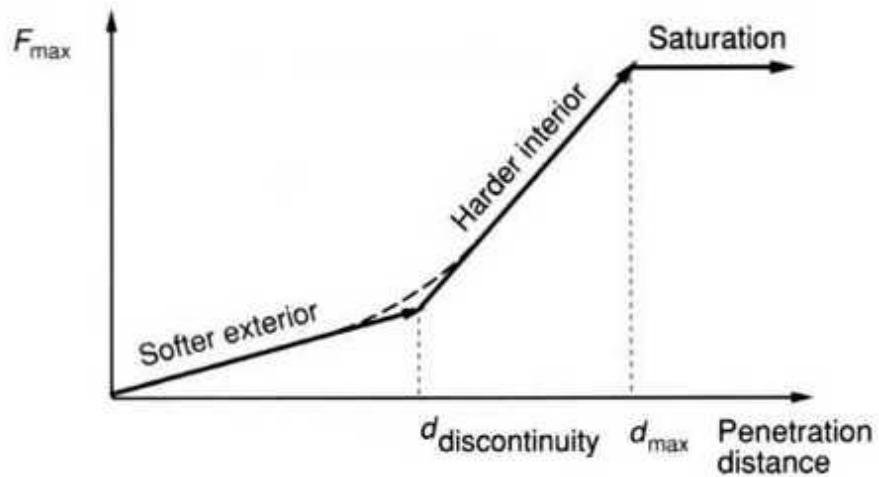


Fig. 5.24 Feedback forces for a nonhomogeneous elastic ball deformation.
Adaped from Dinsmore et al. [1997]. © 1997 IEEE. Reprinted by permission.

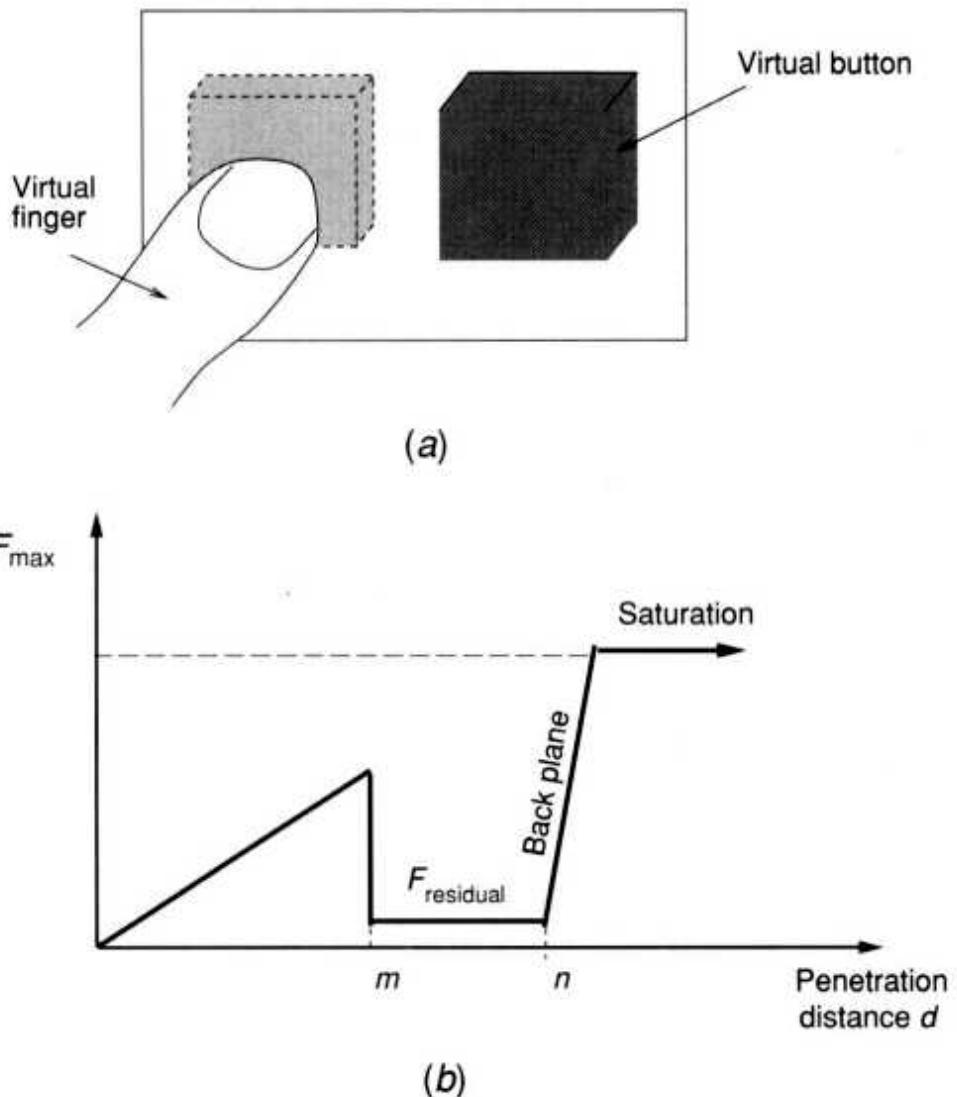


Fig. 5.25 Feedback forces for a virtual pushbutton: (a) virtual control panel; (b) the haptic click. Adapted from SensAble Devices [1993]. Reprinted by permission.

$$F_{\text{constraint}} = K_1 \cdot d \cdot (1 - u_m) + F_{\text{residual}} \cdot u_m + K_2 \cdot (d - n) \cdot u_n \quad (5.25)$$

where u_m and u_n are unit step functions that have zero value for $d < m$ and $d < n$, respectively. This force profile is designed to produce a characteristic haptic click [SensAble Devices, 1993], which confirms the user's actions.

5.3.3.2 Plastic Virtual Objects. These objects have a different force profile when deformed. The undeformed plastic object may in fact behave initially

as an elastic one, as illustrated in Figure 5.26a [Burdea et al., 1992]. However, unlike elastically deformed objects, forces are not present during relaxation, since the surface remains deformed. Thus the feedback force drops to zero as soon as contact with the deformed surface is lost, as expressed in

$$F_{\text{initial}} = I K_{\text{elastic}}' d \text{ if } 0 < d < m \\ 0 \text{ during relaxation} \quad (5.26)$$

For subsequent deformations, the force is zero until contact is reestablished (when the penetration distance from the original shape is m). After that the force $F_{\text{subsequent}}$ grows linearly until the maximum interface force level is reached. The contact force model in this case is given by

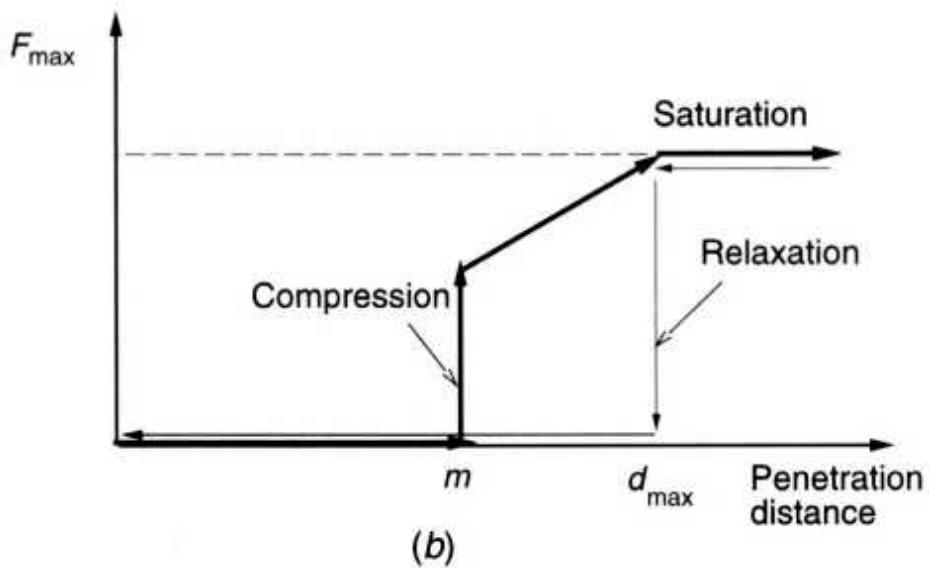
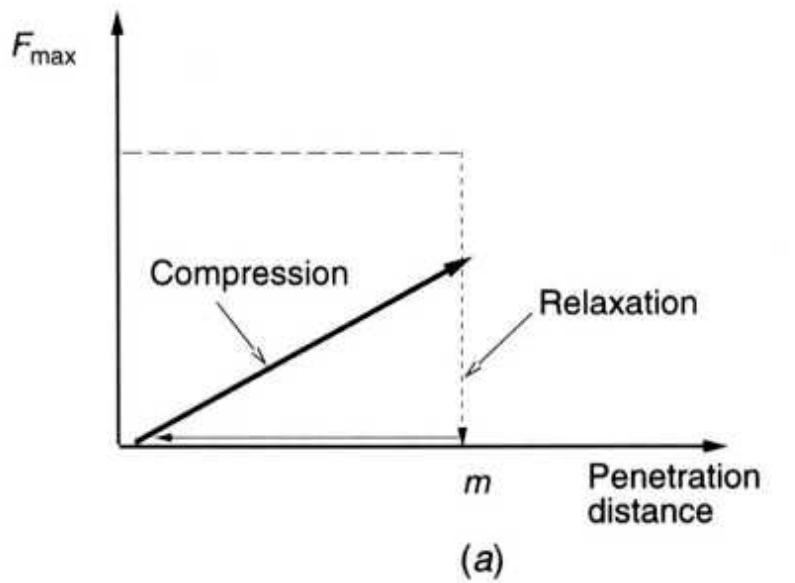


Fig. 5.26 Plastic deformation: (a) initially undeformed object; (b) initially deformed object. Adapted from Burdea et al. [1992]. Reprinted by permission.

$$F_{\text{subsequent}} = \begin{cases} K_{\text{plastic}} \cdot d \cdot u_m & \text{if } 0 \leq d \leq d_{\max} \\ F_{\max} & \text{if } d_{\max} < d \\ 0 & \text{during relaxation} \end{cases} \quad (5.27)$$

5.3.3.3 Virtual Walls. These are a special category of virtual objects that are neither elastic nor plastic. A real wall is rigid, such that it produces a large

and sudden feedback force at the instance of contact. Simulating a wall in VR is problematic, owing to the insufficient stiffness current haptic interfaces can produce. Such stiffness is on the order of thousands of N/m, whereas real walls have a stiffness an order of magnitude larger [Colgate et al., 1993].

Modeling a wall with a simple Hooke's law is affected by the digital nature of the haptic interface control. When a virtual spring is compressed, forces are held constant at the start of each control cycle. Thus the simulated forces are slightly less than those of real springs during compression. This results in a ladder profile, as illustrated in Figure 5.27. Conversely, when a virtual spring is released, the simulated forces are slightly larger than real springs would produce. The difference between the force profiles during compression and release represents energy, such that the wall becomes active and not passive (as real walls are). Such energy generation can induce instabilities in the haptic interface and unwanted oscillations at the moment of contact. In order to absorb such oscillations, we need to introduce a dissipative term, which is the damper B. As a consequence, the proportional model used previously for elastic and plastic objects becomes proportional-derivative, as given in

$$F_{\text{constraint}} = K_1 d + B \cdot \dot{d} \quad \text{for } d < 0 \quad i \quad K_2 d \quad \text{for } d > 0 \quad (5.28)$$

where \dot{d} is the derivative of the surface penetration, that is, velocity. Negative velocity corresponds to an object moving into the wall, positive velocity signifies moving away from the virtual wall.

The quality of the haptic rendering in this case is reflected in the ability of the model to create "wallness". This depends on the crispness of initial contact, the hardness of the rigid surface, and the cleanliness of final release [Rosenberg and Adelstein, 1993]. A simple damper will satisfy the crispness requirement by creating a very realistic sensation of initial contact. It will, however, be detrimental to the cleanliness of final release because it would oppose moving away from the wall. Thus the damper in Equation (5.28) needs to be a directional one, which produces no force when the velocity is positive. It is possible to add a third term to the

expression in Equation (5.28) to compensate for time delays in the sensing and control of the haptic interface. This third term is a pulse applied when crossing into the wall in order to reduce the unwanted penetration distance [Salcudean and Vlaar, 1994].

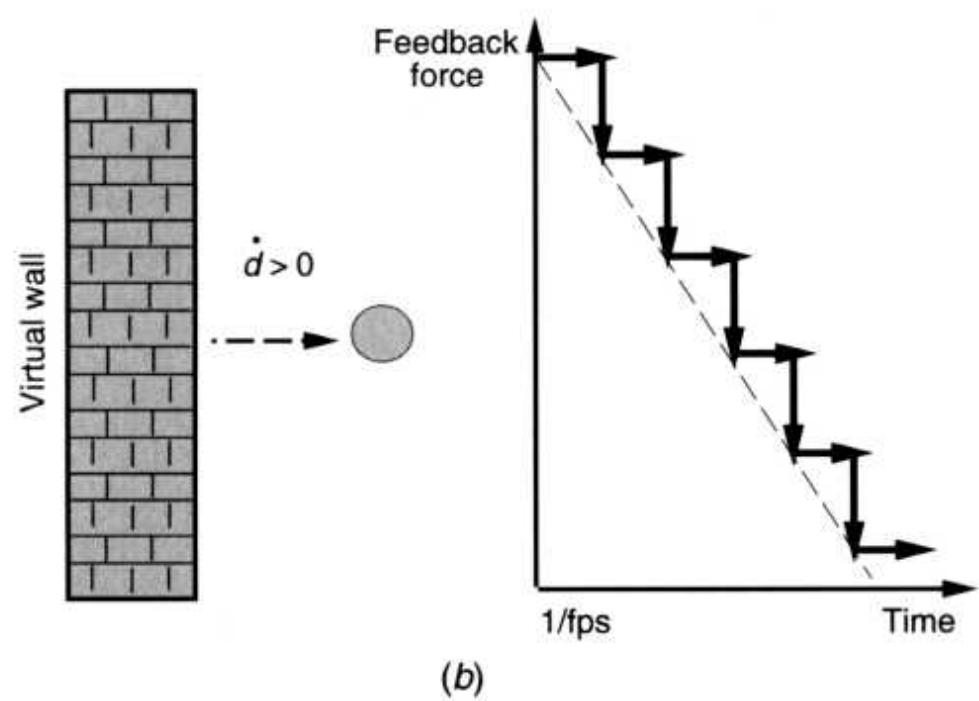
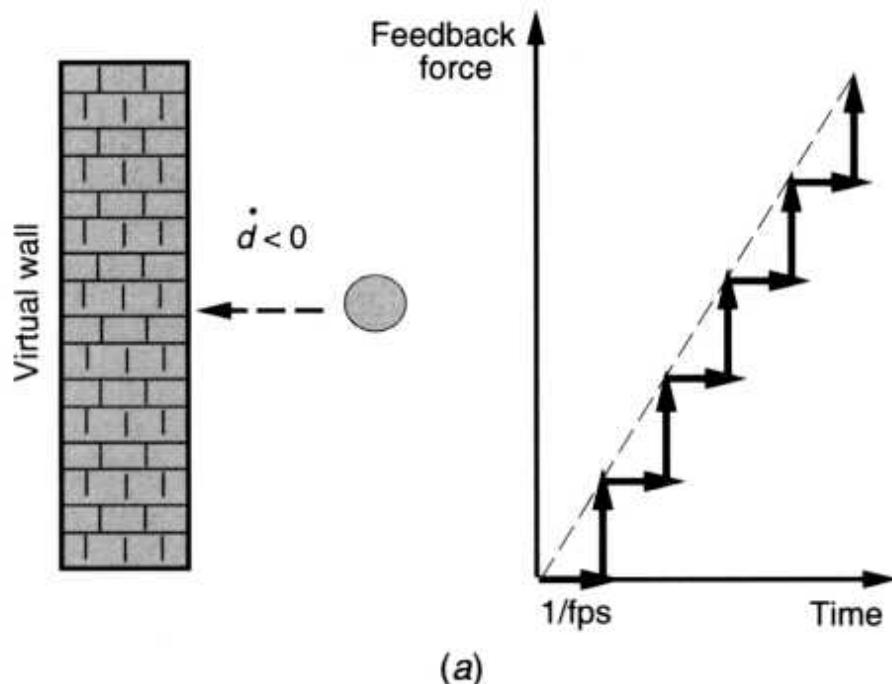


Fig. 5.27 Interaction with a virtual wall: (a) moving into the wall; (b) moving away from the wall. Adapted from Colgate et al. [1993]. © 1993 IEEE. Reprinted by permission.

5.3.4 Force Smoothing and Mapping

The foregoing discussion on contact force modeling assumed that the object surface is frictionless and that a single point of interaction exists. Under such assumptions the direction of the resistive force is that of the surface normal at the point of contact. This simplified approach creates unwanted force discontinuities for curved polygonal surfaces. This is due to the abrupt change in surface normal direction from one polygon to the next. Therefore the object will feel tessellated, or faceted, when rendered haptically, even if its graphics image looks smooth.

Earlier in this chapter we presented local illumination techniques, which used vertex and pixel normal information to render smooth polygonal surfaces. The same shading approach can be extended to the haptic domain [Morgenbesser and Srinivasan, 1996].

Definition Force shading changes the direction of the feedback force produced during interactions with polygonal surfaces in order to simulate contact with smooth curved surfaces.

The direction of the contact force is computed based on the weighted normals of the vertices of the polygon being contacted. This approach does not affect the magnitude of the contact force, only its direction. Let us assume that a cylindrical bump section is approximated by three polygons, as illustrated in Figure 5.28a. If the feedback forces are not smoothed, then the discontinuity in force direction will be clearly felt by the user whenever the edge between polygons is crossed. A real cylinder will have no such discontinuities, such that the feedback force will be oriented radially, as shown in Figure 5.28b. The force magnitude is given by Hooke's law, as previously discussed. However, once force shading is applied, the feedback force direction is changed. Thus Equation (5.23) is modified to become

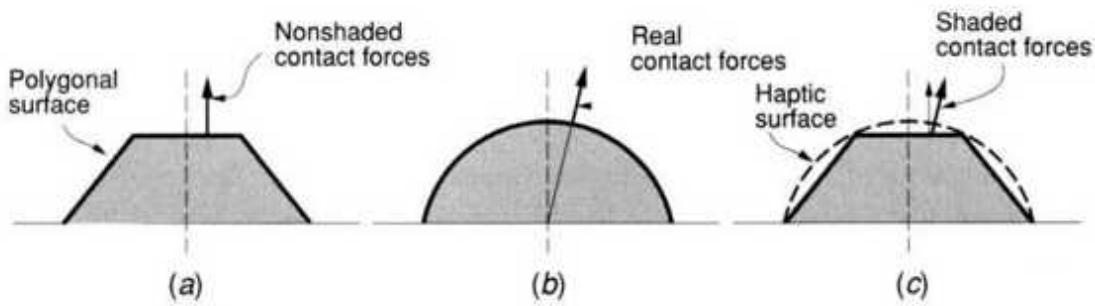


Fig. 5.28 Force shading for polygonal surfaces: (a) nonshaded contact forces; (b) real cylinder contact force direction; (c) contact forces after shading. Adapted from Morgenbesser and Srinivasan [1996]. © 1996 ASME. Reprinted by permission.

$$K_{object} \cdot d \cdot N \text{ if } 0 < d < d_{\text{max}}; F_{\text{smoothed}} \text{ if } d_{\text{max}} < d \text{ or } (5.29)$$

where N is the direction of the contact force based on interpolation of the normals at the polygon vertices.

Equation (5.29) assumes a single point of contact, which works for stylus-based interfaces, such as the PHANTOM. In this case the current location of the point to which the stylus is mapped is called the haptic interface point (HIP) [Ho et al., 1999]. The efficiency of HIP-based force shading grows with the increase in surface polygon count and the decrease in the surface curvature. A flatter cylindrical bump would be easier to reproduce using this approach.

Other haptic interfaces, such as haptic gloves, require a more detailed model of the contact geometry. A haptic glove is usually mapped to a virtual hand, thus the geometry of the virtual fingertip needs to be taken into account. In the haptic domain this corresponds to a cylindrical haptic mesh centered on the fingertip [Popescu et al., 1999]. Figure 5.29a illustrates the haptic mesh at the contact between a fingertip and a polygonal sphere. The sphere models an elastic virtual ball being squeezed by the user wearing a haptic glove. Assuming the glove has not reached saturation, then at each haptic mesh point the contact force is given by

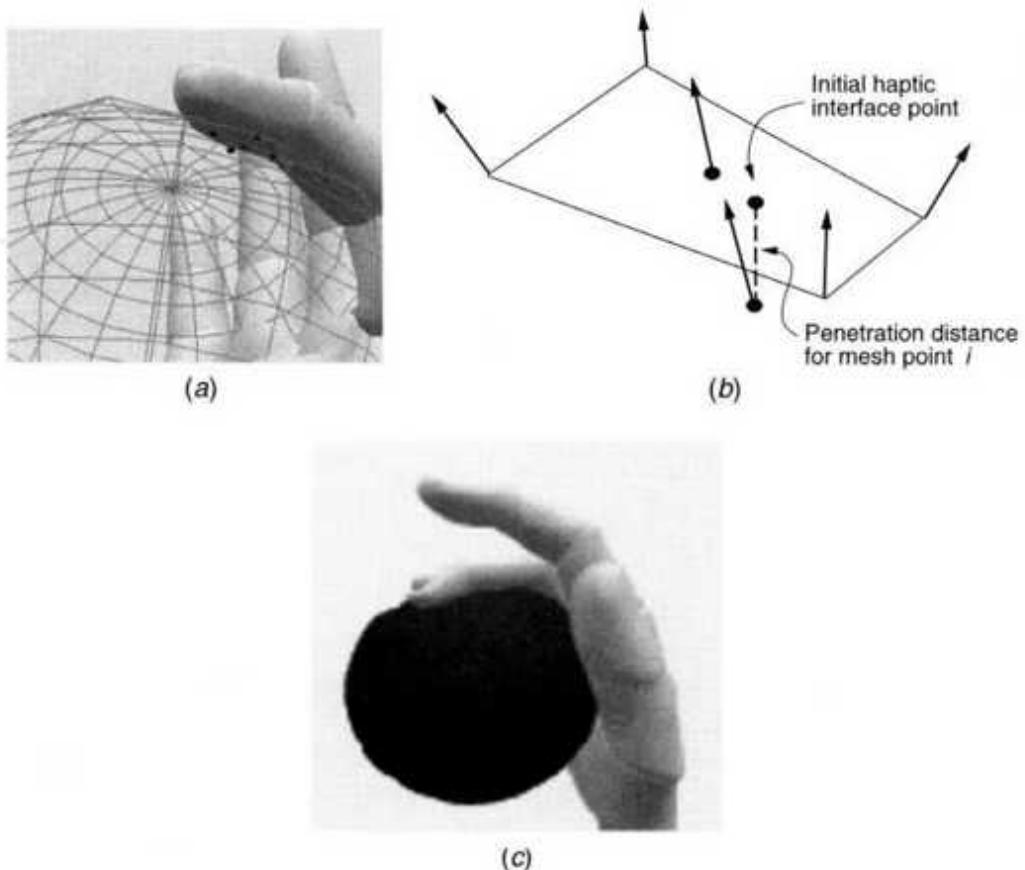
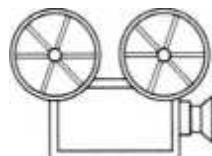


Fig. 5.29 Force smoothing using a haptic mesh: (a) surface detail; (b) force shading at contact point i ; (c) virtual ball surface deformation. From Popescu et al. [1999]. © 1999 IEEE. Reprinted by permission.

$$\text{Fhaptic mesh } i = K_{\text{ball}} \cdot d_{\text{mesh } i} \cdot N_{\text{surface } i} \quad (5.30)$$

where $d_{\text{mesh } i}$ is the penetrating distance at mesh point i . This distance is measured between the initial contact point (called "ideal haptic interface point" (IHIP) [Ho et al., 1997]) and the current location of the mesh point; $N_{\text{surface } i}$ is the weighted surface normal of the contact polygon at the contact point.



A haptic mesh models the fingertip geometry more accurately such that the resulting forces become more realistic. This needs to be coupled with a realistic surface deformation usually in an area of influence around the fingertip. This local deformation approach is illustrated in Figure 5.29c.

Commercial force feedback gloves, such as the CyberGrasp, have a single actuator per finger. Therefore the haptic mesh forces need to be summed and the resultant mapped to the corresponding actuator, as in

$$F_{\text{haptic mesh}} = \cos(\theta) \quad (5.31)$$

where θ is the angle between the haptic glove actuator and the resultant mesh force [Popescu et al., 1999].

5.3.5 Haptic Texturing

Haptic texturing is the last stage of the haptic rendering pipeline, as illustrated in Figure 4.9. Just as graphics textures add realism to the object appearance, haptic textures enhance the realism of the physical model of the object surface. Furthermore, haptic textures can add new information to characterize an object as slippery, cold, smooth, etc. Finally, haptic textures can be cascaded creating new surface effects, similar to the multitexturing approach described previously for graphics.

5.3.5.1 Haptic Textures Produced by Tactile Mice. Textures, such as those produced by the single-degree-of-freedom iFeel mouse described in Chapter 3, are quite simple [Immersion Co., 2001]. These interfaces can apply small forces in the z direction (normal to the desk surface). As a consequence, the surfaces that can be simulated are frictionless, since no resistive forces can be applied in the horizontal (x - iv) plane. The iFeel mouse can produce texture patterns of the type illustrated in Figure 5.30. These are specified by the height, the width, and the spacing between bumps. The height determines the magnitude of the vertical force produced by the mouse actuator, up to the saturation limit. Conversely, the spacing and width of the haptic ridges determine the texture frequency component felt when the mouse pointer scans the object surface. Textures produced this way are directional, such that different height, width, or spacing can be created for the two horizontal

axes. Interestingly, even for a single axis, textures can be programmed to change depending upon whether the mouse moves in the positive or the negative direction. This allows the simulation of velvet-like surfaces, which feel fuzzy in one direction and smooth in the opposite direction. If the bump spacing, the width, and the magnitude are equal in both axes of motion and in both directions, then a gridlike surface (such as the strings of a virtual tennis racket) can be simulated.

5.3.5.2 Haptic Textures Produced by the PHANToM. These textures have a richer content, since this interface can produce forces in any direction. As a consequence, friction forces can be added to contact forces in order to simulate sticky surfaces. Viscosity can be simulated by multiplying the surface velocity with a damper, while inertia can be modeled by a simple mass-acceleration product. Surface texture can be created with displacement maps, similar to the bump maps used in multitexturing. The displacement map models the cross section of the surface through hills and valleys and the force felt by the user is affected by the local gradient of the surface height, as in

$$Ah \text{ Ftexture } x = A \text{ O}x \quad (5.32)$$

$$Ah \text{ Ftexture } v = A \text{ O}y \quad (5.33)$$

where x and y are translation directions, A is a proportionality constant, and h is the surface height above the surface plane. Thus Ah/Ax and Ah/Ay are the local gradients in the x and y direction, respectively. These determine the texture forces applied by the haptic interface [Minsky et al., 1990].

Another approach to smoothness modeling uses sinusoidal functions [SensAble Devices, 1994]. The PHANToM can superimpose high-frequency vibrations on the force feedback signal felt at the user's fingertip. Here surface haptic texture in the z direction is modeled as

$$\text{Ftexture} = A \cdot \sin(mx) \cdot \sin(ny) \quad (5.34)$$

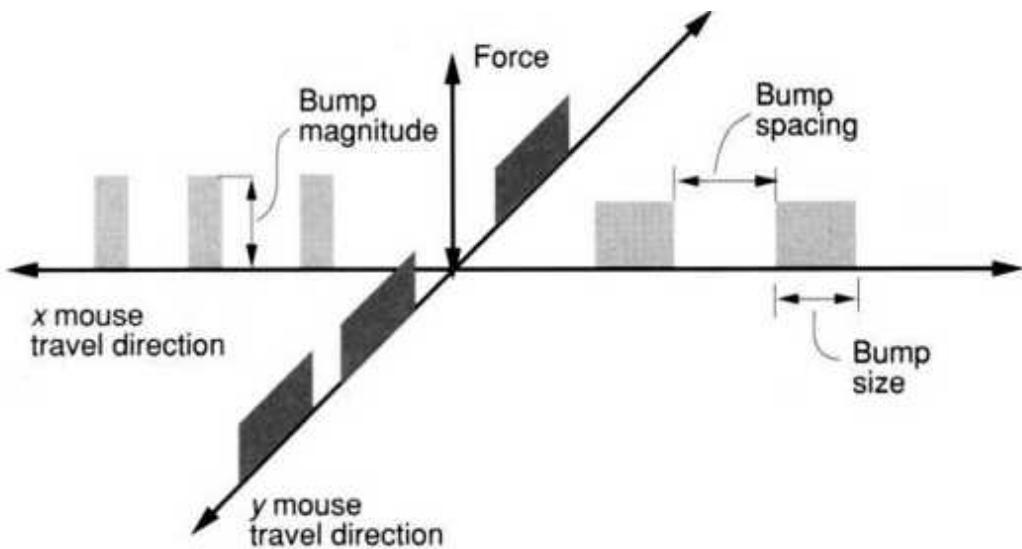


Fig. 5.30 Modeling surface roughness with a haptic mouse. Adapted from Immersion [2001]. @ 2001 Immersion Co. Reprinted by permission.

where A is the constant giving the magnitude (amplitude) of vibrations; m and n are constants that modulate the frequency of vibrations in the x and y directions, respectively; and x and y are in the range of $(0, 2n)$.

The higher the values of the constants m and n , the higher is the vibration frequency felt by the user. Depending on the values of the constants in Equation (5.34), users may perceive such textures as a shape (large A , small m, n) or as friction (small A , very large m, n) [Salisbury et al., 1995].

To summarize, the resistive force F_{resist} the user feels when interacting with an object using the PHANToM (or similar devices) is a sum of several components. One is the constraint force, to which are added the effects of texture and surface friction [Fritz and Barner, 1996]. The overall force is then given by

$$F_{\text{resist}} = F_{\text{constraint}} + F_{\text{texture}} + F_{\text{friction}} \quad (5.35)$$

5.4 BEHAVIOR MODELING

Until now our discussion has been limited to the mathematical modeling of object appearance, kinematics, and physical properties. Whenever objects

interacted, it was assumed that one was controlled by the user. It is also possible to model object behavior that is independent of the user's actions. This becomes critical in very large simulation environments, when users cannot possibly control all the interactions that are taking place.

Consider the modeling of a virtual office, for example. Such an office could have an automatic sliding door, a clock, a window thermometer, a desk calendar, as well as furniture. The time displayed by the clock and the date shown on the current calendar page can be updated by accessing the VR engine system time. The temperature displayed graphically by the window thermometer can be updated based on an external temperature sensor (such as a thermocouple or a thermoresistor) that is interfaced to the VR engine running the simulation. Every time the user enters the virtual office, the sliding door opens and some of the information displayed by the clock, calendar, and window thermometer changes. However, direct user input is limited in this example to just changing the field of view to the simulation.

This example illustrates one method of modeling object behavior by accessing external sensors (system time, thermocouples, and proximity sensors for the sliding door). This provides virtual objects with a degree of independence from the user's actions-a degree of "intelligence" Many current simulations also model virtual humans, called agents.

Definition An virtual human (or agent) is a 3D character that has a human behavior. Groups of such agents are called crowds and have crowd behavior.

Thalmann [2000] considered that the degree of autonomy of virtual environments depends on the autonomy of their components, namely interactive objects, agents, and crowds. Each of these can have three levels of autonomy (LOA), guided, programmed, and autonomous. Here guided refers to the lowest level of autonomy. For example, a guided agent needs a user-specified path in order to travel from one location to another. A guided door has to have its degree of opening controlled either by the user or by an agent, and so on. At the other extreme, an autonomous agent perceives information about the surrounding virtual environment and decides what path to follow accordingly. An autonomous door is in control of its motion, and

can even guide an agent that attempts to open it. Thus the simulation level of autonomy is given by

$$S_{\text{autonomy}} = f[\text{LOA(objects)}, \text{LOA(agents)}, \text{LOA(groups)}] \quad (5.36)$$

Fully autonomous agents, such as the virtual football player in the light uniform seen in Figure 5.31, need to perceive their environment (in this case the opponent) in order to take appropriate actions. The behavior model of the agent includes emotions, behavior rules, and actions. The agent behavior has its hierarchy, reflex behavior being at a lower level. A reflex behavior could be to tackle his opponent every time he sees him. Emotion-based behavior filters perceptual data through likes, dislikes, anger, or fear. It is thus at a higher level than simple reflex behavior. As a consequence, two agents interpreting the same sensorial data will take different actions in the simulation [Thalmann et al., 2000]. In our example, one virtual football player may decide to back away, etc.

An example of an agent exhibiting reflex behavior is Dexter, illustrated in Figure 5.32a [Johnson, 1991]. He was developed at the MIT Media Laboratory in the early 1990s and programmed to shake hands with the user. Reflex behavior was assigned to various parts of the model. User hand data were sampled by a VPL DataGlove, which controlled a virtual hand. Once the virtual handshake took place, users were in control of Dexter's arm. If the user moved to the right, the whole arm of Dexter rotated, and so on. In order to further increase the simulation realism, the agent was programmed to move his head toward the user.

A more complex example of reflex behavior involves agents that recognize and mimic the user's actions, as illustrated in Figure 5.32b [Emering et al., 1999]. Here the avatar in the light sweatshirt (away from the virtual camera) is controlled by the user, who wears a sensing suit. The character closer to the camera is an agent that recognizes the avatar's body posture (expression) and imitates it. Such body expressions include walking, pointing with the left or the right arm, grasping, as well as vertical or horizontal head movement. The same agent can have higher level behavior, for example, in a dance performance. In such a case, instead of imitating the

live dancer's motions (measured by the sensing suit), it performs its own motions out of a menu of prerecorded sequences.

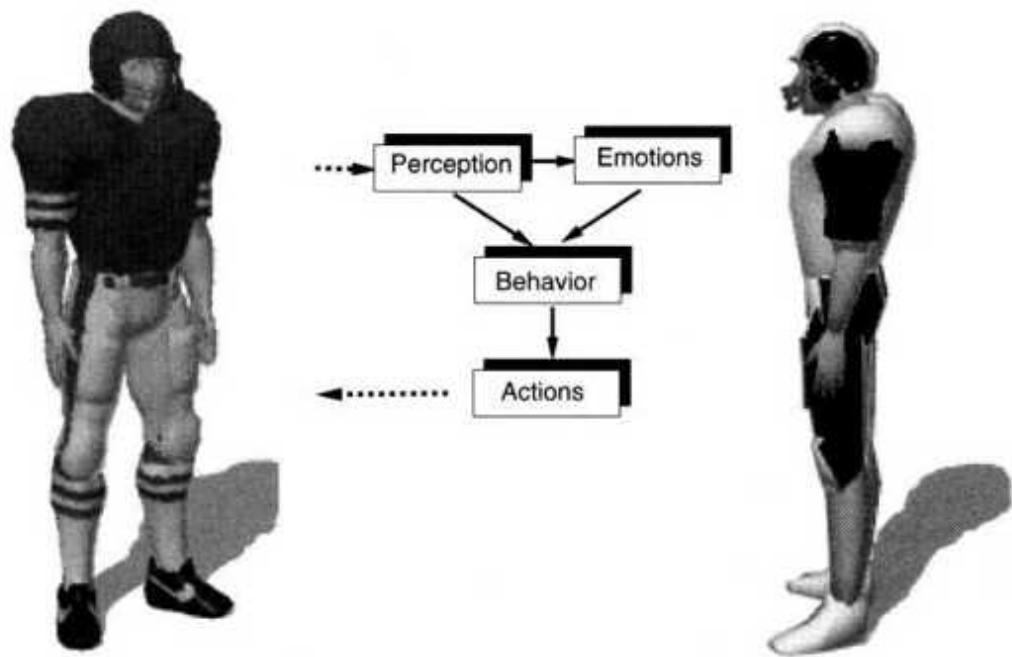
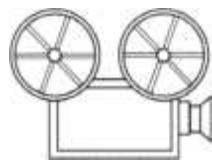
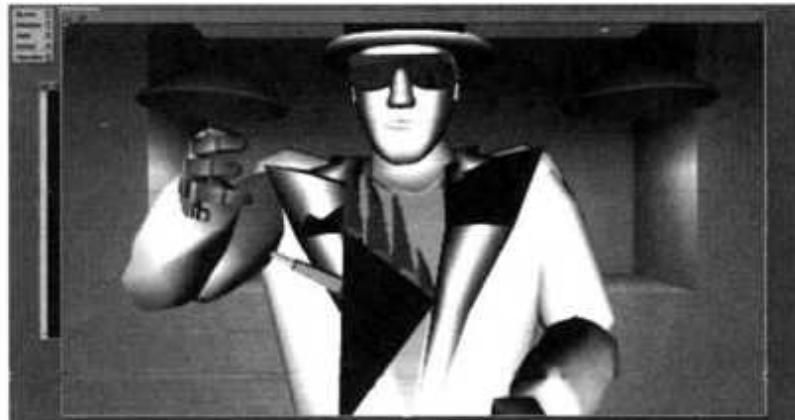


Fig. 5.31 Agent behavioral model. Adapted from Thalmann et al. [2000]. Reprinted by permission of Info rmatik/Informatique. Models from Viewpoint Co. Reprinted by permission.

Groups of agents form crowds, which can be guided, programmed, or autonomous. The level of autonomy of the crowd does not have to match that of its agents. For example, a guided crowd can be formed of autonomous agents. In this case the crowd has a common goal, but the agent has a way to perceive its immediate surroundings (within the crowd) and react. An example is the political demonstration illustrated in Figure 5.33 [Thalmann et al., 2000]. If the crowd is guided, the user needs to specify its way points or demonstration itinerary. Alternatively, the crowd may be programmed to follow its leader. This points to the need for a social hierarchy within a crowd and to negotiations between agents with the same level of autonomy. An element in crowd behavior is memory, such that the crowd would react in the same way to a given repeated event. This is used, for example, in building-evacuation simulations, where the crowd behavior is in response to a common threat.



VC 5.3



(a)



(b)

Fig. 5.32 Examples of agent reflex behavior. (a) Dexter intelligent agent. From Johnson [1991]. Reprinted by permission. (b) An agent mimicking the user's gestures. From Emering et al. [1999]. © 1999 IEEE. Reprinted by permission.



Fig. 5.33 Political demonstration involving autonomous and programmed groups of agents. From Thalmann et al. [2000]. Reprinted by permission of Informatik/Informatique.

5.5 MODEL MANAGEMENT

Geometrical, kinematics, physical, and behavior modeling of a highly populated virtual world will result in a very complex model. The resultant large computation load is difficult, if not impossible, for the VR engine to handle in real time. Additionally, the large memory requirement associated with complex virtual worlds may not fit on the available RAM. This results in costly (timewise) memory swaps, with data being brought from the much slower hard disk memory. Such memory access affects the constancy of the simulation frame refresh rate, with disturbing visual consequences (scene momentarily frozen, etc.).

These problems are faced by architects having to virtualize large buildings with many floors, offices, furniture, hallways, stairways, etc. Such

models may have hundreds of thousands of polygons. The same problem faces the developers of surgical simulators. Human anatomy is so complex that it is impossible to render the whole body in sufficient detail and at interactive rates. Other applications affected by model complexity problems are virtual prototyping, training and servicing, and military command and control.

Definition Model management combines techniques designed to help the VR engine render complex virtual environments at (predefined) interactive rates without a significant impact on the simulation quality.

There are several model management approaches; those we discuss involve selection of object level of detail, cell segmentation, off-line precomputations, and database management. Often the best improvement in simulation speed involves a combination of these approaches.

5.5.1 Level-of-Detail Management

At the beginning of this chapter we mentioned that online databases, such as that of Viewpoint Co., offer object geometry datasets with different numbers of surface polygons. These models represent the same object at different levels of detail. The reason to do so stems from the realization that the human eye perceives less and less detail as an object (real or virtual) is further and further away. Thus it would be wasteful to represent distant objects with a high level of detail.

Definition Level-of-detail (LOD) management combines methods used to improve the graphics pipeline throughput by selecting object level of detail appropriately.

Level-of-detail management can be further classified into static and adaptive.

5.5.1.1 Static LOD Management. These methods are based on object discrete geometry, "alpha blending," and morphing. Discrete-geometry LOD management is the simplest and was the first to be implemented. It divides the scene into concentric zones centered at the location of the virtual camera.

The radii of these concentric zones are user-defined. Object models are loaded by the application stage of the graphics pipeline based on the distance from the camera.

Let us consider a scene where a virtual rabbit hops away from the viewer, as illustrated in Figure 5.34 [Luebke, 2001]. The scene database contains the same rabbit modeled in discrete geometries, from a version with the highest surface detail (69,451 triangles) to one with the lowest (76 triangles). At the application stage of the graphics pipeline the distance between the rabbit and the virtual camera is determined. If it is less than r_1 , then the model with the highest polygonal count is used. If the virtual rabbit hops into the next zone (the distance from the viewer is somewhere between r_i and r_{i+1}), then the rabbit model with 2,502 triangles is loaded, and so on. A problem occurs when distance is equal to one of the user-specified radii, which represents the transition zone from one discrete LOD to the next. One way to avoid cycling between the two versions of the rabbit model is to define a hysteresis buffer A_r . In this way the switching between detail levels happens at different distances, depending upon whether the distance to the object increases or decreases. In our example, the switching to the 2,502-triangle version occurs only after the virtual rabbit reaches a distance of $r_i + A_r$ from the camera.

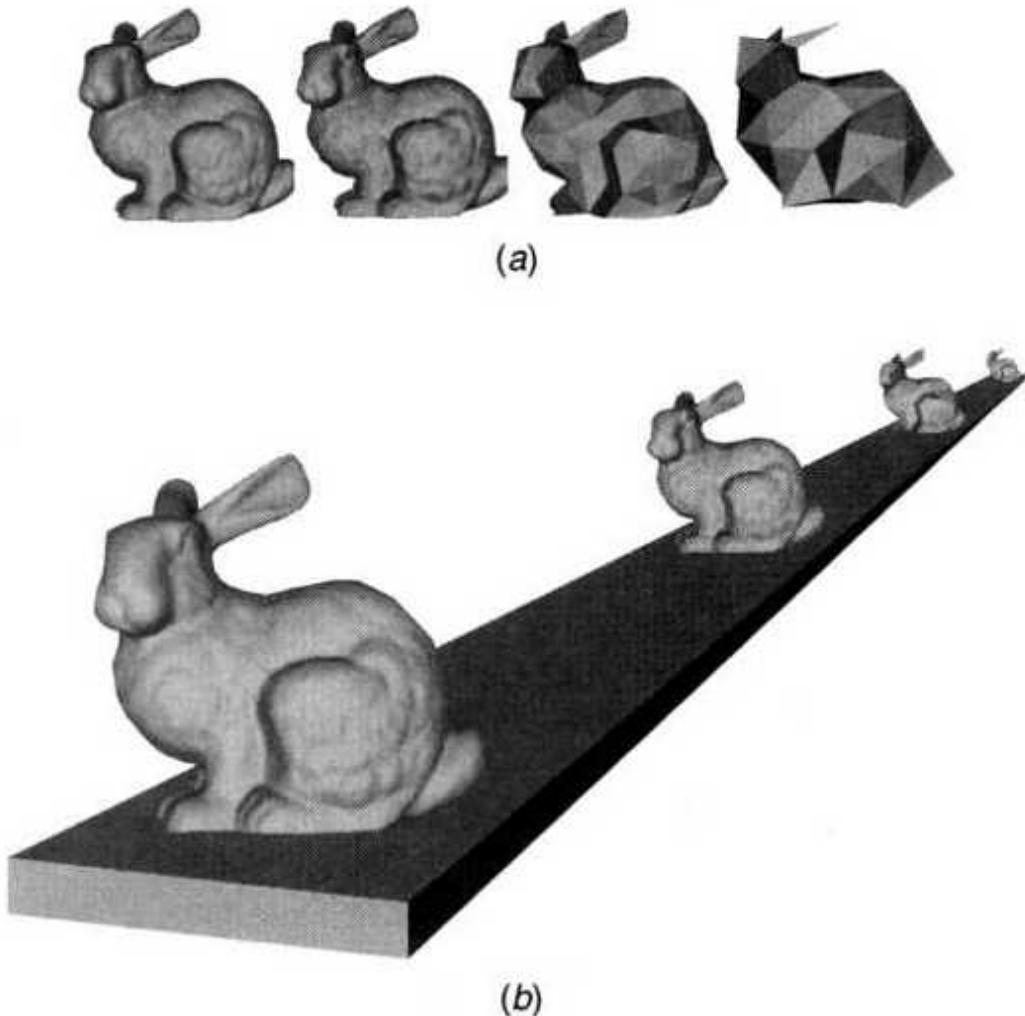


Fig. 5.34 Discrete-geometry LOD management: (a) rabbit model with polygon counts (from left to right) of 69,451, 2502, 251, and 76 triangles, respectively; (b) visual appearance as a function of the model's distance to the virtual camera. From Luebke [2001]. © 2001 IEEE. Reprinted by permission.

Although hysteresis prevents cycling, it does not address "popping" artifacts, which happen whenever there is a sudden transition in the model LOD. Morphing LOD management methods solve this issue by gradually changing the polygonal count of a single model. This is done through mesh simplification, which gradually reduces the polygonal count by collapsing edges. An edge is said to collapse when its two vertices merge. Gradual merging of the vertices can be made to depend on distance from the virtual camera, thus avoiding popping artifacts.

An alternative to morphing is "alpha blending," which sets the single model transparency as a function of its distance to the virtual camera. Models that are closer have high opacity, whereas objects that are far away have high transparency. Once the object becomes completely transparent, it will not be sent down the graphics pipeline, resulting in considerable speed improvement [Moller and Haines, 1999].

5.5.1.2 Adaptive LOD Management. These methods address some of the problems that static LOD management cannot solve. One example is the requirement to allow several LODs to coexist within a single model. Another is to select appropriate LODs to assure a constant frame refresh rate.

An example of an application requiring several coexistent LODs in an object is scientific visualization. Let us consider the example of the high-polygon-count sphere shown in Figure 5.35a [Xia et al., 1997]. Its surface has 8192 triangles, which gives good local illumination detail (sharp change in the high-intensity illumination area). However, the large number of polygons makes rendering very slow. In order to increase interactivity, one could apply a static LOD management approach, such as edge collapse. This reduces the surface detail uniformly, such that the model in Figure 5.35b has only 512 triangles. Although the sphere now takes much less time to be rendered, its visual level of detail is lost due to blurring in the local illumination area. An adaptive LOD management approach solves this by considering the surface lighting information. In this way it is possible to split vertices and increase detail only in the triangle mesh area where detail is needed, leaving the rest of the surface at a much lower detail. For example, the sphere in Figure 5.35c is visually equivalent to the high-LOD one, but has only 537 triangles. This adaptive LOD management involves a precomputation phase, which determines the way vertices are collapsed or split, depending on the specific illumination and viewing angle. This information is stored in a merge tree, which is accessed in real time by the application stage of the graphics pipeline based on the user-controlled viewing angle. Tests done on an SGI Reality Engine 2 showed a significant speedup in rendering when the adaptive LOD approach was used [Xia et al., 1997]. Specifically, the sphere in Figure 5.35a took 0.115 sec to render,

whereas the one in Figure 5.35c was rendered in only 0.024 sec (almost five times faster).

This adaptive LOD approach works when the scene content is known a priory. This is not, however, the case in architectural walkthroughs and other applications where the scene complexity can vary dramatically (from hundreds to tens of thousands of polygons) depending on the user-controlled viewpoint. The result is a significant variation in frame refresh rate from one scene to the next. This disturbing effect cannot be solved by simple static LOD management, which treats objects in isolation. What is needed is an adaptive LOD approach which takes into account the cumulative effect of visible LOD on the frame rendering time. Such a method will then select LODs so that a user-specified frame refresh rate is maintained.

Let us consider the example in Figure 5.36, which illustrates a walkthrough of the auditorium in Soda Hall, University of California at Berkeley [Funkhouser and Sequin, 1993]. The frame rendering time is maximum when the virtual camera is at location A and all the objects in the auditorium are visible. As the observer advances to the front of the auditorium, fewer and fewer rows of chairs are visible, such that a minimum frame rendering time is reached at location B. When the path is reversed (location C to End), the frame rendering time has a similar variation, as shown in Figure 5.36c. In this example the auditorium model had 72,570 polygons and no LOD management was applied. This resulted in a variation in frame rendering time from 0.43 sec (mean) to 0.99 sec (max), with a standard deviation of 0.305 sec. Applying static LOD management to the same scene improves refresh rate overall, but cannot maintain a fixed value (maximum frame rendering time is 0.20 sec and mean is 0.11 sec).

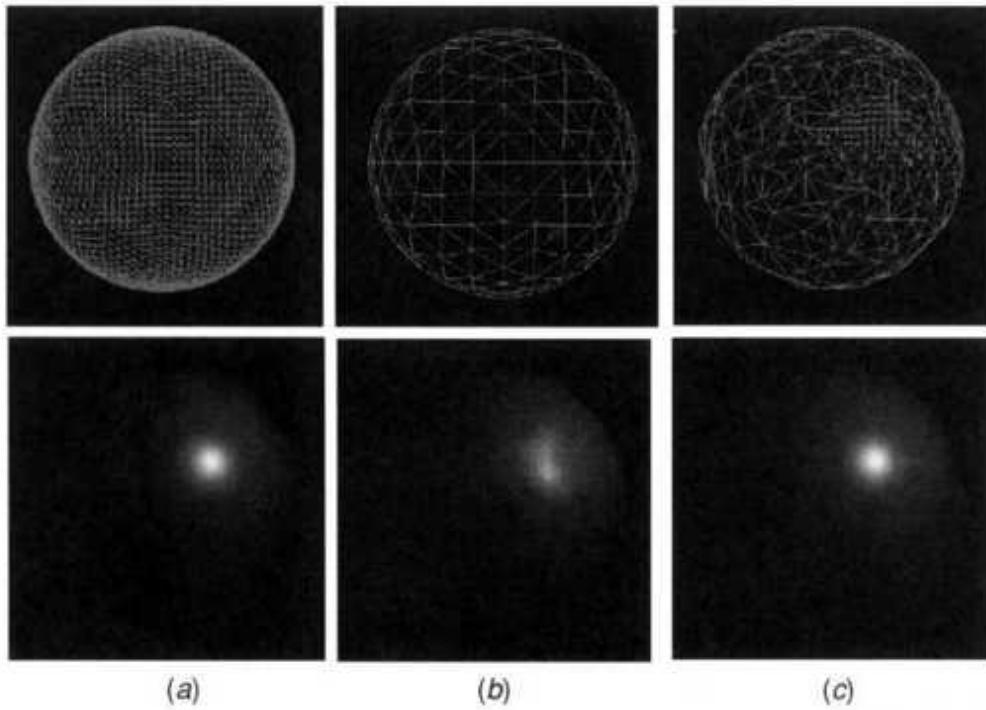


Fig. 5.35 Single-object adaptive LOD management: (a) uniform high-density sphere (8,192 triangles); (b) static LOD low-density sphere (512 triangles); (c) adaptive LOD sphere (537 triangles). From Xia et al. [1997]. © 1997 IEEE. Reprinted by permission.

Figure 5.37a shows the same auditorium, this time rendered with an adaptive LOD, where the aim is to maintain a fixed frame rendering time of 0.1 sec. The method considers that objects have a value for the scene given by the ratio

$$\text{Benefit}(\text{Object}, \text{LOD}, R) / \text{ValueofeCr} = \text{Cost}(\text{Object}, \text{LOD}, R) \quad (5.37)$$

where $\text{Benefit}(\text{Object}, \text{LOD}, R)$ is the benefit of the object to the scene, which depends mainly on its size (how close it is to the virtual camera); $\text{Cost}(\text{Object}, \text{LOD}, R)$ is the cost of rendering that object (the time it takes to do so on a given computer); and R is the rendering mode.

In Equation (5.37) the cost is the largest of either the geometry-stage rendering time or the rasterizing-stage rendering time for that object. In order

to maintain a userspecified frame rendering time, the overall scene rendering cost has to satisfy the condition

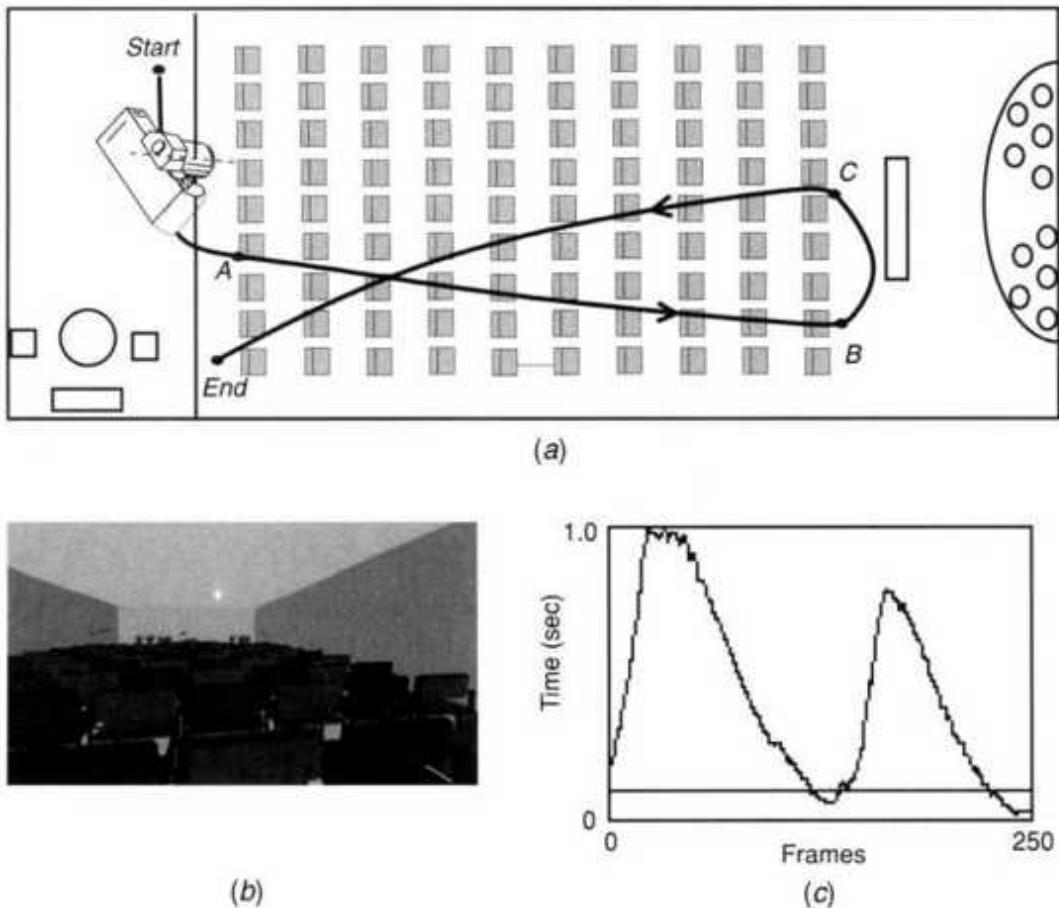


Fig. 5.36 Influence of scene complexity on frame rendering time: (a) virtual camera path; (b) auditorium scene with no LOD segmentation (72,570 polygons); (c) resultant frame rendering time. From Funkhouser and Sequin [1993]. © 1993 ACM Inc. Reprinted by permission.

$$T_{\text{Cost}}(\text{Object}_i, \text{LOD}_i, R) < \text{Target frame time} \quad (5.38)$$

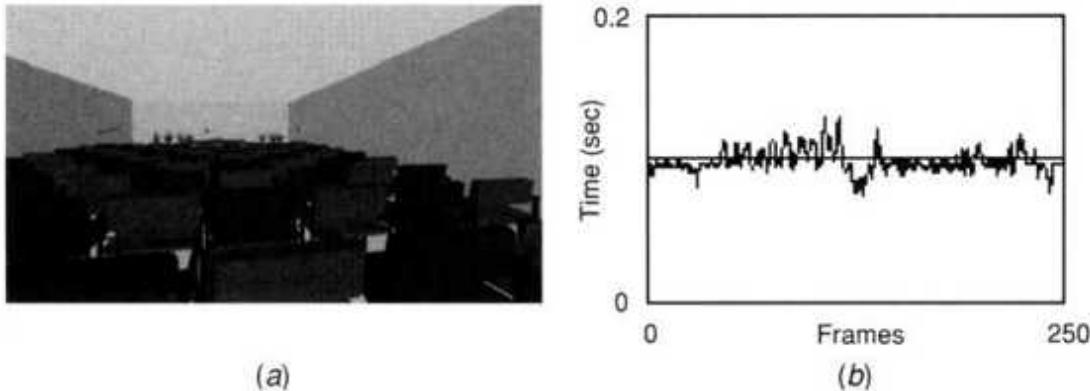


Fig. 5.37 Adaptive LOD management for complex scenes: (a) auditorium scene with optimized level of detail segmentation (5,300 polygons); (b) resultant frame rendering time. From Funkhouser and Sequin [1993]. © 1993 ACM Inc. Reprinted by permission.

With this constraint, the adaptive LOD management method selects objects that maximize the value of the scene, which means objects with the largest Value will be rendered first. This method also considers that objects can have no polygons, corresponding to their lowest LOD. As such low-value objects will not be rendered in the scene, even if they may be visible. Figure 5.37b shows the variability of the frame rendering time with adaptive LOD management. The graph shows that over 250 frames, the average frame rendering time is the desired 0.1 sec and the maximum frame rendering time is 0.13 sec. The standard deviation in this case is only 0.008, which represents a much smoother simulation.

5.5.2 Cell Segmentation

In the foregoing example the model size was small enough to fit in RAM memory, such that there was no need to retrieve data from the hard drive. Larger models need to be rendered in such a way that the impact of memory swaps on simulation frame rate is minimized. This involves partitioning the large model into smaller ones, then rendering those with static or adaptive LOD management.

5.5.2.1 Automatic Cell Segmentation. This process partitions the virtual world into smaller universes, or cells. Only objects within the current universe are rendered, thus reducing complexity significantly. This approach

is illustrated in Figure 5.38 [Pimentel and Teixeira, 1993]. This segmentation method is ideal for architectural modeling of large buildings (rather than a single auditorium). In this case most of the model does not contribute at all to any given image. The set of polygons that appears in each view changes slowly as the viewpoint moves, except when crossing certain thresholds (going from one room to another). Here cells can be approximated to rooms, and segmentation is easier to implement automatically. Additionally, since the model, once done by the architect, does not change easily, segmentation can be done offline in a precomputation stage.

Early work on model segmentation for virtual buildings was done by Airey and his colleagues at the University of North Carolina at Chapel Hill [Airey et al., 1990]. They observed that user behavior in walkthrough simulations was qualitatively different, depending on the frame rate. At very low frame refresh rates (1 fps) the users needed a two-dimensional floor-plan display, or map view, to navigate. The virtual building illusion, where it was possible to navigate with 3D views, needed at least 6 fps, and interactivity needed 20 fps. Motivated by these findings, Airey developed an automated cell segmentation method. Here the union of visible polygons for all viewpoints in a cell was considered the potentially visible set (PVS) for that cell. For any viewpoint in a cell, rendering the PVS for that cell assures that there are no missing polygons in the image. Since the size of the PVS is much smaller than the total size of the model, rendering takes much less time. This algorithm is illustrated in Figure 5.39 for a simple three-room floor plan.

Here rooms 1-3 approximate the cells. If doors are closed, then only polygons in a given room need to be rendered when the viewpoint is in that room. This results in an approximately threefold increase in the fps rate! Floor-plan automatic partitioning identifies and numbers planes according to a partition priority with values from 0 to 1. The weighting of the partition priority is expressed by

$$\text{Partitionpriority} = 0.5 \cdot \text{occlusion} + 0.3 \cdot \text{balance} + 0.2 \cdot \text{split} \quad (5.39)$$

where occlusion refers to doors in walls; balance refers to how evenly a plane separates the model; and split refers to how little the plane splits individual polygons (a split polygon needs to be included in the potentially visible set of both sides of the dividing plane).

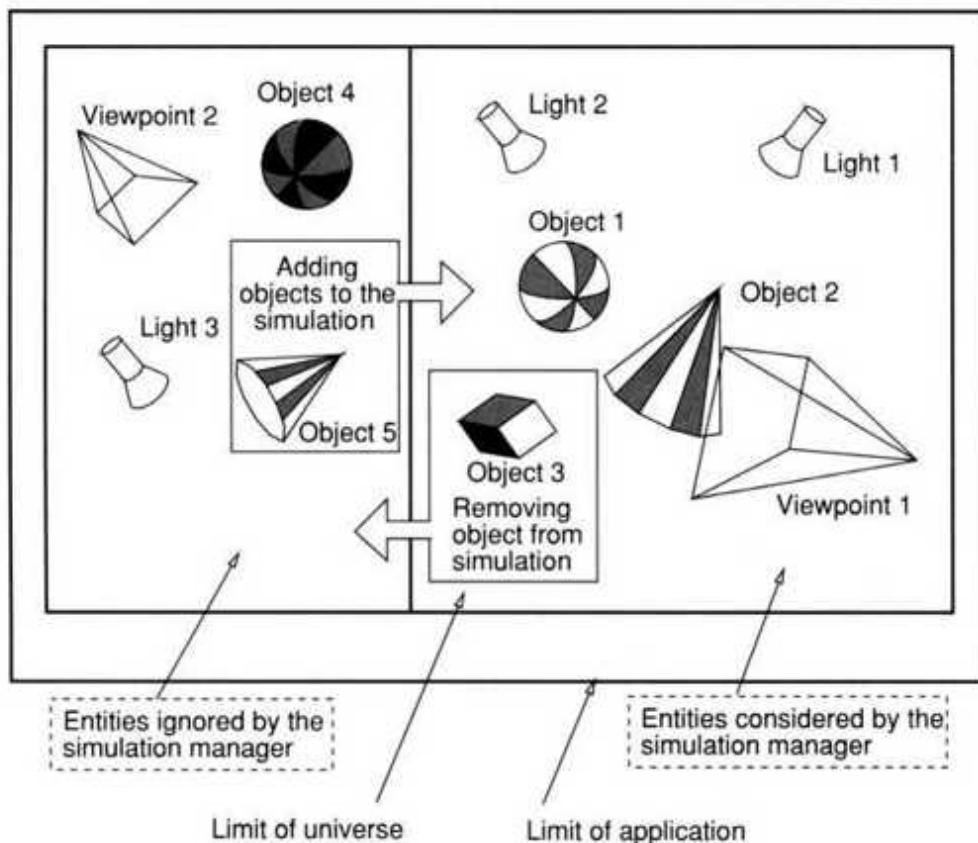


Fig. 5.38 Management of model complexity based on cell segmentation.
Adapted from Pimentel and Teixeira [1993]. Reprinted by permission.

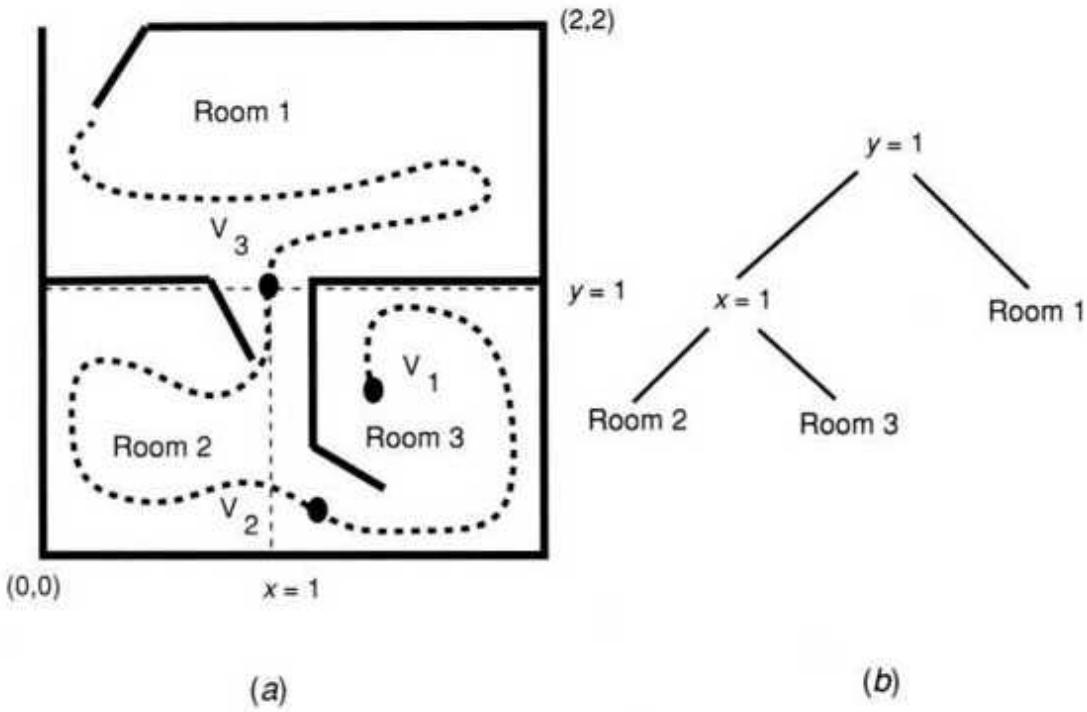


Fig. 5.39 Floor plan segmentation: (a) virtual camera path; (b) tree graph. From Airey et al. [1990]. © 1990 ACM Inc. Reprinted by permission.

We see that priority is weighted in favor of the occlusion factor. For the floor plan shown in Figure 5.39a, the plane $y = 1$ that separates room 1 from room 2 and room 3 would have a higher priority than the plane $x = 1$. This is based on the higher occlusion factor of $y = 1$. The process repeats for splitting room 2 from room 3, resulting in the tree-graph structure shown in Figure 5.39b.

When a cell is completely sealed (no doors), then the PVS is the set of polygons that intersect that cell. If the cell has holes in its boundary, then certain external polygons to a cell have to be added to its PVS. This problem is called the volumeto-polygon visibility problem, and the holes in the cell boundary are called portals. Thus, the cell PVS is enhanced with the union of all the external polygons visible from its portals. Airey subsequently applied this algorithm to the model of Sitterson Hall at the University of North Carolina at Chapel Hill. This 7,125-polygon model was thus partitioned into 269 cells and rendering time was measured. The

resultant frame refresh rate was increased (on average) by a factor of 30 compared to the frame refresh rate of the monolithic (nonsegmented) model.

5.5.2.2 Combined Cell, LOD, and Database Methods. This approach is needed for the management of very large models. The auditorium used as an example for adaptive LOD management-based walkthrough had about 80,000 polygons. However, the modeling of the whole floor of Soda Hall required 242,668 polygons. Cell segmentation subdivided the model in 1,280 cells with a total of 3,600 portals. Simple adaptive LOD management will not assure constant frame rendering time for a model of this size. Thus it is necessary to add a database management layer, which pre-fetches portions of the model, depending on the viewpoint location. The algorithm estimates how far the virtual camera will rotate over the next N frames, and the corresponding cells are pre-fetched from the hard disk. Subsequently, LOD is used on the fetched cells to prioritize the order in which models are loaded in the pipeline. The highest priority is assigned to models with low LOD, while models with high LOD are loaded last. Furthermore, these high-LOD models are loaded only in cells adjacent to the one occupied by the virtual camera, as illustrated in Figure 5.40 [Funkhouser, 1996]. In this way the frame rendering time has much less variability compared to the case where page faults occur because the needed model is not in system RAM. This is made apparent by the graphs in Figures 5.40b, and 5.40c.

5.6 CONCLUSION

This chapter introduced the various aspects involved in virtual object modeling. These include geometrical modeling, kinematic modeling (defining parent-child hierarchies), physical modeling, and object intelligent behavior. The limited computational power available and the desire for simulation realism require management of model size and complexity, including level of detail and cell segmentation. Several of these features are included in software toolkits designed specifically for VR programming. These are discussed in the next chapter.

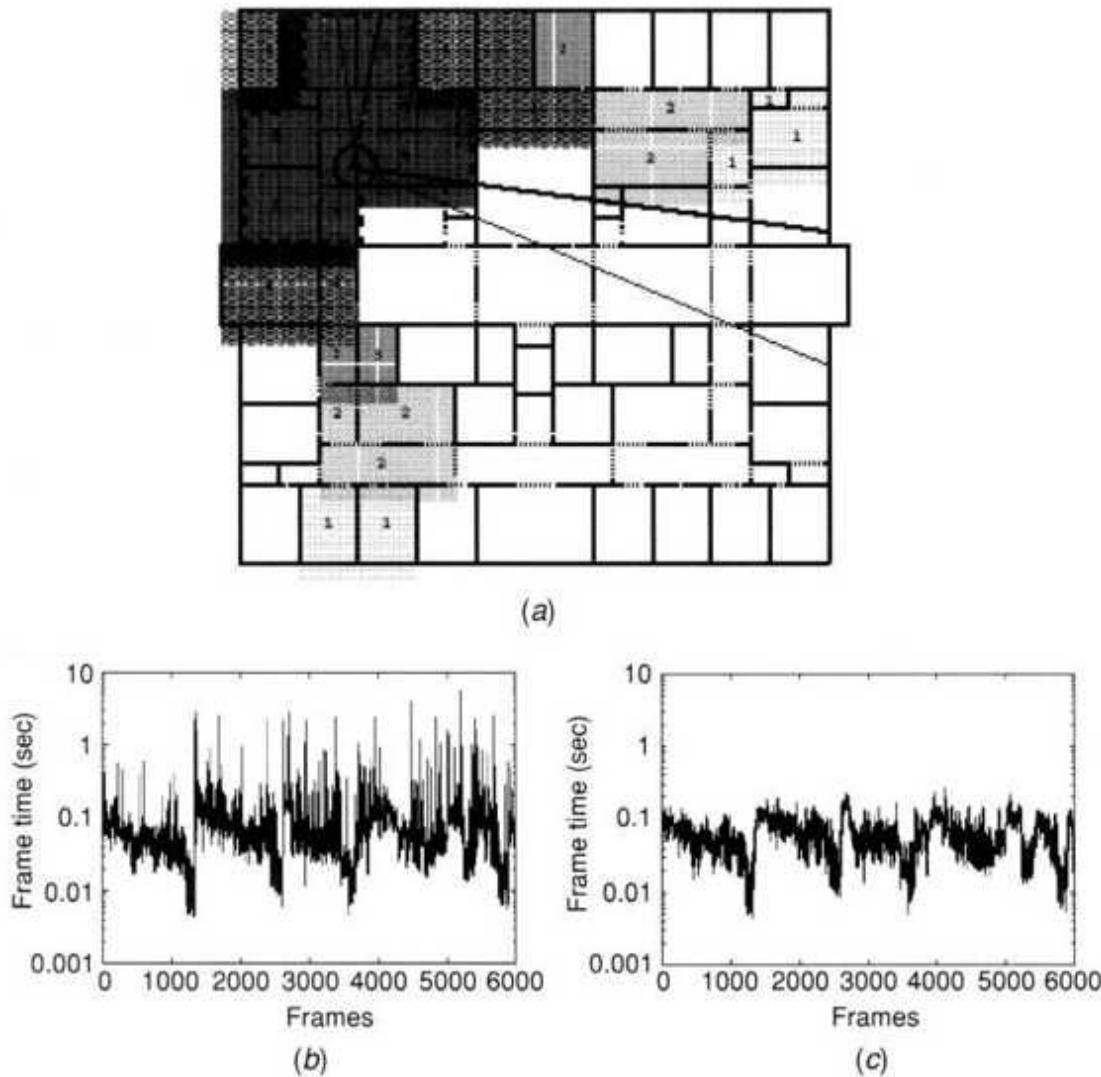


Fig. 5.40 Database management for large models: (a) lookahead set (LOD 1 is the lowest, LOD 4 is the highest); (b) frame rendering time without database management; (c) frame rendering time with database management. From Funkhouser [1996]. Reprinted by permission from the Canadian Information Processing Society (CIPS).

5.7 REVIEW QUESTIONS

1. What is the mathematical description for object position/orientation in virtual environments? How can invariants be used?

2. What is the difference between local and global illumination? How can global illumination be made interactive?
3. What are splines? How are they used in shape modeling and in surface texturing? Make a drawing and explain. What are advantages of textured scenes versus nontextured smooth-shaded ones?
4. Give examples of multitexturing and explain their importance.
5. What are homogeneous transformation matrices? How are they inverted? How is object translation and scaling done? What is the viewing transform? Make a drawing and explain.
6. How do the projection, clipping, and screen mapping substages of the geometry pipeline stage work? Make a drawing and explain.
7. What is a hierarchical structure in VR? How does it apply to a virtual hand?
8. What kind of bounding boxes exist? What are they used for?
9. How do you model contact forces for an elastic object that is homogeneous? How about one that has a harder interior kernel?
10. Repeat Question 9, but for a plastically deformed object.
11. Repeat Question 10, but for a wall object.
12. Describe the active surface deformation model. How can it be applied to surface cutting?
13. What is object intelligent behavior in VR?
14. What is a virtual human agent? What is its behavior model? What levels of autonomy (LOA) does it have?
15. How does model segmentation influence rendering speed? Describe methods and comment.

16. What is cell-based segmentation. How can it be improved?

REFERENCES

- Airey, J., J. Rohlf, and F. Brooks, Jr., 1990, "Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments," Computer Graphics, Vol. 24(2), pp. 41-50.
- Autodesys Inc., 1998, "3D Digitizing," online at ftp://ftp.formz.com/pub/formz/PDF_files/digitize.pdf.
- Bishop, G., 1986, "Fast Phong Shading Algorithm," in Proceedings of SIGGRAPH 1986, pp.103-105.
- Blinn, J., 1978, "Simulation of wrinkled surfaces," in SIGGRAPH Proceedings, Vol. 11(3), pp. 286-292.
- Blinn, J., and M. Newell, 1976, "Texture and Reflection in Computer Generated Images," CACM, Vol. 19(10), pp. 542-547.
- Bui-Tuong, P., 1975, "Illumination for Computer Generated Pictures," CACM, 18(6), pp. 311-317.
- Burdea, G., 1993, "Virtual Reality Systems and Applications" [Short Course], in Electra '93 International Conference, Edison, NJ.
- Burdea, G., and P. Coiffet, 1993, La Realite Virtuelle, Hermes, Paris.
- Burdea, G., E. Roskos, D. Silver, F. Thibaud, and R. Wolpov, 1992, "A Distributed Virtual Environment with Dextrous Force Feedback," in Proceedings of Interface to Real and Virtual Worlds Conference, Montpellier, France, pp. 255-265.
- Catmull, E., 1974, "A Subdivision Algorithm for Computer Display of Curved Surfaces," Ph.D. Thesis, Technical Report UTEC-CSc-74-133, Computer Science Department, University of Utah, Salt Lake City, UT.

Cohen, J., M. Lin, D. Manocha, and M. Ponamgi, 1995, "I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments," in Proceedings of ACM Interactive 3D Graphics Conference, Monterrey, CA, pp. 189-196. Also online at <ftp://ftp.cs.unc.edu/pub/users/manocha/PAPERS/COLLISION/paper3dint.pdf>.

Colgate, E., P. Grafing, M. Stanley, and G. Schenkel, 1993, "Implementation of Stiff Virtual Walls in Force-Reflecting Interfaces," in Proceedings of the IEEE Virtual Reality Annual International Symposium (VRAIS), Seattle WA, pp. 202-208.

Computer Graphics Systems Development, 2000, "RealTexture Library Complete Catalog," Computer Graphics Systems Development Co., Mountain View, CA, 2000. Also online at www.cgsd.com/texture/Brochure.pdf.

Denavit, J., and R. Hartenberg, 1955, "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices," Journal of Applied Mechanics, Vol. 77, pp. 215-221.

Dinsmore, M., N. Langrana, G. Burdea, and J. Ladeji, 1997, "Virtual Reality Training for Palpation of Subsurface Tumors," in Proceedings of IEEE International Symposium on Virtual Reality and Applications (VRAIS '97), Albuquerque, NM, pp. 54-60.

Emering, L., R. Boulic, and D. Thalmann, 1999, "Body Expression in Virtual Environments," in Proceedings 8th IEEE International Workshop on Robot and Human Interaction RO-MAN '99, Pisa, Italy, pp. 177-182. Also online at ligwww.epfl.ch.

Foley, J., A. van Dam, S. Feiner, and J. Hughes, 1990, Computer Graphics: Principles and Practices, Addison-Wesley, Reading, MA.

Foley, J., A. van Dam, S. Feiner, J. Hughes, and R. Phillips, 1994, Introduction to Computer Graphics, Addison-Wesley, Reading MA.

Fritz, J., and K. Barner, 1996, "Stochastic models for haptic texture," in Proceedings of SPIE International Symposium on Intelligent Systems and Advanced Manufacturing- Teleoperator and Telepresence Technologies III, Boston, pp. 34-44. Also online at www.ee.udel.edu/InfoAccess/Technology/haptic.html.

Fu, K., R. Gonzalez, and C. Lee, 1987, Robotics: Control, Sensing, Vision and Intelligence, McGraw-Hill, New York.

Funkhouser, T., and C. Sequin, 1993, "Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments," Computer Graphics, Vol. 27, pp.247-254.

Funkhouser, T., 1996, "Database Management for Interactive Display of Large Architectural Models," in Proceedings of Graphics Interface, Toronto, Canada, pp.1-8.

Goldberg, E., 2001,"Scan Your World with 3D Lasers," Cadalyst Magazine, 2001 (February), online at www.cadalyst.com/features/0201cyra/.

Gouraud, H., 1971, "Continuous Shading of Curved Surfaces," IEEE Transactions on Computers, Vol. C-20(6), pp. 623-629.

Hewlett-Packard, 1991, Starbase Display List Programmer's Manual, Hewlett-Packard Co., Palo Alto, CA.

Ho, C., C. Basdogan, and M. Srinivasan, 1999, "An efficient haptic rendering technique for displaying 3D polyhedral objects and their surface details in virtual environments," Presence, Vol. 8(5), pp. 477-491.

Hsu, W., J. Hughes, and H. Kaufman, 1992, "Direct Manipulation of Free-Form Deformations," Computer Graphics, Vol. 26(2), pp. 177-184.

Immersion Co., 2001, "Immersion TouchSense Fundamentals," online at <http://www.immersion.com/developer/downloads>.

Johnson, M., 1991, "Build-a-Dude: Action Selection Networks for Computational Autonomous Agents," MS Thesis, MIT Media Lab,

Cambridge, MA.

Lin, M., 1993, "Efficient Collision Detection for Animation and Robotics," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA.

Lin, M., D. Manocha, J. Cohen, and S. Gottschalk, 1996, "Collision Detection: Algorithms and Applications," in J.-P. Laumond, and M. Overmars (Eds.), in Proceedings of Algorithms for Robotics Motion and Manipulation, pp. 129-142.

Luebke, D., 2001, "A Developer's Survey of Polygonal Simplification Algorithms," IEEE Computer Graphics and Applications, 2001 (May/June), pp. 24-35.

McAllister, D., 1998, "Introduction to Computer Graphics (Comp 136)," Computer Science Department, University of North Carolina at Chapel Hill, online at www.cs.unc.edu/~davemc/Class/136.

Minsky, M., M. Ouh-young, O. Steele, F. Brooks, Jr., and M. Behensky, 1990, "Feeling and Seeing: Issues in Force Display," Computer Graphics, Vol. 24(2), pp. 235-243.

Morgenbesser, H., and M. Srinivasan, 1996, "Force Shading for Haptic Shape Perception," in Proceedings of the ASME Dynamics Systems and Control Division, DSC-Vol. 58, ASME, pp. 407-412.

Moller, T., and E. Haines, 1999, Real-Time Graphics, A. K. Peters, Natick, MA.

Mortensen, Z., 1995, "OTMPHONG.DOC-A New Approximation Technique for the Phong Shading Model Based on Linear Interpolation of Angles," online at www.gamedev.net/reference/articles/article324.asp.

NVIDIA, 2000, "Per-Pixel Lighting and Bump Mapping with the NVIDIA Shading Rasterizer," Technical Brief, NVIDIA Co., Santa Clara, CA. Also online at www.nvidia.com/docs/10/86/ATT/Nsr.pdf.

Okino Inc., 1997 "Image Galery," online at www.okino.com/slideshow/venus.htm.

Pimentel, K., and K. Teixeira, 1993, Virtual Reality: Through the New Looking Glass, Windcrest McGraw-Hill, New York.

Polhemus, S., 1998, "Handheld Laser Scanner," company brochure, Polhemus Inc., Colchester, VT. Also online at www.polhemus.com/fastscan.htm.

Popescu, V., G. Burdea, and M. Bouzit, 1999, "Virtual Reality Simulation Modeling for a Haptic Glove," in Computer Animation'99 Conference, Geneva, pp. 195-200.

Raindrop Geomagic, 1999, "Wrap," company brochure, Raindrop Geomagic Inc., Research Triangle Park, NC. Also online at www.geomagic.com/advantage/gallery/process.php3.

Realworld Imagery Inc., 2002, "ImageCELS Professional Trees & Shrubs Catalog of Thumbnail Images," online at www.imagecels.com/thumbnaiUI15/115.html.

Reed, A., 1990, "Achieving CAD Photorealism and Pattern Matching," MicroCAD News, vol. 5(10), 1990 (October).

Robinett, W., and R. Holloway, 1992, "Implementation of Flying, Scaling and Grabbing in Virtual Worlds," in Proceedings of the 1992 Symposium on Interactive 3D Graphics, ACM, pp. 189-192.

Rosenberg, L., and B. Adelstein, 1993, "Perceptual Decomposition of Virtual Haptic Surfaces," in Proceedings of IEEE 1993 Symposium on Research Frontiers in Virtual Reality, San Jose, CA, pp. 46-53.

Salcudean, S., and T. Vlaar, 1994, "On the Emulation of Stiff Walls and Static Friction with a Magnetically Levitated Input/Output Device," in Proceedings of ASME WAM, DSC- Vol. 55-1, ASME, pp. 303-309.

Salisbury, K., D. Brock, T. Massie, N. Swarup, and C. Zilles, 1995, "Haptic Rendering: Programming Touch Interaction with Virtual Objects," in

Symposium on Interactive 3D Graphics, ACM, Monterey, CA, pp. 123-130.

Schlaroff, S., and A. Pentland, 1991, "Generalized Implicit Functions for Computer Graphics," *Computer Graphics*, Vol. 25(4), pp. 247-250.

Schoffel, F., 1997, "Online Radiosity in Interactive Virtual Reality Applications," in Proceedings of Virtual Reality Software and Technology Symposium '97, ACM, Lausanne, Switzerland, pp. 201-208.

SensAble Devices, 1993, PHANToM Master User's Manual, SensAble Devices Co., Cambridge, MA.

Soler, C., and F. Sillion, 1998, "Automatic Calculation of Soft Shadow Textures for Fast, High Quality Radiosity," in Proceedings of 1998 Eurographics Rendering Workshop, New York, pp. 199-210.

Song, G.-J., and N. Reddy, 1995, "Tissue Cutting in Virtual Environments," K. Morgan, R. Satava, H. Sieburg, R. Mattheus, and J. Christensen (Eds.), in *Interactive Technology and the New Paradigm for Healthcare*, IOS Press, Amsterdam, pp. 359-364.

Sturman, D., 1992, "Whole-Hand Input," Ph.D. Thesis, School of Architecture and Planing, Massachusetts Institute of Technology, Cambridge, MA.

Thalmann, D., S. Musse, and M. Kallmann, 2000, "From Individual Human Agents to Crowds," *Informatique*, 2000 (No. 1), pp. 6-11. Online at ligwww.epfl.ch/.

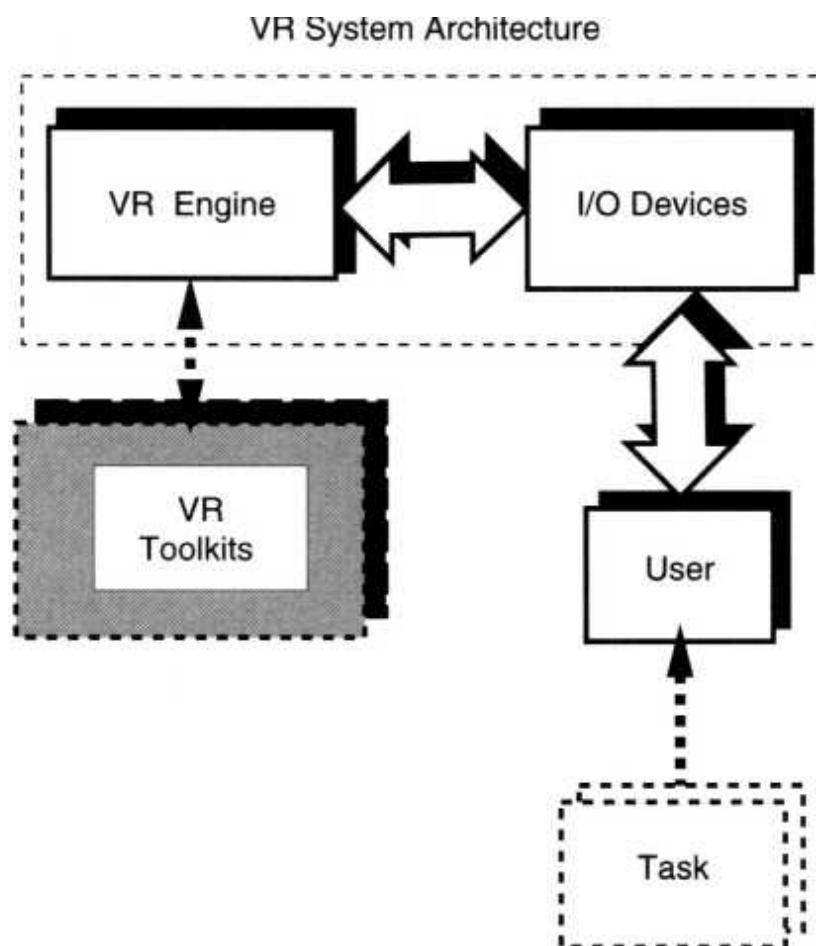
Viewpoint Inc., 2002a, "Viewpoint Catalog," Model #: VP22880, online at www.viewpoint.com/vp/catalog.jsp?style=premier.

Viewpoint Inc., 2002b, "Viewpoint Catalog," Model #: VP2309, online at www.viewpoint.com/vp/catalog.jsp.

Xia, J., J. El-Sana, and A. Varshney, 1997, "Adaptive Real-Time Level-of-Detail-Based Rendering for Polygonal Models," *IEEE Transactions on Visualization and Computer Graphics*, Vol 3(2), pp. 171-181.

CHAPTER 6

VR PROGRAMMING



Chapter 5 showed how to model the geometry, appearance, physical, and behavior properties of virtual objects as well as their parent-child object hierarchy. It also described ways to manage the resulting world (or universe) in order to reduce the frame rendering time. These are the basic components of programming, or authoring, a virtual environment.

Definition An authoring environment is an application programming interface (API)-based method of creating a virtual world. A run-time environment

subsequently allows the user's real-time interaction with the authored world.

6.1 TOOLKITS AND SCENE GRAPHS

For maximum programming flexibility authoring can be done with low-level graphics languages (such as OpenGL). This is a trial-and-error iterative process that requires skill and time. The task is complicated by the lack of standards among various programming languages [Roehl, 1995] and by a knowledge gap hampering nonprogrammers. Therefore the VR industry has created a number of advanced software tools to help in the authoring effort, called VR toolkits.

Definition A toolkit is an extendable library of object-oriented functions designed for VR specifications. Simulated objects are parts of classes and inherit their default attributes, thus simplifying the programming task.

Since the libraries are extendable, it is possible for developers to write applicationspecific modules and still use the same simulation kernel. The functions are written to use various hardware platforms and to be generic in nature. This is realized by having high-level functions not "know" the specific hardware they are running on. Low-level translators identify the specific I/O devices at run time. This is of great help when porting an application from one platform to another. Furthermore, toolkits support some form of networking in order to allow multiuser interaction.

The first toolkits, introduced in the early 1990s, included VCToolkit (VCT) [Division, 1993], Cyberspace Developer Kit (CDK) [Autodesk, 1993], Virtual Reality Toolkit (VRT3) [Dimension International, 1993], and Rend386 [Stampe et al., 1993]. These were hampered by the low-performance and expensive graphics hardware available at the time. As a consequence the low-end toolkits, such as VRT3 and Rend386, were designed to work without graphics accelerators, by allowing only flat shading and reduced scene complexity.

Rend386 was developed at the University of Waterloo, Canada. It was distributed free in order to encourage the creativity of students and scientists who were interested in virtual reality. The toolkit used integer arithmetic and a combination of C and assembler code. It was possible to render up to 22,000 polygons/sec on a PC without a graphics accelerator. A sample virtual scene authored with Rend386 is shown in Figure 6.1 [Stampe et al., 1993].

The foregoing toolkit examples were all general purpose, being designed for a variety of simulation applications. Modern general-purpose toolkits include WorldToolKit (WTK) [Sense8 Co., 2001], Java 3D [Sowizral et al., 1998], and Virtual Reality Modeling Language (VRML) [Ames et al., 1997]. Other toolkits in use are special-purpose libraries such as GHOST [SensAble Technologies, 1999] for haptics programming and PeopleShop [Boston Dynamics, 2001], used mostly for military simulations.

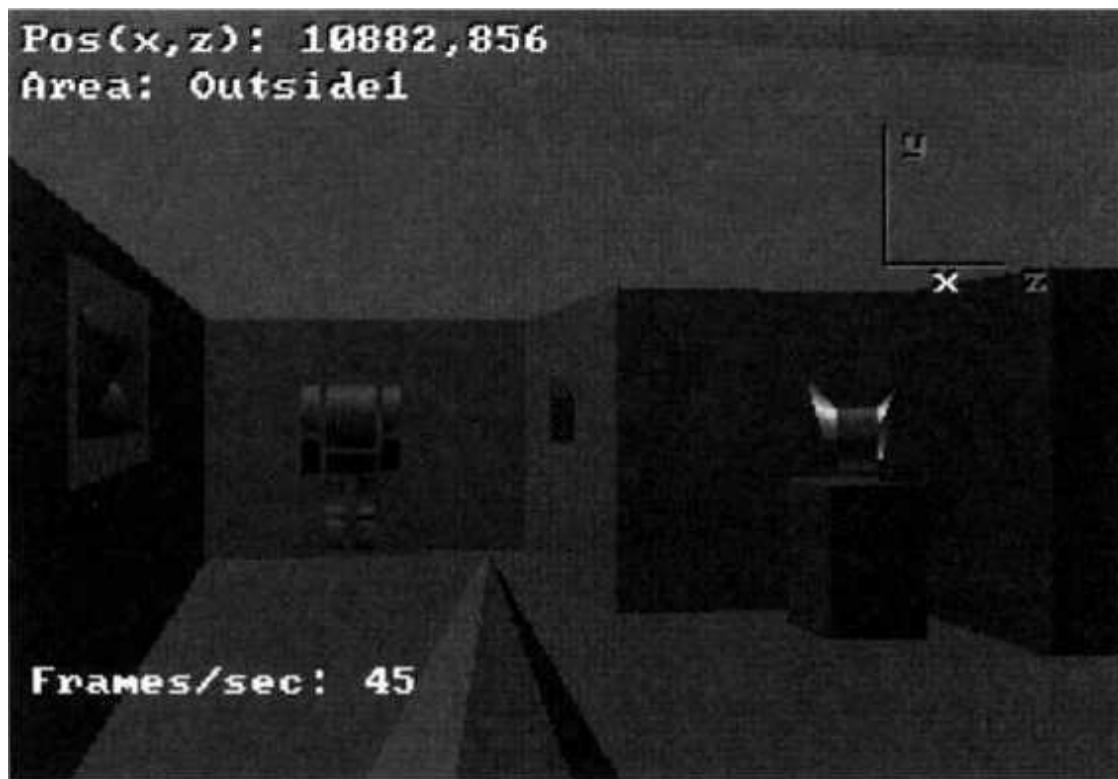


Fig. 6.1 A sample scene produced by Rend386. From Stampe et al. [1993]. © Pearson Education Inc. Reprinted by permission.

Another way to classify toolkits is by whether they are in the public domain (free) or are proprietary. Public domain toolkits, such as Java 3D and VRML, are well suited for laboratory-based VR instruction. Both are used in the VR Teaching Laboratory associated with this book. Proprietary toolkits, such as WorldToolKit, are better documented and support many more I/O devices, such as trackers, sensing gloves, HMDs, and workbenches. This increased functionality comes at a cost (thousands of dollars for licenses).

Table 6.1 summarizes the characteristics of the modern toolkits discussed in this chapter. The interested reader should also consult the annual survey of run-time software packages published by Real Time Graphics. At the time of this writing the survey listed 17 proprietary packages rated according to over 120 criteria [Anon, 2002].

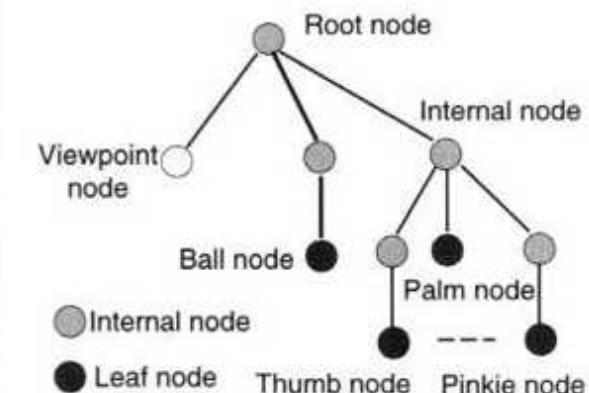
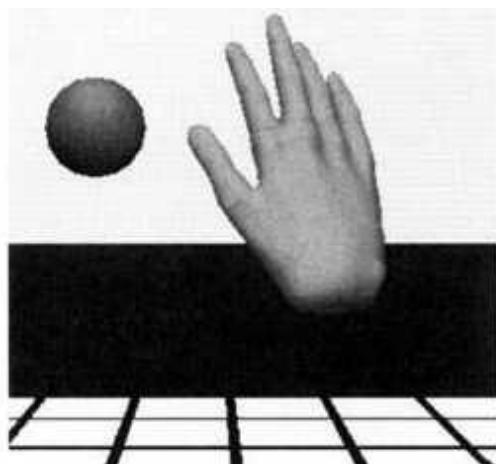
Our discussion of VR toolkits follows the steps of the authoring pathway, namely modeling geometry, linking objects to input devices, defining intelligent behavior, and providing networking. A common authoring task required in both graphics and haptics programming is the construction of a scene graph.

Definition A scene graph is a hierarchical organization of objects (visible or not) in the virtual world (or universe), together with the view to that world.

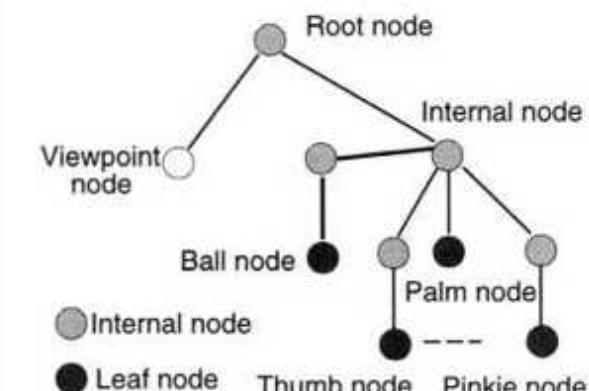
The scene graph represents a tree structure, with nodes connected by branches. The topmost node in the hierarchy is the root, which is the parent node to the whole graph. It connects downward to multiple internal nodes, which in turn are parents to other internal nodes or to external nodes. The external nodes, also called leaves, have no children [Moller and Haines, 1999]. They represent visible objects with their properties (geometry, appearance, behavior). Internal nodes typically represent transformations, which position their children objects in space in relation to the root node or in relation to other objects. Any transformation applied to a given internal node will affect all its children.

TABLE 6.1. Toolkit Classification

Toolkit Name	Application Domain	Proprietary	I/O Devices Supported	Library Size
WorldToolKit (EDS/Sense8)	General purpose	Yes	Most	>1000 C functions
Java 3D (Sun Microsystems)	General purpose	No	Mouse, keyboard	C and Java (275 classes)
GHOST (SensAble Technologies)	Haptics	Yes	PHANToM	C++
PeopleShop (Boston Dynamics)	Military	Yes	Mouse, joystick	C/C++



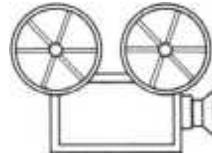
(a)



(b)

Fig. 6.2 A scene graph: (a) before the virtual ball is grasped; (b) after the ball is grasped. © Rutgers University. Reprinted by permission.

Scene graphs are not static, but change to reflect the current state of the virtual world. Such changes may be due to the user's input or to object intelligent behavior (such as the changing position of the arms of a virtual clock). Let us consider, for example, a simulation in which a user wears a sensing glove with a tracker and attempts to grasp a virtual ball. Figure 6.2a illustrates the initial scene showing both the virtual hand and a bouncing ball. In the scene graph the ball leaf node (containing its geometry and color) is a child of an internal node that changes its value to reflect the bouncing effect. After the virtual ball had been grasped, it becomes a child of the palm object, as shown in Figure 6.2b. Now its internal node parent represents its position in the palm. This way any transformation applied to the palm will also affect its children (fingers and ball) to indicate that they move together. The internal node parent of the palm represents the transformation due to the tracker worn by the user.



VC 6.1

The scene graph updates happen once every frame and affect the node attributes. This process is done recursively, from the root node down to the leaves and back to the root node. The downward loop involves changes in the node attributes (including updating transformations), while the upward loop computes the node bounding volumes [Chuang, 2001]. Bounding volumes are bounding boxes that enclose the virtual object represented by the node. The bounding volume of a parent node encloses all the bounding volumes of its children.

Once the scene graph is updated, it is ready for processing by the graphics pipeline. This is called traversal and results in the generation of a new frame. Scene graphs play an important role in speeding up collision detection because they reflect the scene's spatial and temporal coherence (as

discussed in Chapter 5). Furthermore, scene graphs can speed up the graphics pipeline by allowing the CPU to perform view-frustrum culling. In the application stage the CPU performs an intersection check between portions of the scene graph (called subtrees) and the virtual-camera view frustum. If the bounding volume that encloses a subtree is outside the view frustum, all its associated objects are not visible. Therefore they are not sent to the geometry stage of the graphics pipeline for processing. If a bounding volume intersects the view frustum, then its objects are sent down the graphics pipeline and will be subject to clipping in the geometry stage. This bounding-volume hierarchical culling reduces the amount of computation with a resulting increase in the frame refresh rate [Moller and Haines, 1999].

6.2 WORLDTOOLKIT

WorldToolKit is one of the oldest toolkits for VR programming, having been introduced in the early 1990s. Its popularity stems from support of most commercial I/O devices (such as trackers, stereo glasses, and sensing gloves), multiplatform portability (from PCs running Windows, to SGI, HP, and Sun platforms running some form of Unix), and substantial code examples that facilitate setting up a new simulation [Harp, 1999]. The toolkit is mostly used for small to medium-size models owing to the requirement to fit the entire virtual world on available system RAM memory.

Although many of the early toolkits have disappeared, WTK has grown in functionality and size (from 400 functions in 1994 to over 1000 functions in its latest release). The WTK functions use an object-oriented naming convention and are organized in classes. These classes include geometries, nodes, viewpoints, windows, lights, sensors, paths, motion links, and others. A special class is the universe, which manages the simulation and contains all the other objects.

6.2.1 Model Geometry and Appearance

The virtual object geometry can be created in WTK from scratch, using geometry primitives as well as vertices and polygons [Sense8 Co., 1998]. For example,

```
/*using geometry primitives - sphere and cone*/
WTgeometry_newsphere();
WTgeometry_newcone();                                /*custom geometry*/;
WTgeometry_begin();
.....
WTpoly_addvertex();
WTgeometry_save();
```

An alternative is to import already created geometry from AutoCAD and Pro/Engineer authoring tools that output DXF files. Other file formats that can be imported are those produced by 3D Studio (.3DS), Wavefront (.OBJ), Multigen (.FLT), and VRML (.WRL) and neutral file formats (.NFF). For example, if the user created the geometry of a virtual hand and saved it in a file called "hand," then it can be imported in WTK with

```
WTgeometrinode_load(hand);
```

Note that the object geometry needs to be declared as a node in order to be incorporated in the scene graph and thus become visible. The appearance of the object is determined by its material properties (shininess, ambient and diffuse light, and emissiveness for surfaces that emit light in the dark). Such properties are contained in material tables, which are indexed to the number of materials they contain. More than one geometry may point to the same material table. Material properties can also be specified by the authoring tools used to create the surface geometry (e.g., 3D Studio, VRML, etc.). If a material table exists, it can be loaded and (if needed) modified, as in

```
WTmtable_load(filename);
WTmtable_setvalue();
```

Another important object appearance property is surface texture. Textures can be imported into WTK (in RGB, TGA, and JPEG formats) and then

applied to object surfaces. If needed, textures can also be manipulated (translated, rotated, scaled), and blended. For example,

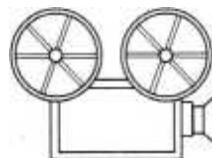
```
WTtexture_load(filename);           /*to import textures*/  
WTtexture_setfilter();             /*to manipulate textures*/  
WTpoly_settexturestyle();          /*textures transparent or  
blended*/
```

6.2.2 The WTK Scene Graph

Figure 6.3a illustrates an example of scene graph implemented in WTK. At each frame the scene graph is traversed from top to bottom and from left to right. As such the nodes will be traversed in the order node A (root), node B (view point), node C, ... , node J. Since the viewpoint node is traversed before any other node (except the root), its value will influence the way all the other visible objects appear in the scene.

WTK allows multiple display windows to coexist, as shown in Figure 6.3b. In this way the user can have multiple views to the same scene. In such a case the composition of the scene graphs is identical, except for the values of node B. Alternatively, the scene graphs will be different, maintaining only a common root node. In this case the user views different portions of the same virtual world (universe).

In Figure 6.3a, nodes F, G, H, and I are siblings because they have the same parent node (E). Since node F is traversed first, its value could influence the way its visible siblings may appear in the scene. Because it is hard for the programmer to foresee all these influences, WTK simplifies the scene graph construction through the use of movable nodes. A movable node is a self-contained structure, grouping separator, transform, and content (geometry, light, level-of-detail) nodes. The separator node prevents the value of the transform node from influencing those of other movable nodes traversed subsequently.



VC 6.2

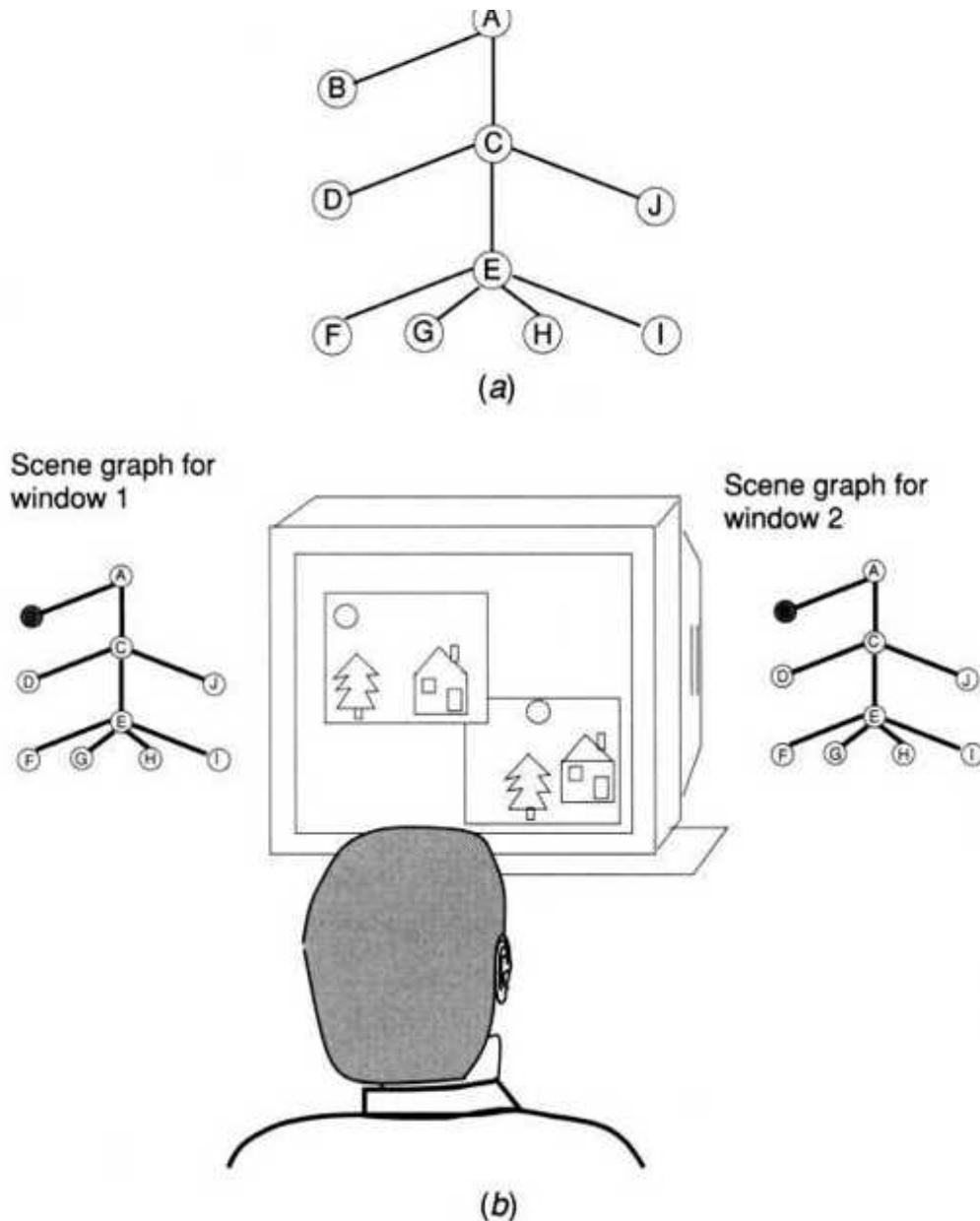


Fig. 6.3 The WTK scene graph: (a) traversal order; (b) multiple scene graphs. Adapted from Sense8 Co. [1998]. Reprinted by permission.

Several movable nodes can be connected in a hierarchy, as illustrated in Figure 6.4. Here the movable nodes are used to construct a virtual hand. The palm movable node is parent to the thumb, index, middle, ring, and small proximal movable nodes. The thumb proximal movable node is parent to the thumb distal movable node, whereas the index proximal node is parent to the

index middle movable node, and so on. The nodes at the bottom of the hierarchy are the finger distal movable nodes. Assuming that the geometry for each of these hand segments exists (in NFF format, for example), it first has to be loaded into WTK. Subsequently the scene graph is constructed by attaching the movable nodes, as in the following example:

```
/*load hand model geometry*/
Palm = WTmovnode_load(Root, ``Palm.nff, '' 1.0);
ThumbProximal = WTmovnode_load(Palm,
  ``ThumbProximal.nff, '' 1.0);
  ThumbDistal = WTmovnode_load(ThumbProximal,
    ``ThumbDistal.nff, '' 1.0);
IndexProximal = WTmovnode_load(Palm,
  ``IndexProximal.nff, '' 1.0);
IndexMiddle = WTmovnode_load(IndexProximal,
  ``IndexMiddle.nff, '' 1.0);
IndexDistal = WTmovnode_load(IndexMiddle,
  ``IndexDistal.nff, '' 1.0);
MiddleProximal = WTmovnode_load(Palm,
  ``MiddleProximal.nff, '' 1.0);
MiddleMiddle = WTmovnode_load(MiddleProximal,
  ``MiddleMiddle.nff, '' 1.0);
MiddleDistal = WTmovnode_load(MiddleMiddle,
  ``MiddleDistal.nff, '' 1.0);
```

```

RingProximal = WTmovnode_load(Palm,
    ``RingProximal.nff, `` 1.0);
RingMiddle = WTmovnode_load(RingProximal,
    ``RingMiddle.nff, `` 1.0);
RingDistal = WTmovnode_load(RingMiddle,
    ``RingDistal.nff, `` 1.0);
SmallProximal = WTmovnode_load(Palm,
    ``SmallProximal.nff, `` 1.0);
SmallMiddle = WTmovnode_load(SmallProximal,
    ``SmallMiddle.nff, `` 1.0);
SmallDistal = WTmovnode_load(SmallMiddle,
    ``SmallDistal.nff, `` 1.0);
                                /*construct the scene graph*/
WTmovnode_attach(Palm,ThumbProximal);
WTmovnode_attach(ThumbProximal,ThumbDistal);
WTmovnode_attach(Palm,IndexProximal);
WTmovnode_attach(IndexProximal,IndexMiddle);
WTmovnode_attach(IndexMiddle,IndexDistal);
WTmovnode_attach(Palm,MiddleProximal);
WTmovnode_attach(MiddleProximal,MiddleMiddle);
WTmovnode_attach(MiddleMiddle,MiddleDistal);
WTmovnode_attach(Palm,RingProximal);
WTmovnode_attach(RingProximal,RingMiddle);
WTmovnode_attach(RingMiddle,RingDistal);
WTmovnode_attach(Palm,SmallProximal);
WTmovnode_attach(SmallProximal,SmallMiddle);
WTmovnode_attach(SmallMiddle,SmallDistal);

```

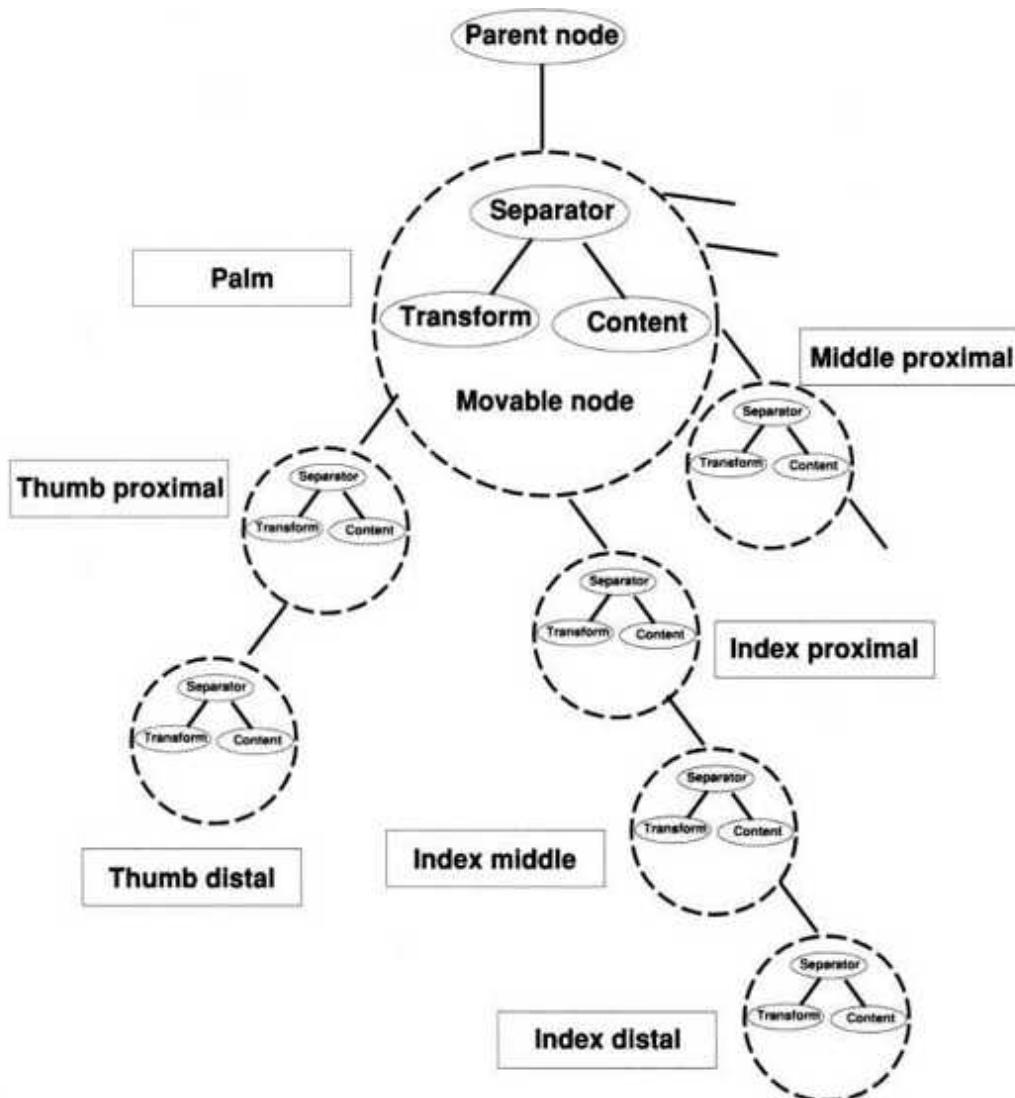


Fig. 6.4 Use of WTK movable nodes to construct a virtual hand. Adapted from Sense8 Co. [1998]. Reprinted by permission.

6.2.3 Sensors and Action Functions

We assume that at this point the universe exists and is populated by various objects (dynamic and static). This is realized by successive calls to `WTuniversenew()` to initialize the universe, followed by `WTuniverseload(filename)` to load the particular world to be simulated. The simulation loop is then entered by the WTK `WTuniversego()` function (for infinite repetitions) and exited with `WTuniversestop`.

During the simulation the user's input is sampled through input devices called sensors. This input is read in real time at each WTK simulation cycle in order to minimize latency. The data are then used to update virtual object position, shape, velocity, etc. The resulting scene is displayed graphically on the user's HMD or some other display. Similarly, audio and haptic feedback are also displayed. Ordering, or scheduling, all these events in real time is an important programming task handled by the underlying toolkit functions.

The universe has user-specified action functions, which are invoked before rendering is done for the current frame. These action functions are set with WTuniversesetaction calls and allow application control of the simulation. Each graphical virtual object can also perform tasks once per frame. Examples of action functions are collision detection and collision response (through sound, haptic, or graphics feedback). Tasks are specified by WTobj ectsettask and model object behavior. Finally, the universe is rendered and the loop starts a new cycle by reading the I/O devices (sensors). The resulting simulation real-time loop is illustrated in Figure 6.5 [Sense8 Co., 1998].

Let us consider the particular case of an immersive simulation in which the user navigates using a trackball. In this case there are two external sensors that need to be monitored. One is the HMD tracker (assumed to be a Polhemus tracker), controlling the view of the universe. The other sensor is the trackball (assumed to be a Space Ball), which changes the viewpoint position in the universe.

WTK treats sensors as objects that generate position, orientation, and other types of data by reading inputs from the real world. Once a sensor is created, it is automatically maintained by the simulation manager, as are the graphical objects to which the sensor is attached. Sensor objects can be created with the generic function called WTsensornew or with WTK device-specific macros. These include the WTPolhemusnew () for the Polhemus tracker and the WTSpaceballnew () for the Space Ball trackball. The "polhemus" and "ball" sensor objects are then created simply with

```

WTsensor *polhemus, *ball;
    /*attach the Polhemus tracker to the serial
       port 1*/
polhemus=WTpolhemus_new(SERIAL1);
    /*attach the Space Ball trackball to the serial
       port 2*/
ball=WTspaceball_new(SERIAL2);

```

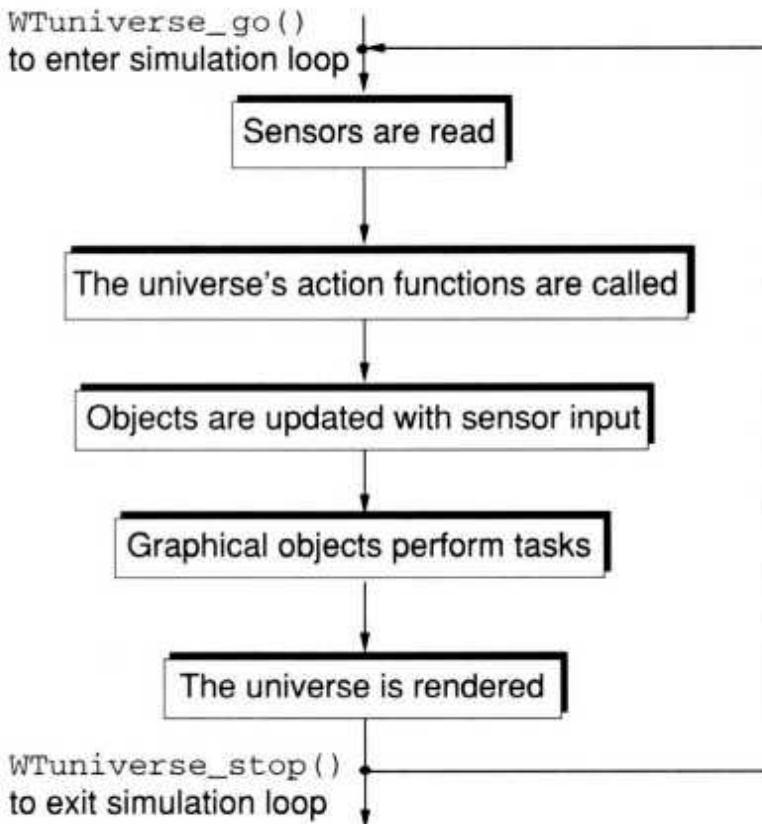


Fig. 6.5 Event scheduling during the WTK simulation loop. Adapted from Sense8 Co. [1998]. Reprinted by permission.

The WTKWTviewpointaddressor () function attaches the simulation view to the external sensor controlling it. If a viewpoint has more than one sensor attached to it, then each sensor input affects the current viewpoint position and orientation. The universe has only one active viewpoint at any given time, and it is accessed by the function WTuniversegetviewpoint (). Therefore, for the two sensors we need successive calls. A small example

integrating these sensors and showing the coding of the run-time loop is given as follows:

```
main() {
WTsensor *polhemus, *ball
Wtnode *root, *scene;
                                /*initialize the universe*/
WTuniverse_new(WTDISPLAY_DEFAULT, WTWINDOW_DEFAULT)
                                /*load scene and link to root node*/
root = WTuniverse_getrootnodes();
scene = WTNODE_load(root, ``myuniverse'', 1.0);
/*attach the Polhemus tracker to the serial port 1*/
polhemus=WTpolhemus_new(SERIAL1);
/*attach the Space Ball trackball to the serial port 2*/
ball=WTspaceball_new(SERIAL2);

                                /*attach viewpoints*/
WTviewpoint_addressor(WTuniverse_getviewpoint(),
    polhemus);
WTviewpoint_addressor(WTuniverse_getviewpoint(),
    spaceball);

                                /*start the simulation*/
WTuniverse_ready();
WTuniverse_go();
/*stop the simulation when user presses a button on the
trackball*/
WTuniverse_delete();
return0;
}
```

6.2.4 WTK Networking

Networking is realized in WTK through its World2World library extension [Sense8 Co., 1997]. Unlike typical client-server architectures in which the server does both administrative and data communication tasks, World2World uses the two-tier clientserver architecture illustrated in Figure 6.6. The server tasks are split between a server manager and multiple simulation servers. The server manager is the initial point of contact for any

new client (user) that requests to be connected to the simulation. Once the client is authenticated, it is assigned to one of the simulation servers for further simulation tasks. In this way the simulation is not disrupted whenever a new client joins the pool of users.

Once handed over to their simulation server, clients register interest in a number of virtual objects. The simulation server then manages the distribution of shared properties by sending update information to interested clients. Furthermore, a simulation server can grant locks on shared groups of objects on a first-come, first-served basis. A client that locks a shared group of objects is the only one allowed to change the properties of those objects. Other clients interested in those same objects receive updates on their state, which continue to be sent by their simulation server.

The World2World communication protocol is a layer on top of the UDP Internet protocol. UDP has lower data traffic than TCP/IP protocols used in client-server communication, because data packets are not acknowledged. UDP is thus a best-effort communication protocol that, unlike TCP/IP, does not guarantee that data is properly received. World2World remedies that by sending an acknowledgment when data are received, bundled with new data packets. In order to reduce the number of updates sent, and thus simulation latencies, World2World uses a global synchronization time. This allows the application to extrapolate at intermediate positions using dead reckoning or smoothing algorithms [Engineering Animation, 1999].

6.3 JAVA 3D

Java, introduced by Sun Microsystems in the mid 1990s, has become the programming environment of choice for platform-independent, highly distributed applications. Java 3D is one of the Java APIs, which was designed for object-oriented programming of interactive 3D graphics applications [Sun Microsystems, 1998]. Similar to WTK, Java 3D uses OpenGL and Direct3D low-level graphics library functions as well as the graphics accelerators that implement them. Unlike WTK, however, Java 3D can be downloaded free from the Web.

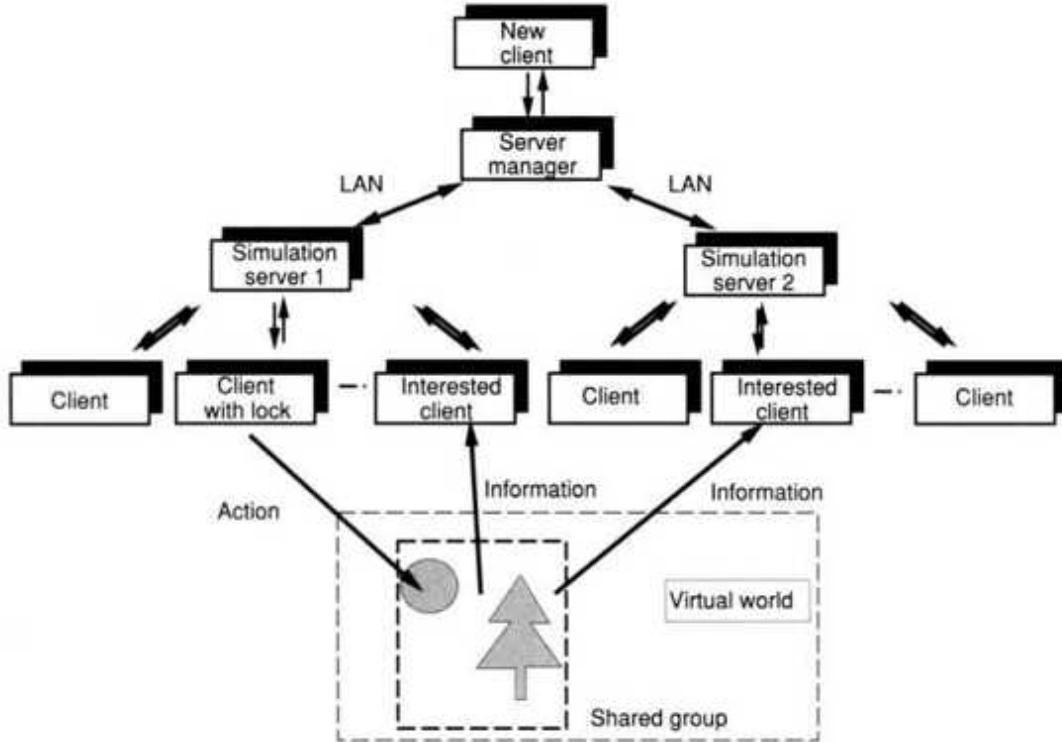


Fig. 6.6 Message passing in WTK. Adapted from Sense8 Co. [1997]. Reprinted by permission.

6.3.1 Model Geometry and Appearance

An object's 3D shape and appearance are specified by the Java 3D class `Shape3D`, which is an extension of the scene-graph leaf nodes. The specific values within the `Shape3D()` class are set by functions called methods, namely `setGeometry()` and `setAppearance()`.

Geometries can be constructed from scratch, using points, lines, triangles, quads, arrays, etc. For example, if a geometry is built using triangle arrays, then

```

        /* Create list of 3D coordinates for vertices*/
point3f[] myCoords = {
    new Point3f 0.0f, 0.0f, 0.0f,
    . . .
}
        /* Create list of vertex normals for lighting*/
Vector3f[] myNormals = {
    new Vector3f 0.0f, 0.0f, 0.0f,
    . . .
}
        /* Create the triangle array*/
TriangleArray myTris = new TriangleArray(
    myCoords.length,
    GeometryArray.COORDINATES |
    GeometryArray.NORMALS);
myTris.setCoordinates(0, myCoords);
myTris.setNormals(0, myNormals);
        /* Assemble the shape*/
Shape3D myShape = new Shape3D(myTris, myAppear);

```

An alternative way of setting object geometry is to import files in formats such as 3DS, DXF, NFF, and WRL. The geometry importing is done by methods called loaders. Java 3D provides loaders for geometry created by Lightwave and Wavefront authoring tools. Loaders for other file formats, such as those created by WTK, or VRML can be downloaded free from www.j3d.org/utilities/loaders.html.

Loaders add the content of the loaded file to the scene graph as a single object. If the object geometry needs to be segmented in order to maintain its intrinsic parentchild dependences, then parts need to be accessed individually by subsequent method calls. Let us consider the case of a virtual hand geometry file Hand.wrl created with VRML. In order to access its parts (fingers) we need to

```

        /* add file to scene graph*/
Scene SC = loader.load(``Hand.wrl'');
BranchGroup Bg = Sc.getSceneGroup();
/* access to the finger subparts of the loaded model*/
Thumb = Bg.getChildAt(0);
        Index = Bg.getChildAt(1);
        Middle = Bg.getChildAt(2);
        Ring = Bg.getChildAt(3);
        Small = Bg.getChildAt(4);

```

The object's appearance (color, rendering mode, texture, transparency, etc.) is specified by the Java 3D Appearance () class. These material and texture attributes need to be defined first and then grouped to form a new appearance as in

```

Mat = new Material();
Mat.setDiffuseColor(r,g,b);
Mat.setAmbientColor(r,g,b);
Mat.setSpecularColor(r,g,b);
                                /* import texture file */
TexLd = new textureLoader(``checkered.jpg'',..);
Tex = TexLd.getTexture();
                                /* create the appearance and set it */
Appr = new Appearance();
Appr.setMaterial(Mat);
Appr.setTexture(Tex);
Geom.setAppearance(Appr);

```

6.3.2 Java 3D Scene Graph

Figure 6.7 [Sowizral and Deering, 1999] illustrates the major components of the Java 3D scene graph. The virtual world, or universe, has a collection of scene graphs (typically one per application). It has a Locale node, which anchors the multiple branch graphs in the world. Only branches that are linked to the Locale are live and thus rendered in the scene. Removing a branch graph reverses the process [Sowizral and Nadeau, 2001].

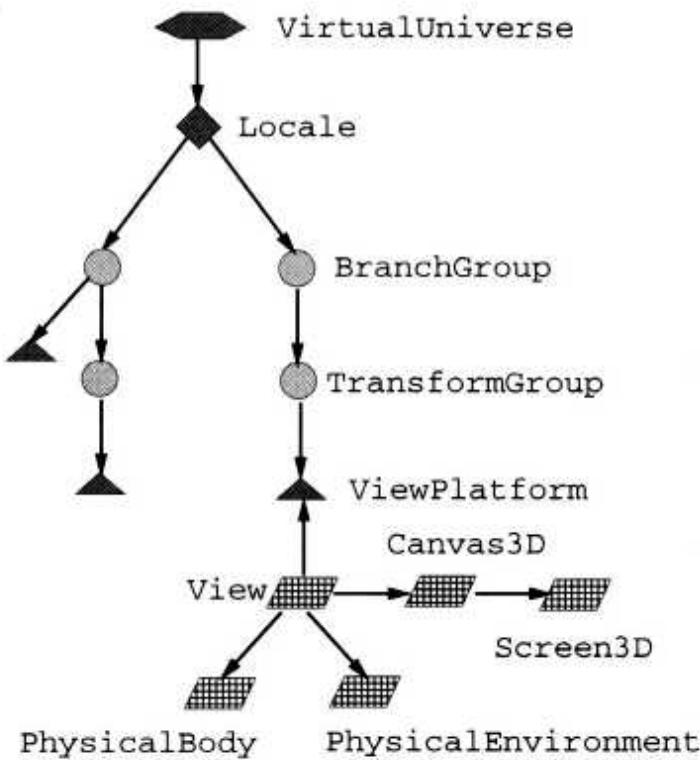


Fig. 6.7 Java 3D scene graph. Adapted from Sowizral and Deering [1999]. © 1999 IEEE. Reprinted by permission.

Each branch graph has a group node called `BranchGroup`, which defines a compilable subgraph and acts like a glue that holds the elements of the branch together. Group nodes position and orient the leaf nodes with `TransformGroup` nodes. `Switch` nodes are used to select among several subgraphs within the group in order to set a level of detail. Finally, `OrderedGroup` nodes set the order in which the branch leaf nodes (objects) are rendered. Thus, unlike WTK, Java 3D has a less structured traversal order of its scene graph.

Leaf nodes do not have children, and contain information or refer to node component objects. A very important leaf node is the `ViewPlatform`, which positions and orients the virtual camera inside the world. A Java 3D application can manipulate the values of the `ViewPlatform` node in order to allow the user to navigate inside the virtual world. The `ViewPlatform`

together with the View object (described later in this chapter) form the Java 3D platform-independent 3D view model.

The Shape3D leaf node contains information about the virtual object's geometry, and appearance (which were previously discussed) as well as the bounding box necessary to perform collision detection. Other leaf nodes specify lighting conditions (AmbientLight, PointLight, DirectionalLight), predefined behaviors, fog, and universe background color.

If a virtual hand needs to be added to the virtual world, then its VRML WRL file needs to be loaded into the scene graph. The following example shows how each segment of a virtual hand can be loaded separately, and subsequently how the parent-child hierarchy is defined between these segments (nodes):

```

/* load geometry from VRML and place it in the Group*/
Palm = loader.load(``Palm.wrl'').getSceneGroup();
ThumbProximal = loader.load(``ThumbProximal.wrl'')
    .getSceneGroup();
ThumbDistal = loader.load(``ThumbDistal'')
    .getSceneGroup();
IndexProximal = loader.load(``IndexProximal.wrl'')
    .getSceneGroup();
IndexMiddle = loader.load(``IndexMiddle.wrl'')
    .getSceneGroup();
IndexDistal = loader.load(``IndexDistal.wrl'')
    .getSceneGroup();
MiddleProximal = loader.load(``MiddleProximal.wrl'')
    .getSceneGroup();
MiddleMiddle = loader.load(``MiddleMiddle.wrl'')
    .getSceneGroup();
MiddleDistal = loader.load(``MiddleDistal.wrl'')
    .getSceneGroup();
RingProximal = loader.load(``RingProximal.wrl'')
    .getSceneGroup();
RingMiddle = loader.load(``RingMiddle.wrl'')
    .getSceneGroup();
RingDistal = loader.load(``RingDistal.wrl'')
    .getSceneGroup();
SmallProximal = loader.load(``SmallProximal.wrl'')
    .getSceneGroup();
SmallMiddle = loader.load(``SmallMiddle.wrl'')
    .getSceneGroup();
SmallDistal = loader.load(``SmallDistal.wrl'')
    .getSceneGroup();

/* Create virtual hand hierarchy */
Palm.addChild(ThumbProximal);
ThumbProximal.addChild(ThumbDistal);
Palm.addChild(IndexProximal);
IndexProximal.addChild(IndexMiddle);
IndexMiddle.addChild(IndexDistal);
Palm.addChild(MiddleProximal);
MiddleProximal.addChild(MiddleMiddle);
MiddleMiddle.addChild(MiddleDistal);
Palm.addChild(Palm);

```

```

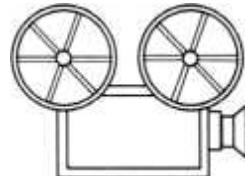
Palm.addChild(RingProximal);
RingProximal.addChild(RingMiddle);
RingMiddle.addChild(RingDistal);
Palm.addChild(SmallProximal);
SmallProximal.addChild(SmallMiddle);
SmallMiddle.addChild(SmallDistal);

```

6.3.3 Sensors and Behaviors

Input to the simulation programmed in Java 3D is provided by a number of 6DOF input devices specified in the class `PhysicalEnvironment`. Unlike WTK, Java 3D does not support trackers, or other VR-specific input devices. Instead, the API provides an input device interface to which device drivers (written by developers in C/C++) need to be linked. The user then defines methods to open, close, and read the specific device as well as set and query its state.

The foregoing approach is taken in order to maintain Java 3D platform independence and generality as a high-level graphics API. This generality is exemplified by the fact that the input device interface accepts data from real as well as virtual devices. This allows the simulation to be driven by data stored in a file, as is customarily done during playback of the user's actions. Alternatively, the simulation can read data from a network connection (remote sensing). Finally, the user can build a "translator" between mouse 2D data and tracker 6D data, such that a mouse becomes a virtual tracker device.



VC 6.3, 6.4

Each `InputDevice` object has associated with it one or more sensor objects. Each sensor object's most recent data are contained in a number of `SensorRead` objects arranged in a circular buffer. `SensorRead` objects contain time-stamped 6DOF (or fewer) data as well as an integer array of pushbutton values [Sowizral and Deering, 1999]. By providing time-

stamped data, Java 3D facilitates the task of averaging sensor data as well as that of prediction based on existing data. The button values indicate whether the button is in a button-up, a button-down, or a trigger state.

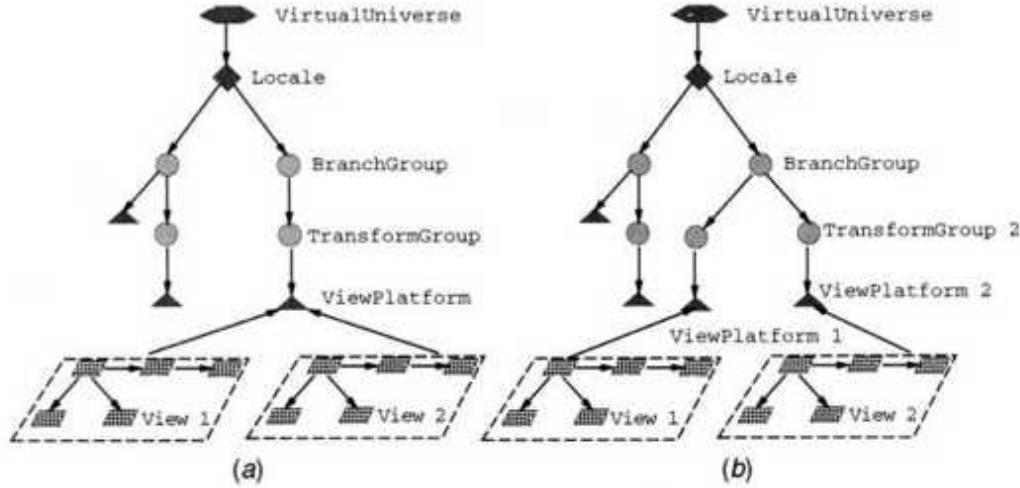


Fig. 6.8 Java 3D scene graph: (a) single ViewPlatform node and multiple Views; (b) multiple ViewPlatform nodes. Adapted from Sowizral and Nadeau [2001]. Reprinted by permission.

The input from the simulation input devices and the output to the graphics and sound displays are associated with the Java 3D View object. It describes the type of graphics display used in the simulation (whether CRT, stereo HMD, or CAVE) specified in the PhysicalEnvironment. Every View has one or more Canvas3D objects, which select the screen area on which it is drawn. This area is part of the screen of a display called Screen3D. The View object also describes the user's initial position relative to the display screen and the tracker availability and position set by the viewing policies. The user's characteristics (such as height and IPD) are specified by the PhysicalBody.

This View model provides a clean separation between the virtual world (provided by the ViewPlatform node) and the real I/O devices used for interaction. This separation allows conservation of simulation code when I/O devices change and is at the core of Java 3D portability. Several users can be tracked simultaneously and be mapped to the same location in the virtual world. This is the case illustrated in Figure 6.8a, where there are

several Views and one ViewPlatform. Conversely, a single user can control several ViewPlatforms, as illustrated in Figure 6.8b [Sowizral and Nadeau, 2001]. Since each ViewPlatform node has its own view (and Screen3D), the user can teleport between remote locations in the virtual world, each corresponding to a different ViewPlatform.

The ViewPlatform form node has an activation radius, which represents the adjacent region of the virtual world. Whenever the user navigates in the simulation, he or she changes the location of the activation radius. Java 3D detects when the activation radius intersects the scheduling bounds of virtual objects that have behavior. These are user-specified bounding volumes (such as spheres or boxes) enclosing objects with behavior. An intersection with the ViewPlatform activation radius triggers a waking up of the behavior specified by the user. The behavior is then executed, as long as it was scheduled, and can result in a scene graph change, an animation, a change in transformations, etc. Several behaviors can be executed in parallel without mutual interference. A specialized behavior is level-of-detail, which is generalized by the Java 3D class LOD. It tracks the ViewPlatform location and computes the distance to an object surface. This distance is then mapped to the appropriate child of a Switch node. The distance-based switching of the level of detail of an object is implemented by the DistanceLOD class.

6.3.4 Java 3D Networking

Java 3D does not provide a built-in solution for networked virtual environments, but relies on Java's powerful network features. This allows Java 3D applications to run as applets in a Web page or alternately as standalone. The necessary reduction in development effort to construct 3D-enabled Web pages is currently being addressed by the virtual reality transfer protocol (VRTP) [Brutzman, 1997]. Currently under research, VRTP combines client-server and peer-to-peer communications optimized for the desktop. It contains an area-of-interest management layer, similar to that of WTK, and is designed to support the distribution of VRML and Java 3D scene graphs.

6.3.5 WTK and Java 3D Performance Comparison

A number of tests were performed by the authors in order to compare the performance of WTK (Release 9) and Java 3D (version 1.3 Beta 1). The rendering was done by a Wildcat 4110 graphics accelerator with 64 MB of frame buffer memory and 64 MB of texture memory. The board was installed on a dual Pentium III 933-MHz PC with 512 MB of RAM running under Windows NT 4.0.

Both toolkits were used to program the scene shown in Figure 6.9. It consisted of a number of rotating spheres (of 420 polygons each) and a virtual hand with 2,270 polygons. By changing the number of spheres it was possible to set the scene complexity at 5K, 10K, 20K, 30K, 40K, and 50K polygons, respectively. The scene was Gouraud-shaded, with either one, or five light sources, and rendered in stereo. The virtual hand was driven by a sensing glove (the Rutgers Master II [Bouzit et al., 2002]) and by a Polhemus tracker. The benchmark simulation ran for either 1,000 frames or for 1 minute, whichever came first.

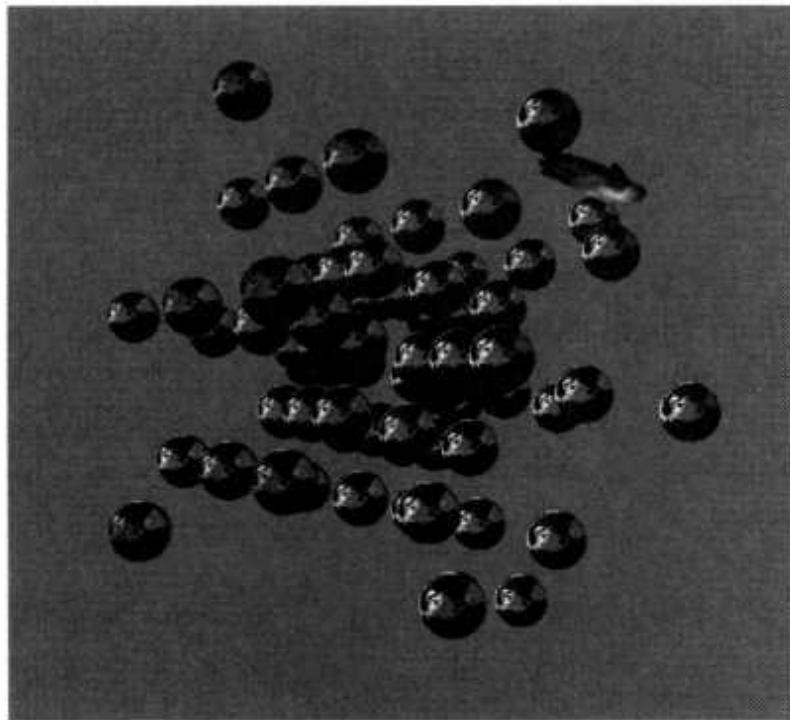


Fig. 6.9 Screen image of the virtual world used for comparison between WTK Release 9 and Java 3D version 1.3 Beta1. From Boian and Burdea [2001]. ©

Rutgers University. Reprinted by permission.

The scene complexity and number of light sources affected the frame rendering time, as illustrated in Figure 6.10. It can be seen that for scenes of fewer than 15,000 polygons WTK performed faster, while for higher polygonal counts Java 3D rendered more frames per second. This was true for both single and multiple light sources in the scene. Interestingly, Java 3D had an almost constant refresh rate of over 30 frames/sec for scene complexities of between 5,000 and 40,000 polygons. By contrast, WTK had a sharp decrease in rendering speed, which at 40,000 polygons is only 10 frames/sec (for five light sources).

Another important parameter for a VR simulation is system latency, which is illustrated in Figure 6.11. In this test latency was measured as the time between the user's input (change in tracker position) and simulation response (change in the rendered scene). As can be seen the system latency increased with scene complexity as well as with the number of light sources. For scenes containing fewer than 15,000 polygons WTK produced lower system latency than Java 3D. However, Java 3D assured a faster system response for scenes with larger polygonal counts. The system latency when using Java 3D and a single light source was almost constant (50 msec) as long as the scene contained 40,000 polygons or less. By comparison WTK produced a sharp increase in system latency, which at 40,000 polygons increased to 80 msec (for scenes with one light source). This represents a 60% degradation in system response compared to Java 3D rendering an identical scene and on an identical hardware setup.

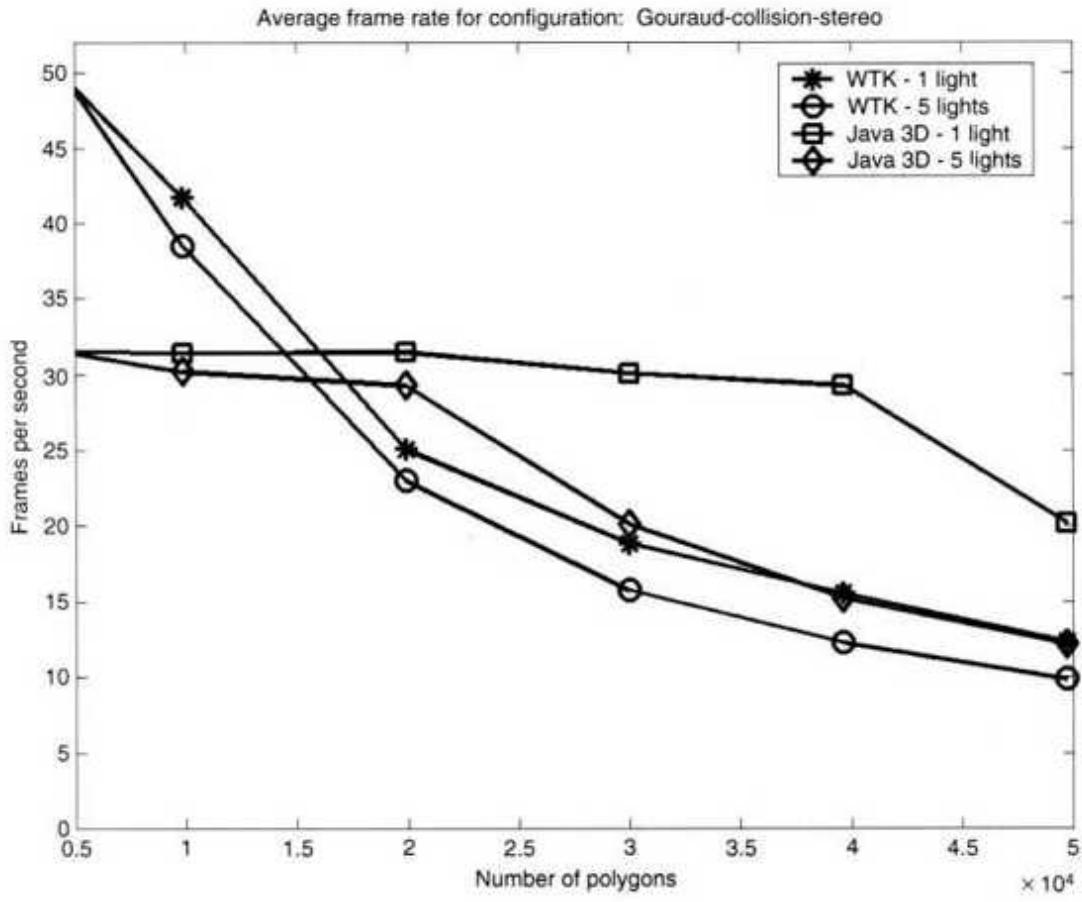


Fig. 6.10 Comparison between the frame refresh rates of WTK Release 9 and Java 3D version 1.3 Beta1 as a function of scene complexity and number of light sources. Programming credit Rares Boian.

Finally, tests were done to determine the variability of frame rendering time over a sequence of 200 frames. These tests measured the time it took the graphics pipeline to output a new frame when all other factors (polygonal count, rendering mode, number of light sources, and user's input) were kept constant. Figure 6.12 plots the frame rendering time for a scene complexity of about 50,000 polygons rendered in stereo and Gouraud-shaded with five light sources. As can be seen from the graphs, WTK produced less variability (smaller standard deviation) than Java 3D. It should be noted that the Java 3D version tested here was a clear improvement over earlier versions, which exhibited even more frame rendering time variability [Boian and Burdea, 2001]. Nevertheless, the Java 3D version 1.3 Beta1 still exhibits occasional spikes of much larger frame

rendering time due to its "garbage collection" mechanism. These spikes can produce occasional freezing of the graphics scene, with unwanted effects on simulation interactivity and the user's feeling of immersion.

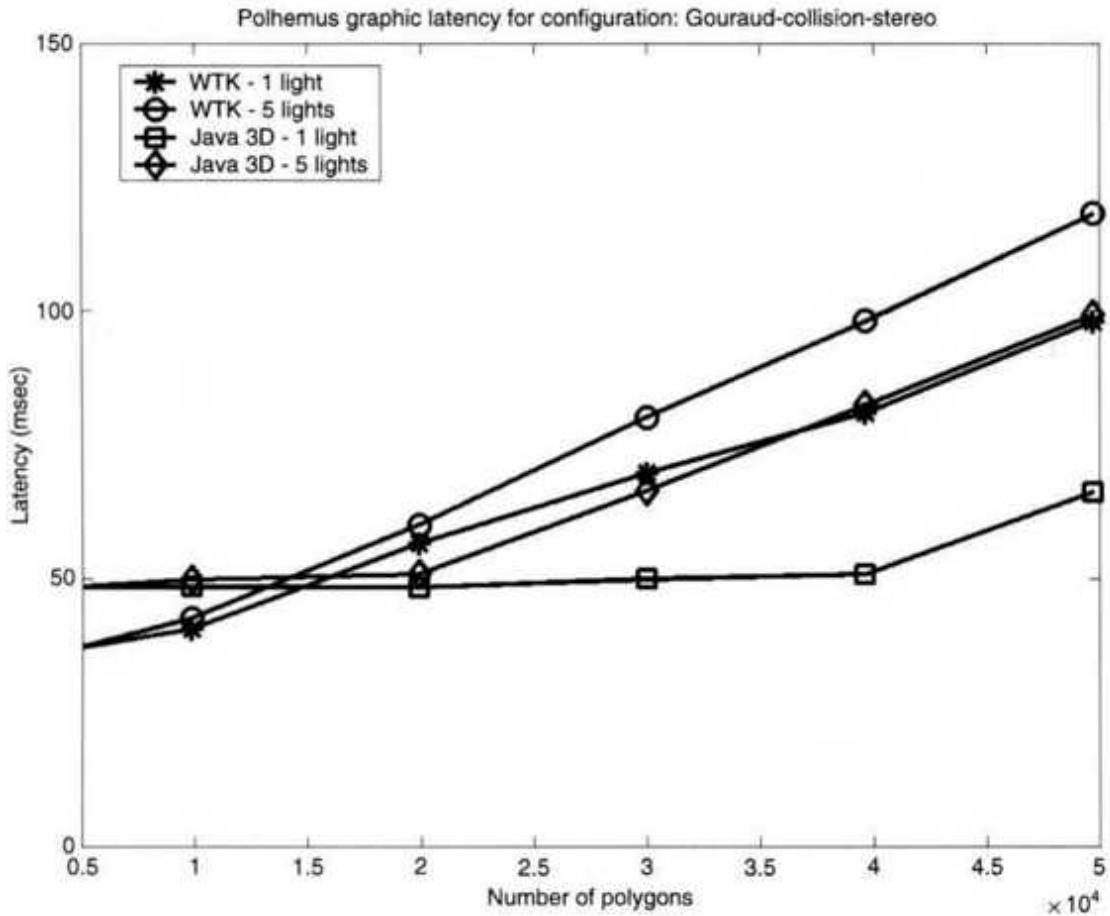


Fig. 6.11 Comparison between the system latencies when using WTK Release 9 and Java 3D version 1.3 Beta1 as a function scene complexity and number of light sources. Programming credit Rares Boian.

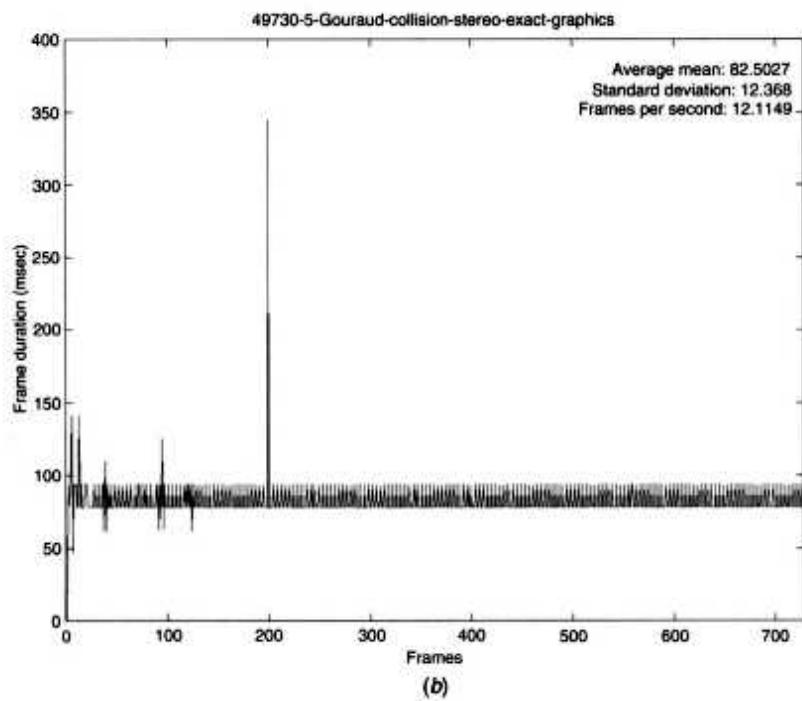
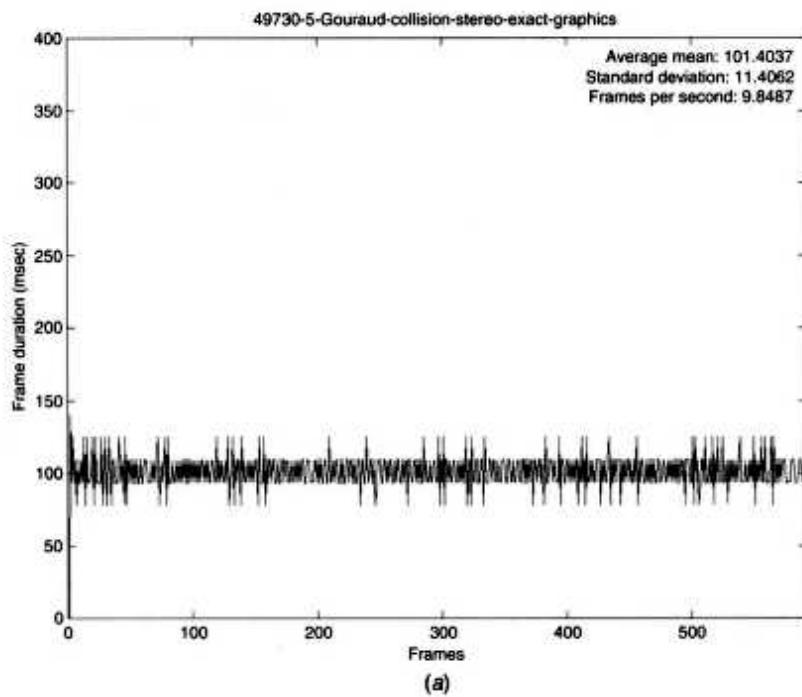


Fig. 6.12 Comparison of the frame rendering time variability for a scene with about 50,000 polygons and five light sources rendered in stereo and Goxirand-Shaded: (a) WTK Release 9; (b) Java 3D version 1.3 Beta1.
Programming credit Rates Boian.

6.4 GENERAL HAPTICS OPEN SOFTWARE TOOLKIT

The WTK and Java 3D toolkits support primarily graphics and 3D sound feedback. They leave the task of integrating haptics to be done by the application developer. Toolkits designed specifically to support commercial haptics interfaces have emerged in recent years. They include the VirtualHand [Virtual Technologies, 2001], used in conjunction with the CyberGrasp and CyberTouch haptic gloves, as well as the User Interface Toolkit [ReachIn Technologies, 2002] for interactions using the PHANToM. Another such toolkit is the General Haptics Open Software Toolkit (GHOST) [SensAble Technologies, 2001a], which is described next.

6.4.1 GHOST Integration with the Graphics Pipeline

GHOST facilitates the programming of haptic effects in simulations that use OpenGLbased graphics pipelines. The VR engine renders the graphics and haptics rendering pipelines as separate processes, as illustrated in Figure 6.13. Preferably each pipeline has its own assigned CPU owing to the large volume of computations involved.

The VR application models its graphics scene graph, traverses it about 30 times every second, and communicates with the haptics process as needed. The haptics loop includes a separate scene graph, which is traversed 1000 times every second (owing to the servo loop requirements of the PHANToM). GHOST manages both haptics scene graph traversal and communication with the application process. It sends input to the servo loop, which runs on the PHANToM control electronics (part of the haptic interface itself). Messages are sent to the application when haptic events occur using graphics callback functions (discussed later in this section).

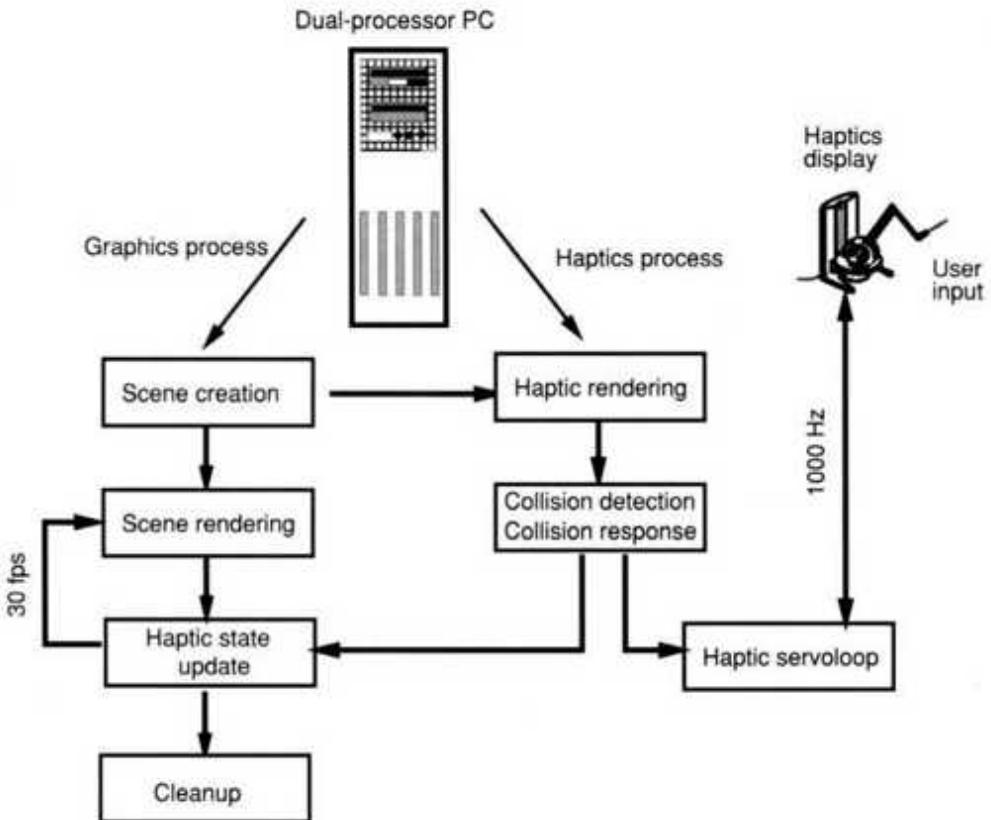


Fig. 6.13 GHOST integration with the graphics pipeline. Adapted from SensAble Technologies [2001a]. Reprinted by permission.

6.4.2 The GHOST Haptics Scene Graph

The haptics environment at the object level is created by the hierarchical collection of nodes within the haptics scene graph. Internal nodes orient and scale their subtree as well as specify the subtree dynamic properties. Such nodes are separator nodes, which create the scene hierarchy. A transformation applied to a separator node will automatically affect its subtree nodes. Leaf nodes specify object surface geometry and its orientation and scaling versus the parent node. Although geometry is not visible, it is necessary in order to compute contact forces. Similar to WTK and Java 3D toolkits, geometries can be imported from VRML files.

A special type of leaf node is that corresponding to the PHANToM interface interaction point. GHOST computes the interaction forces between this point and object geometries (or haptics effects) and sends them to the

servo loop for display by the haptic interface. Another feature specific to this scene graph is that traversal occurs only from top (root node) to bottom (unlike WTK or Java 3D). Each node has only one parent, such that it can be reached by a unique traversal of the scene graph from the root node. As far as haptics is concerned, each object interaction is unique, even if its geometry is not. Therefore, unlike WTK, GHOST does not allow instances of the same object to be used in the scene graph. An example of a GHOST scene graph for the collection of nodes forming a robot is illustrated in Figure 6.14 [SensAble Technologies, 2001a].

GHOST consists of nine C++ classes, namely Abstract node, Geometry node, Dynamic Property node, Haptic Interface Device node, Effects node, Manipulator node, PHANToM Dynamic node, the Scene class, and Data Types.

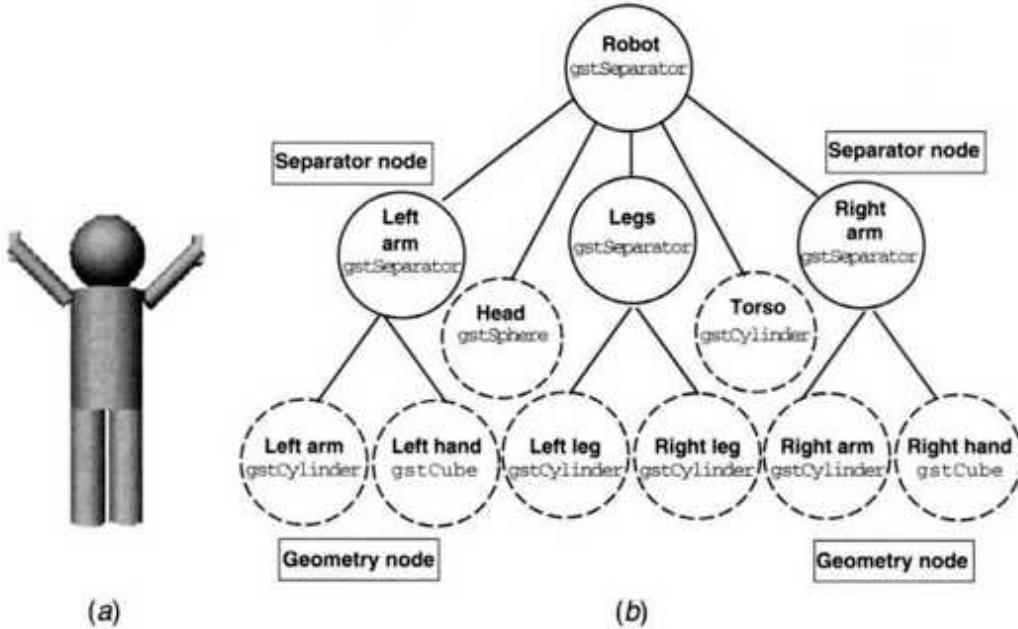


Fig. 6.14 GHOST scene graph for a robot: (a) geometry; (b) scene graph hierarchy. Adapted from SensAble Technologies [2001 a]. Reprinted by permission.

The Abstract node class includes `gstNode` for all scene graph nodes and `gstTransform`, which adds 3D transformations and callbacks to nodes. Also part of this class are `gstBoundedHapticObject` for nodes (objects) that have

a bounding volume and `gs tDynamic` , which adds dynamic attributes to its child subgraph.

The Geometry node class supports two types of geometric representation. The first is geometric primitives, such as `gstCube`, `gstCone`, `gstCylinder`, `gstSphere`, and `gstTorus`. The second type of geometry corresponds to objects built using triangle meshes, supported by `gstTriPolyMeshHaptic`.

Dynamic behaviors can be added to any object geometry, allowing it to move and reorient in response to external forces. The Dynamic Property class includes `gstDial`, `gstSlider`, and `gstRigidBody`, which makes an object exhibit the dynamics of a rigid body. The Dynamic Property node class also includes `gstButton`, which makes an object exhibit a haptic click (as illustrated in Fig. 5.25).

The PHANToM interface is represented in the scene graph by the Haptic Interface node class. It contains the interface state information stored by `gstPHANToM` node as well as `gs tPHANToMSCP` , which represents the surface contact point (described later). It is also possible to change the behavior of the interface when there is no contact with an object. This is done using the Effects node class, such as `gstIner-t i aE f f e c t` , which adds viscosity to the interface endpoint, making it feel more sluggish to the user. The endpoint can be in fact constrained to move on a plane only or along a line, using the `gstConstraintEffect` node. Finally, the PHANToM handle can be made to vibrate by incorporating a `gstBuzzEffect` into the scene graph.

The haptic scene and simulation are managed by the `gstScene` node. In order for the haptics loop to be executed, the haptic scene graph needs to be attached to a `gstScene` and the `startServoLoop()` method needs to be called. The following GHOST example creates a sphere of 20 mm diameter centered at the origin (default). The servo loop runs until the user presses a key which calls the `s topServoLoop()` method:

```

#include <stdlib.h>
#include <gstBasic.h>
#include <gstSphere.h>
#include <gstPHANToM.h>
#include <gstSeparator.h>
#include <gstScene.h>
main()
gstScene *scene = new gstScene;
gstSeparator *root = new gstSeparator;
gstSphere *sphere = new gstSphere;
sphere->setRadius(20);
gstPHANToM *phantom = new gstPHANToM(``PHANToM name``);
root->addChild(phantom);
root->addChild(sphere);
scene->setRoot(root);
scene->startServoLoop();
while(!scene->getDoneServoLoop())
    // end application by calling scene->stopServoLoop();

```

6.4.3 Collision Detection and Response

The GHOST scene graph can contain up to four `gstPHANToM` nodes, which allows four haptic interfaces to be used in one simulation. During scene graph traversal, collision detection is performed between these nodes and the geometry nodes. Whenever a collision occurs, the state of the `gstShape` abstract node changes from untouched to touched. This triggers the change of the scene graph by adding one or several `gstPHANToMSCP` nodes. These nodes are children of their respective `gstPHANToM` nodes, and the number of `gstPHANToMSCP` nodes added depends on the number of collisions detected. Thus up to four surface contact points can coexist in a simulation.

One form of collision response is the force computed by adding surface friction and normal force components. The normal force is computed based on a spring-damper model that takes into account the distance between the SCP and the PHANToM's actual position inside the contacted geometry. The force information is then added to the state of the `gstDynamic` parent node of the contacted object.

Callback functions allow the synchronization of object information contained in the haptics and graphics loops. When the application is ready to render a new scene it issues an updateGraphics call, which queries the state nodes with user-defined graphics callbacks. One such node is gstPHANToM, which changes state if the user moves the interface handle. In this way it is possible to create a visible object (such as a pencil) in the application, which is mapped to the PHANToM device. Callbacks are also associated with gstDynamic nodes. When they change state due to the PHANToM interaction with one of their children, a user-defined callback function is activated. For example, the user can program the application to quit when the state of a gstButton child node changes from pressed to released. Callbacks should also transmit changes in the graphics scene graph back to GHOST. For example, if the simulation calls for an object to break, then its change in topology needs to be transmitted to the haptics loop. The application, however, is not allowed to remove nodes from the GHOST scene graph while the servo loop is running, as it may lead to errors.

6.4.4 Graphics and PHANToM Calibration

The virtual camera view and the PHANToM workspace need to be aligned in order to help the user's feeling of immersion in the simulated world. Thus the PHANToM workspace needs to be centered along the Z axis of the frustum, as illustrated in Figure 6.15 [SensAble Technologies, 2001b]. Furthermore, the PHANToM workspace needs to be centered in the viewing cone (between the clipping planes) such that the objects of interest appear in the scene. This is accomplished by centering the PHANToM workspace with the virtual camera and then moving it by an offset along the Z axis.

In order to be able to move the pencil or other graphics object mapped to the PHANTOM over the whole screen area (top, bottom, left, right), the PHANTOM workspace needs to be scaled. If it is too small (as illustrated in Figure 6.15a), then the cursor can only be moved in a small portion of the screen. Conversely, if it is too large, the cursor disappears from view. The scaling factor is determined by the equation

$$S_{\text{frustrum}} = D_{x\max}/D_{\text{PHANToMmax}} \quad (6.1)$$

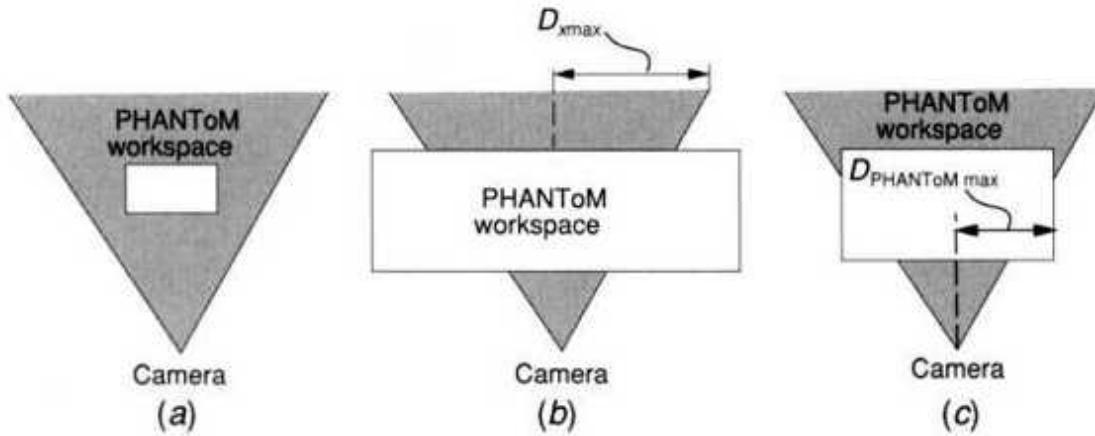


Fig. 6.15 Mapping of the virtual camera viewing cone to the PHANToM workspace: (a) too large; (b) too small; (c) appropriate mapping. Adapted from SensAble Technologies [2001b]. Reprinted by permission.

where $D_{x\max}$ is the distance from the camera focal point to the frustum and $D_{\text{PHANToM}\max}$ is the distance from the center of the nonscaled PHANToM workspace to the side.

An undesirable effect of scaling is that contact forces are scaled, too. As a consequence, `gstShape` node physical properties (compliance and damping) need to be equally scaled, to maintain haptic fidelity. Therefore

$$\text{Surface } K_{\text{spring}}_{\text{new}} = \text{Surface } K_{\text{spring}}_{\text{current}} / S_{\text{frustrum}} \quad (6.2)$$

and

$$\text{Surface } K_{\text{damping}}_{\text{new}} = \text{Surface } K_{\text{damping}}_{\text{current}} / S_{\text{frustrum}} \quad (6.3)$$

where `SurfaceKspring` and `SurfaceKdamping` are `gstShape` compliance and damping parameters respectively.

6.5 PEOPLESOP

Both WTK and Java 3D toolkits are script-based languages which require excellent programming skills when it comes to creating and choreographing

multiple avatars. Driving such avatars in real time is also expensive since each needs input from a sensing suit to create realistic motions.

PeopleShop API is a C/C++ linkable object library built on top of OpenGL or of Direct3D. Unlike WTK, Java 3D, or VRML, PeopleShop allows programming at the task (rather than individual joint) level. Furthermore, it is a graphical programming environment, which constructs simulations by simply choosing from menus of characters, actions, and simulation modes using a computer mouse. This makes it ideal for building civilian and military training scenarios using a large number of avatars without requiring a substantial programming effort.

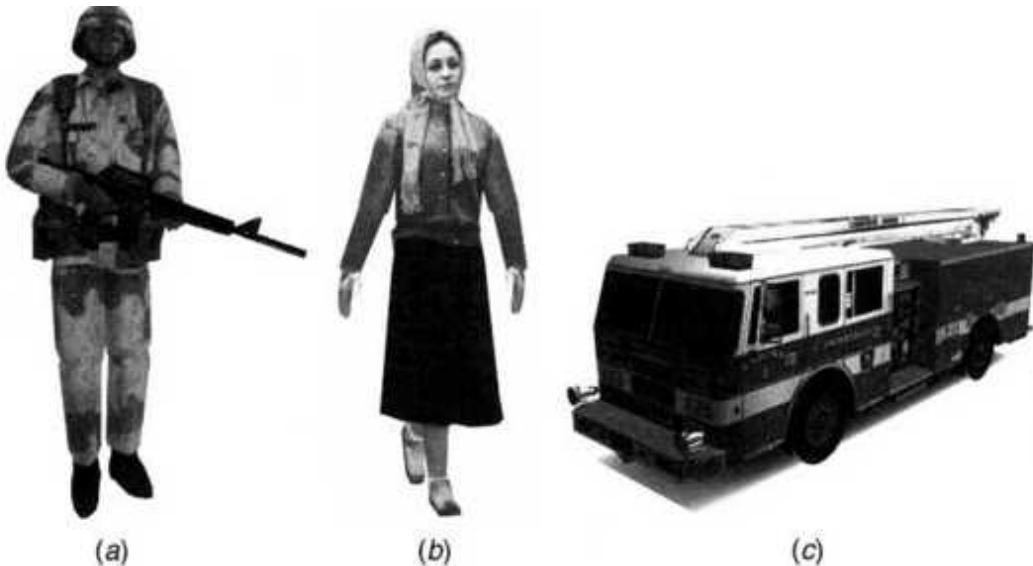


Fig. 6.16 PeopleShop character types and appearance: (a) soldier; (b) civilian; (c) vehicle. From Boston Dynamics [2001]. Reprinted by permission.

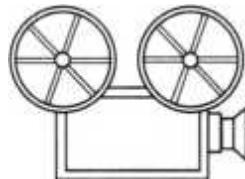
6.5.1 DI-Guy Geometry and Path

Building a PeopleShop program involves first opening a 3D viewing window (called a display palette) and a control palette used to open other palettes and to save the program in .PSS format. Subsequently, people, vehicles, buildings, and so on are added in the simulation by choosing from a library of models, such as those shown in Figure 6.16 [Boston Dynamics, 2001]. These have various appearances (uniform, dress color, camouflage

paint, etc.) created through texturing. Human models (also called DI-Guy's) have a facial expression that shows emotions, move their eyes for gaze, and move their mouth for speech. Models have several levels of detail, in order to speed up graphics rendering.

Avatars are placed in the scene by dragging the model to a desired start location and then specifying a path they will take during the simulation. Paths are spline curves specified with a start point, waypoints, and an endpoint. A character can have one or multiple paths. Figure 6.17 highlights the path of a character in an accident re-creation scenario. The path is shown in white and the waypoints are postlike structures, such as the one next to the car. Each waypoint has a slope adjuster that can be rotated with the mouse in order to change the orientation of the path at that point. Waypoint slope adjusters can be made larger or smaller, thus enlarging or reducing their zone of influence over the path slope.

While PeopleShop characters travel along a path they perform actions, which are selectable from an action palette. Actions depend on the chosen character type. Civilians have traveling actions (such as walking, jogging, crawling) and stationary actions (standing, gesturing, kneeling). Soldier actions include additional options (aiming, firing a weapon, etc.). Actions are symbolized by action beads placed at the start of a character's path and at other locations on the path. When a character arrives at a bead, then that action is executed. Traveling actions have a specified length for execution, while stationary actions have a specified duration. Action beads can be stacked up such that at that location characters can execute several actions at a time. For example, a soldier can have an action bead for walking and another for aiming his weapon. A special action bead is the end bead located at a person's last action. This is symbolized in Figure 6.17 by the dark hexagon stop sign, indicating that the running man will stop by the accident victim.



VC 6.5, 6.6

6.5.2 Sensors and Behaviors

PeopleShop sensors are rectangular volumes of space that are placed at critical locations in the 3D scene using the mouse. They detect the presence of people or vehicles and send a signal to the simulation. Figure 6.18 makes visible the sensor detection volume around the entrance to a building. Thus when the soldier approaches the building entrance, he is detected by enemy forces.

Sensor signals facilitate the coordination of persons or vehicles and allow a character to react to the behavior of others. The basic if-then-else decision logic that creates character behaviors is associated with decision beads placed along the path. When a person or a vehicle crosses a decision bead, their subsequent actions depend on events in the scenario. The location and size of the decision region are specified by the decision bead distance and length. Distance indicates the start of the decision region relative to the start of the path. Length specifies the distance from the decision bead location to the end of the decision zone. A soldier patrolling near the building shown in Figure 6.18 may have a decision zone associated with his beat. While walking inside the length of the decision bead, he will switch paths if the sensor at the building entrance detects an enemy. Complex decision logic can be created by converting to Perl script [Siever et al., 1999] stored in a script bead. The script bead can then be edited to create arbitrarily complex behavior.

From the foregoing discussion it would appear that PeopleShop interactivity is limited to scenario creation and playback, useful in scenario visualization. The level of interactivity is in fact higher since PeopleShop allows external processes to access and change the values of sensor signals. Since PeopleShop scenarios depend on those values, it is possible to interactively change them. A further enhancement in interactivity is related to the ability to export PeopleShop .PSS files to other applications. This is done by using the PeopleShop embedded run-time module, which is placed in the application code. This way a VR application can use HMDs or other displays as well as VR-specific interfaces (sensing gloves, trackers) to

control the virtual camera looking at a PeopleShop scene. This is especially useful in distributed interactive simulations (DIS), which are discussed next.

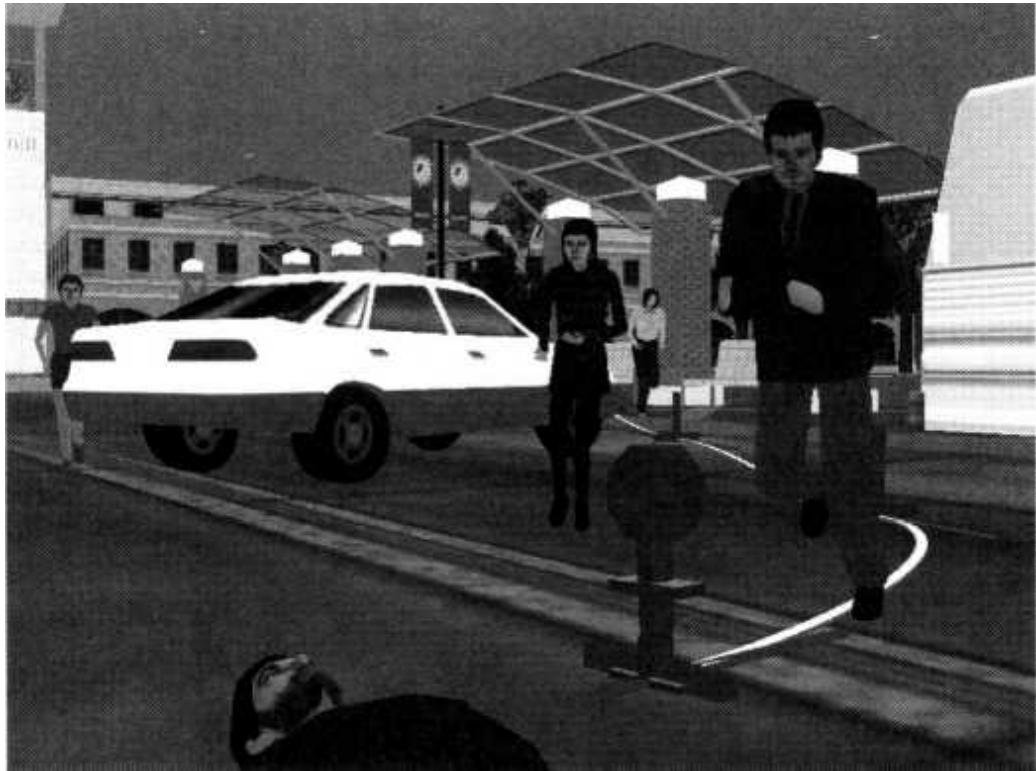
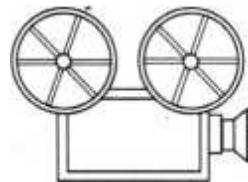


Fig. 6.17 Character path and action bead. From Boston Dynamics [2001]. Reprinted by permission.



VC 6.7

6.5.3 PeopleShop Networking

DIS simulations typically send virtual world state updates over the network in order to maintain simulation coherence. This approach is well suited for vehicles, but ill suited for simulations involving people. A single avatar with 40 joints updated at 20 Hz requires 800 data packets to be sent every second. Such large traffic becomes prohibitive for a large number of participants.

PeopleShop addresses this problem by sending only one or two data packets every second. This is possible by having the packet receiver include a motion generator, as illustrated in Figure 6.19 [Koechling et al., 1997]. The motion generator then translates the high-level commands received over the network into interpolated joint motion of the PeopleShop characters. The packet sender uses a similar motion generator, which inputs data into a stealth display. This shows the simulation as seen at the remote site and its visual feedback allows the user (human in the loop) to compensate for network delays and input device characteristics. The approach is called live reckoning, to distinguish it from the traditional dead reckoning used in DIS vehicle simulations.

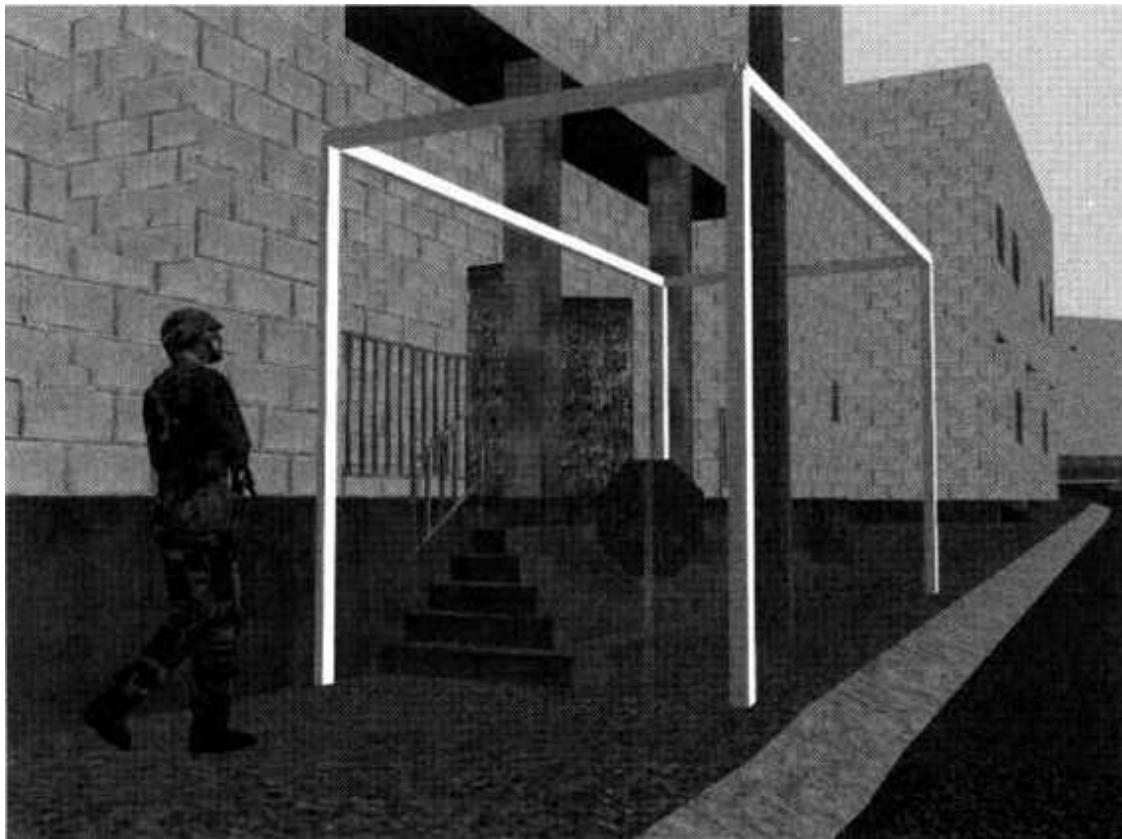


Fig. 6.18 Sensor detection volume made visible. From Boston Dynamics [2001]. Reprinted by permission.

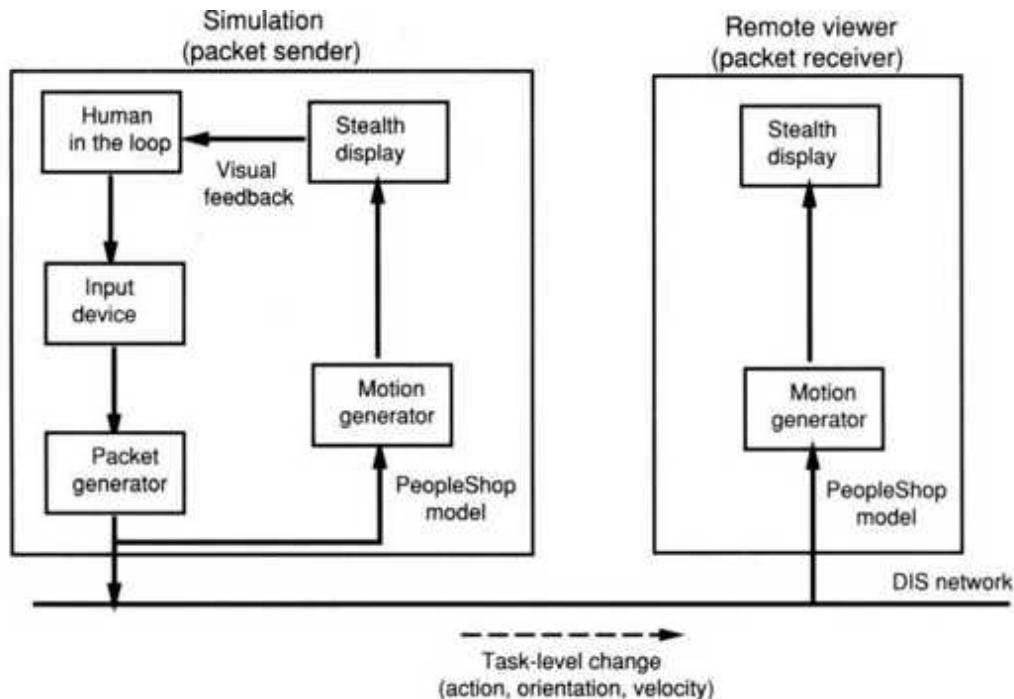


Fig. 6.19 Distributed interactive simulations live reckoning. Adapted from Koechling et al. [1997]. Reprinted by permission.

6.6 CONCLUSION

This chapter introduced a number of VR toolkits designed to help programmers in system integration and application development. These are extremely useful tools and can cut programming time substantially. Commercial packages, such as WTK, support a large variety of high-end I/O devices, allowing advanced graphics and networking for multiuser simulations. Noncommercial toolkits, such as VRML, support less expensive I/O devices and have slower graphics. They are, however, useful for proof-of-concept in initial research stages as well as for teaching students the fundamentals of virtual reality. Java 3D performance has been improving in recent releases and is becoming a powerful VR programming tool, with excellent marketing potential (free). Special-purpose toolkits, such as GHOST (for haptics) or PeopleShop (for military and civilian scenario-based training simulations), help accelerate the application development cycle. Thus they represent useful additions to the VR toolkit family.

6.7 REVIEW QUESTIONS

1. What are VR toolkits? How can we classify them? Give examples of each category and describe advantages and disadvantages.
2. What were some of the early toolkits (early 1990s) and their characteristics?
3. What is a scene graph? Is it limited to graphics?
4. What is the difference between a parent node and a predecessor node?
5. How is object geometry and appearance modeled in WTK?
6. How is the WTK scene graph traversed? What happens when we display two views of the same universe on the same CRT?
7. What are WTL movable nodes and how are they used to model a virtual hand in WTK?
8. What are WTK sensors and how are they used? What are action functions? When are these executed in the WTK real-time loop? Make a drawing and explain.
9. What is a WTK event scheduling loop?
10. What kind of networking is used by WTK and why? Make a drawing and explain. How are shared VEs networked?
11. How can a virtual camera be moved in a WTK scene? Make a drawing and explain. Suppose two users observe the same scene on networked computers. What communication mechanism is used by WTK to transmit changes in the camera viewpoint over the net? How can one user have exclusive (but temporary) control over the camera?
12. What is Java 3D?
13. How are existing geometry files imported in Java 3D?
14. What are Java 3D sensors and behaviors? How is Java 3D networked?

15. In tests done at Rutgers University, what scene was used to compare WTK and Java 3D?
16. What did the frame rendering speed look like in time (number of frames) for Java 3D and why?
17. What is the relation between the GHOST run-time loop and the graphics runtime loop?
18. What are GHOST node classes?
19. How is the haptic scene graph traversed and why? How is the PHANToM represented in this scene graph?
20. How is collision detection done and what is the collision response mechanism in GHOST? Make a drawing and explain.
21. How is the graphics scene updated based on the haptics change of state? How is the interface work volume mapped to the virtual scene?
22. How is the camera fulcrum and PHANToM workspace calibrated? Make a drawing and explain.
23. How is a character path defined in PeopleShop?
24. What sensors and behaviors are used in PeopleShop?
25. What is live reckoning? Make a drawing and explain.

REFERENCES

- Ames, A., D. Nadeau, and J. Moreland, 1997, VRML 2.0 Sourcebook, 2nd ed., Wiley, New York.
- Anon, 2002, "Annual Survey of Run Time Software Packages," Real Time Graphics, Vol. 11(5), pp. 6-13.

Autodesk, 1993, "Cyberspace Developer Kit Concepts & Components," CyTechnical Note #1, Autodisk Inc., Sausalito, CA.

Boian, R., and G. Burdea, 2001, "WolrdToolKit vs. Java 3D: A Performance Comparison," CAIP Center Technical Report 259, Rutgers University, Piscataway. Also online at www.caip.rutgers.edu/vrlab/publications/papers-2001.html.

Boston Dynamics, 2001, People Shop 1.5 User Manual, Boston Dynamics, Cambridge, MA. Also online at www.bdi.com.

Bouzit, M., G. Burdea, V. Popescu, and R. Boian, 2002, "The Rutgers Master II-ND Force Feedback Glove," IEEE/ASME Transactions on Mechatronics, Vol. 7(2), pp. 256263.

Brutzman, D., 1997, "Graphics Internetworking: Bottlenecks and Breakthroughs," in C. Dodsworth (Ed.), Digital Illusions, Addison-Wesley, Reading, MA, pp. 61-97. Also online at [//web.nps.navy.mil/brutzman/vrml/breakthroughs.html](http://web.nps.navy.mil/brutzman/vrml/breakthroughs.html).

Chuang, J., 2001, Introduction to VR, National Chiao Tung University, Hsinchu, Taiwan, Chapter 5. Also online at cggmwww.csie.nctu.edu.tw/course_vr2001/doc/VRcourse5.pdf.

Dimension International, 1993, Virtual Reality Systems Guide, Dimension International, Berkshire, England.

Division, 1993, Amaze User Manual, Division, Ltd., Bristol, U.K.

Engineering Animation, 1999, "World2World-The Industry's Multi-User Client-Server Networking Solution," company brochure, Engineering Animation Inc., Mill Valley, CA.

Harp, G., 1999, "SENSE8/Engineering Animation's WorldToolKit," Real Time Graphics, Vol. 8(3), pp. 10-11.

Koechling, J., A. Crane, and M. Raibert, 1997, "People Are Not Tanks: Live Reckoning for Simulated Dismounted Infantry Using DI-Guy," in

Proceedings of Fall Simulation Interoperability Workshop, Orlando, FL, pp. 781-786.

Moller, T., and E. Haines, 1999, Real-Time Rendering, A. K. Peters, Natick, MA.

ReachIn Technologies, 2002, User Interface Toolkit Programmer's Guide, Reach In Technologies, Stockholm, Sweden. Also online at [//support.reachin.se/Downloadable/documents/UserInterfaceToolkit.pdf](http://support.reachin.se/Downloadable/documents/UserInterfaceToolkit.pdf).

Roehl, B., 1995, "Thinking about Standards," Virtual Reality Special Report, 1995 (November/December), pp. 11-13.

SensAble Technologies, 1999, "GHOST SDK API Version 2.1," SensAble Technologies Inc., Cambridge, MA.

SensAble Technologies, 2001 a, "GHOST SDK API Version 3.1," SensAble Technologies Inc., Cambridge, MA.

SensAble Technologies, 2001b, "Getting to Know GHOST," class notes, SensAble Technologies Inc., Cambridge, MA. Also at www.sensable.com/support/ghost_class.htm

Sense8 Co., 1997, "World2World Release 1 Technical Overview," Sense8 Co., Mill Valley, CA.

Sense8 Co., 1998, "WorldToolKit Release 8 Technical Overview," Sense8 Co., Mill Valley, CA.

Sense8 Co., 2001, "WorldToolKit Release 9," online at www.sense8.com/products/wtk.html.

Siever, E., S. Spainhour, and N. Patwardhan, 1999, Perl in a Nutshell: A Desktop Quick Reference, O'Reilly, Sebastopol, CA.

Sowizral, H., and M. Deering, 1999, "The Java 3D API and Virtual Reality," IEEE Computer Graphics and Applications, 1999 (May/June), pp. 12-15.

Sowizral, H., and D. Nadeau, 2001, "An Introduction to Programming AR and VR Applications in Java 3D," Center for Academic Computing, Pennsylvania State University, online at www.psu.edu/dept/cac/ait/viz/J*ava3d/java3d.htm.

Sowizral, H., K. Rothforth, and M. Deering, 1998, Java 3D API Specification, Addison-Wesley, Reading, MA.

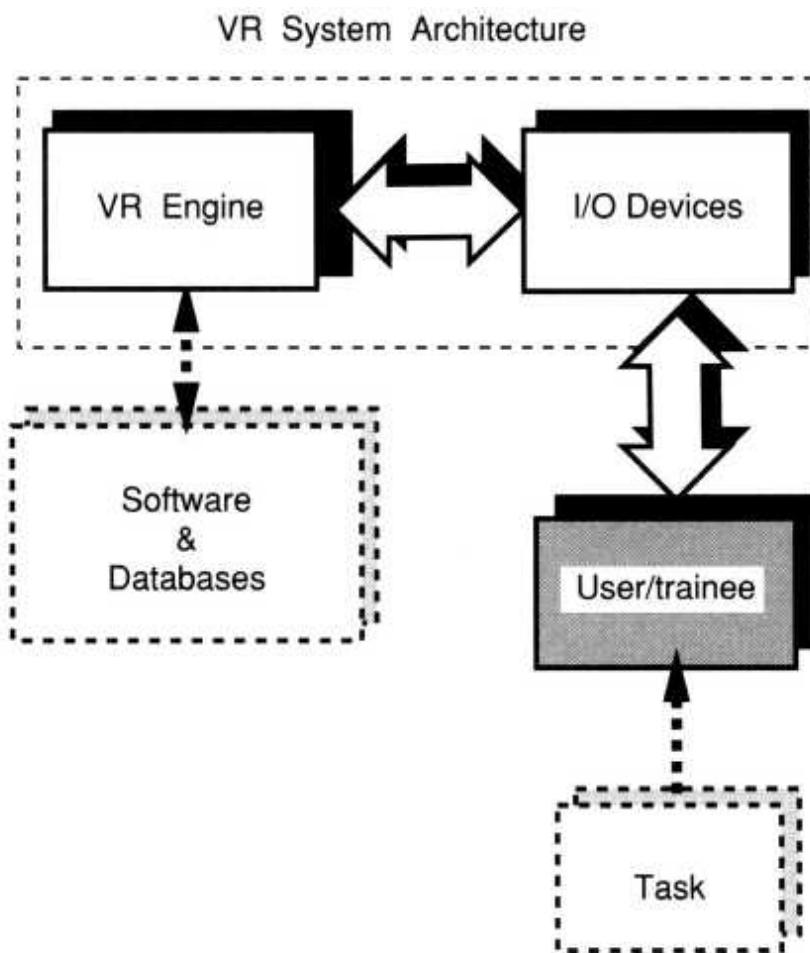
Stampe, D., B. Roehl, and J. Eagan, 1993, Virtual Reality Creations, Waite.

Sun Microsystems, 1998, "Java 3D," company brochure, Sun Microsystems, Palo Alto, CA. Also online at www.sun.com/desktop/java3d.

Virtual Technologies, 2001, "VirtualHand Programmer's Guide v. 2.5, Virtual Technologies, Palo Alto, CA, 2001. Also online at www.immersion.com/manuals/vhsprogrammersguidev2.5.pdf.

CHAPTER 7

HUMAN FACTORS IN VR



The preceding chapters described various VR I/O devices, rendering architectures, the way virtual worlds are modeled, and the essentials of VR programming. It was argued that multimodal interfaces (with graphics, sound, and haptics feedback), realistic modeling, and fast rendering pipelines (high frame refresh rate, reduced system latency), and increased system stability combine to produce quality simulations. It is now time to measure the user's performance when interacting with the simulation. It is also important to gauge the user's response to the technology in order to iteratively improve the VR system or the particular application design.

Furthermore, it is necessary to understand why some user responses lead to simulation sickness, what are its causes, and what can be done to minimize its effects. Finally, at a higher level, it is wise to consider the benefits and negative effects that VR can have with regard to society at large.

These issues fall into the area of human factors research. This consists of systematic studies by multidisciplinary teams of engineers and applied psychologists to gauge which tasks are more suitable for users, which user characteristics influence the VR simulation performance, how VR technology should be improved to better meet user needs, what kind of designs enhance user performance, the negative societal impact from the users' misuse of the technology, and so on. These interrelated human factors research directions are illustrated in Figure 7.1 [Stanney et al., 1998].

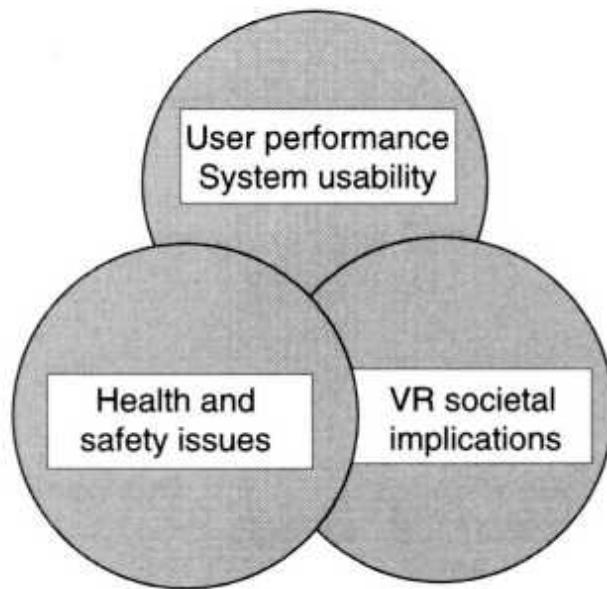


Fig. 7.1 Areas of human factors research in VR. Adapted from Stanney et al. [1998]. © Massachusetts Institute of Technology. Reprinted by permission.

No comprehensive model of human behavior exists owing to its multidimensionality as well as large individual variability. The validity or goodness of a simulation is qualitative at best and cannot be easily quantified mathematically. It is therefore understandable that it is even more difficult to analyze human-machine interaction. The more human parameters are involved (as is the case with VR), the more difficult it is to have a valid

understanding of such an interaction. Thus, determining the performance of a VR simulation is somewhat subjective. Despite these difficulties, the remainder of this chapter presents a structured approach which builds upon an increasing body of human factors research knowledge.

7.1 METHODOLOGY AND TERMINOLOGY

Let us look at the main stages of a VR human factors study, as illustrated in Figure 7.2. Depending on its focus, human factors research can be classified as usability studies, user performance studies, user safety studies, and sociological studies. Usability studies look at ways to improve system or application design in order to make them easy to use. User performance studies are conducted to measure the user's response to a given simulation and a particular hardware system. User safety studies are conducted to better understand simulation sickness causes and effects as well as increase the user's safety (and minimize physical injury). Finally, sociological studies, as their name implies, are designed to measure factors related to effects of VR on society.

Definition VR human factors studies consist of a series of experiments, performed under very rigorous conditions, aimed at determining users' response to VR technology, VR technology usability, VR user safety, and the related societal impact of VR.

Regardless of their particular focus, human factors studies have to adhere to a well-documented experimental protocol. This establishes the structured sequence of experiments that all participants in the study need to perform. This sequence consists of trials, sessions, and rest periods.

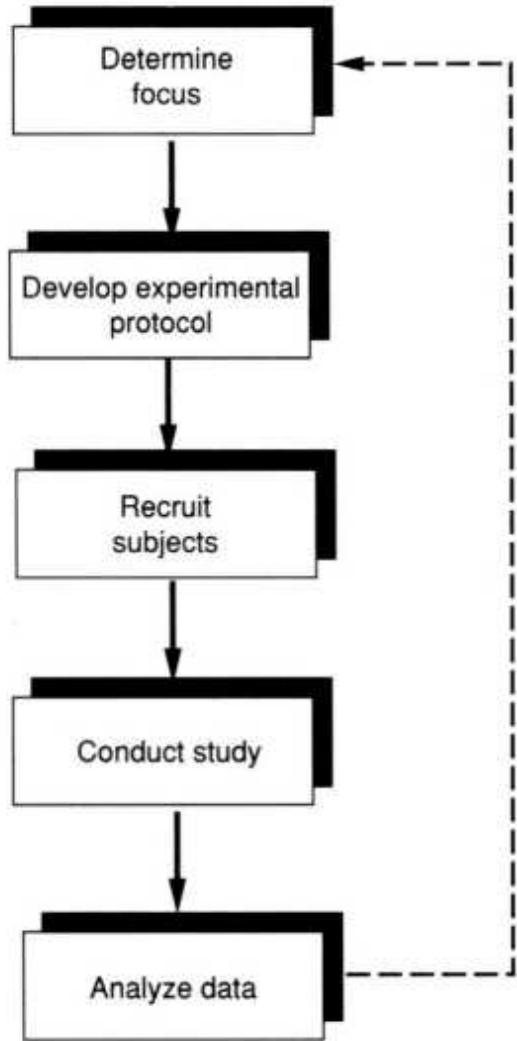


Fig. 7.2 The main stages of a VR human factors study.

Definition A trial represents a single instance of the experiment to be performed as part of a human factors study. A sequence of repeated trials constitutes a session. Sessions (and sometimes trials) are separated by rest periods for the participant in the study.

The experimental protocol details the exact number of trials per session, the number of sessions performed by a participant in a day, and the total number of days the study will last. The experimental protocol further specifies how many participants there will be, depending on both the scope of the study and available budgets.

The participants in the study are called subjects and they need to be recruited before the experiments can commence. The gender and age of the subjects are determined by the particular study objectives. For example, if the study is aimed at determining what role age plays in simulation sickness, then older individuals will be recruited. Conversely, if the study is aimed at gauging the response of pregnant women to a simulation, then they will be enlisted, and so on. Small human factor studies, which enlist a few subjects, are called case studies. Studies that use a larger number of subjects are called control studies.

Definition A control study divides the subjects into experimental and control groups. The subjects in the experimental group perform the experiments as specified in the protocol, while the subjects in the control group do not. They are used as a basis of comparison.

Sometimes the experimental group is divided into subgroups, each assigned a slightly different simulation (different interaction technique, different graphics hardware, different task accuracy requirements, etc.). This is done in order to test the influence of the different variables on the subjects. In this case the study is said to use a between-subjects (experimental) design. Conversely, it is possible to ask all subjects to perform all tasks (under all conditions), in which case the study uses a within-subject design. Finally, there are cases where the number of variables is so large that the study requires both between-subjects and within-subject designs. An example is the virtual environment (VE) testbed for selection and placement tasks [Bowman et al., 2001] discussed later in this chapter.

All subjects participating in a human factors study, regardless of its design, need to sign a consent form. This document informs the subject of the scope of study, the benefits and potential risks, and the right of the subject to withdraw from the study at any time without fear of retribution. Consent forms include additional information indicating the name and contact information for the principal investigator, who is the person responsible for the overall study. These forms also indicate the address and contact person for the Institutional Review Board (IRB), an office within the research

organization conducting the study that is mandated by law to oversee the proper conduct of human factors studies. No such study can begin without an approval from the IRB, which is typically given for a year and then renewed/updated.

Before the start of the study, subjects are recruited through advertisements, targeted e-mails, Web postings, support/focus groups, and other such venues. During the recruitment process subjects are screened for suitability for the study. For example, poor vision would preclude a subject from being enlisted in a study aimed at graphics-oriented research. Subjects are then assigned a code to protect their identity and privacy and sign a release for use of data (including photographs) in research, publications, recruitment of other subjects, etc. At this time baseline measures of the subject's abilities are taken and stored in a database. The same database will store experimental data measured during the trials. The particular types of variables and the frequency of data sampling are set in the experimental protocol. Following the completion of the trials some more measurements may be needed. These posttrial measurements are taken immediately upon completion of the trials or at weeks or months thereafter. These are done to gauge what lasting effects (either positive or negative) exist from the VR exposure. Subjects may also be asked to fill out a subjective evaluation questionnaire, as explained later.

7.1.1 Data Collection and Analysis

Virtual reality has great advantages compared to classical (paper and pencil) methods of data collection [Lampton et al., 2002]. First, the amount, temporality, and diversity of data that can be sampled during trials using a VR system are much larger than those obtained by manual recordings. Second, VR systems allow researchers to have a comprehensive view of all subject's actions while immersed in the simulation and to do so from varying viewpoints. Third, the subjects' actions can be recorded online and played back during task debriefing (as is regularly done in military training). Lastly, researchers need not be colocated with the subjects owing to the use of LANs and WANs and distributed virtual environments.

For all its advantages, VR technology has definite drawbacks that have to be taken into account when designing the experimental protocol. These drawbacks can have a negative impact on the quality of the subject's performance measures. Such performance measures need to have sensitivity, meaning that they need to be capable of discriminating between novice and expert subjects performing a given task. Performance measures need to be reliable, that is, they need to be repeatable and have internal consistency. A performance measure is repeatable if it returns the same value when measurements are done over different trials and at different times. Repeatability by itself does not assure validity of data. Validity means the data are truthful to the subject's actions, and is usually determined by independent expert judgment. VR hardware and software problems have a negative impact on the quality of performance measures taken during the simulation. Large system latencies, for example, will adversely affect both the reliability and the validity of experimental data obtained during the study. This is true especially for studies where a subject's reaction time needs to be determined. Sensor noise (such as that associated with 3D trackers, detailed in Chapter 2) will make it harder to distinguish between novices and experts in a task involving precise motion, for example. Thus it will affect the sensitivity of such performance measures. Sensitivity is also affected by the large variability of available VR hardware and software. Some subjects will have more experience than others with computers, some will have more experience with HMDs, etc. Space here does not allow us to mention all possible VR-related factors that affect the quality of the experimental design; rather we want to alert the reader to this important issue. The discussion that follows assumes that the experimental protocol design was done correctly and minimized the foregoing drawbacks through careful hardware/software selection and subject screening.

The last stage of the human factors study is to analyze the data stored in the experimental database. This data analysis usually uses the analysis of variation (ANOVA) [Stockburger, 1996], which determines whether there are statistically significant differences between data corresponding to different trials or different conditions. Experimental findings are then used to fine tune the interface design, the control algorithm, or the application

features. Sometimes these findings uncover new problems, which lead to a follow-up study, as symbolized by the dashed arrow in Figure 7.2.

Subject performance measures are based on objective and subjective criteria. Objective criteria refer to such variables as subject's task completion time, error rates, and task learning time. Subjective criteria refer to the subject's expressed preference (or lack of same) for a given interface device, control modality, or application feature. Subjective criteria also involve the degree of perceived difficulty or fatigue experienced when using a given interface or a simulation application.

Definition Task completion time represents the time span between the subject starting and ending a particular action (or sequence of actions) constituting the task.

Time is measured from the moment a subject performs the given action, for example, when first touching a virtual object or when first seeing a moving target. The end of the experiment is also linked to an action, such as releasing the virtual object, hitting a target, etc. Time can be measured online (using the system clock on the computer running the experiment) or offline with a stopwatch. Alternatively, time can be measured through the processing of a signal from a sensor actuated by the subject [Howe and Kontarinis, 1992].

Definition Task error rate measures the type, magnitude, and frequency of errors made by the subject when performing the simulation task. What constitutes an error is of course task-dependent and is established by the experimental protocol.

An error could be, for example, the breakage of a "fragile" virtual object manipulated by the subject or the placement of an object outside a given target area. For a given task, errors vary from trial to trial and from subject to subject. It is necessary to perform error averaging over all subjects within a group and determine what the standard deviation is relative to this average. The smaller the standard deviation, the more uniformly the subjects performed during the experiment.

Figure 7.3a shows a hypothetical curve illustrating the variation of task error rates (or of task completion time) over repeated trials. When tasks are done repeatedly there is generally a decrease in error rates and completion times over all subjects. This reduction with the trial count represents the subject's task learning process. The steeper the curve, the quicker is the task learning process. Furthermore, learning leads to a reduction in the standard deviation (SD) of the measured variable since there is more uniformity among subjects' performance.

Figure 7.3b illustrates the effects of prior learning, meaning the knowledge accumulated by subjects prior to the study. The hypothetical Group A data show that they did not learn the VR task between the first and second trials. There is subsequent learning, as shown by the decrease in error rates, task completion times, and group standard deviation at trial 3. The question is, what did the subjects learn? The subjects in group A had a prior mental image of the task. If that prior image conflicted with the way the simulation presented the task, then subjects did not learn the task. Instead they learned how to better use the simulation system.

Group B's prior learning of the experimental task did not conflict with the way it was presented in the VR simulation. Since these subjects already knew the task well, it had very low difficulty for them. As a consequence, Group B has a small error rate/task completion time, which stays constant with repeated trials. The Group B results also had a small standard deviation, which remained so with repeated trials. An extreme case (not shown in Figure 7.3b) is a group that shows consistently high error rates and task completion times and large standard deviations. For such a group the task is too difficult to perform. In such cases the experimental protocol needs to be changed to reduce the task difficulty accordingly.

Task completion time, error rate, and task learning time represent performance criteria that are applicable to VR simulations in general. There are, however, performance measures that are (feedback) modality-specific or specific to a particular type of study. One example of modality-specific measure is the average contact force, or torque [Das et al., 1992], when a subject is interacting with a virtual object. This is given by

$$N \text{ Average force} = 'N f \quad (7.1)$$

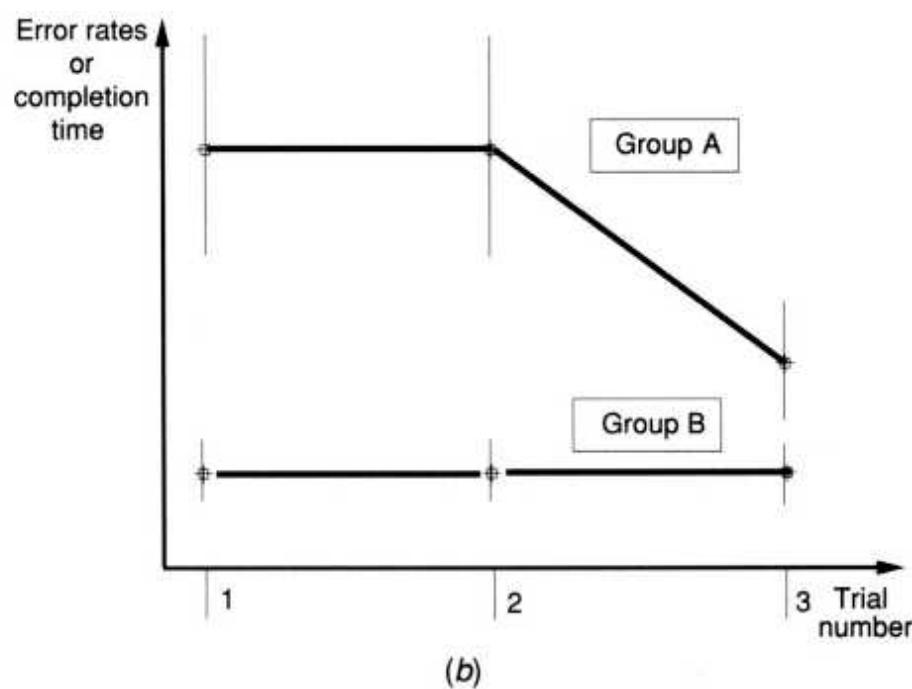
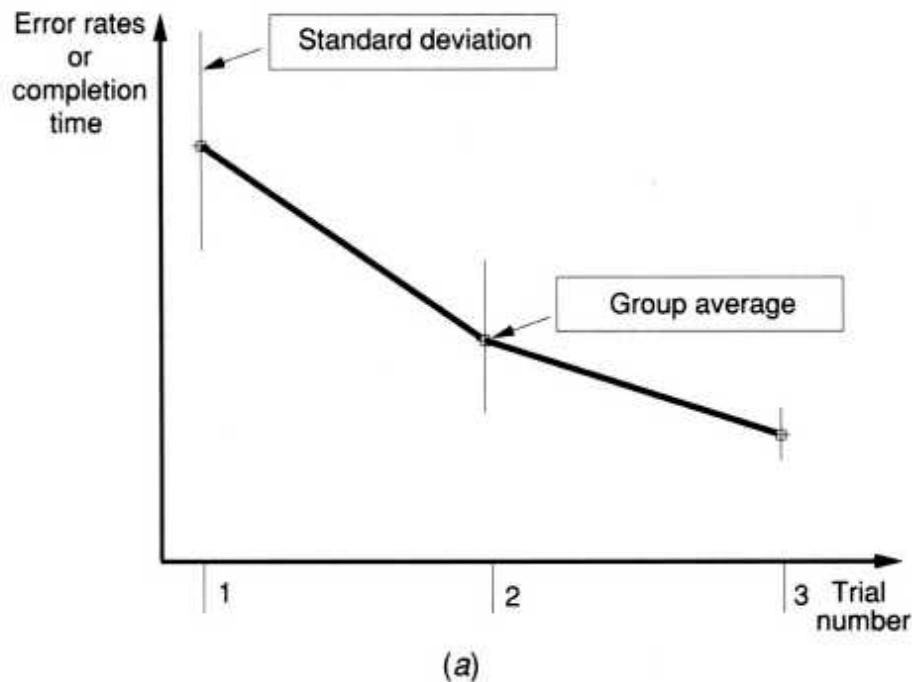


Fig. 7.3 Graph showing task learning effects: (a) normal learning process in VR; (b) prior knowledge effects.

where N is the number of data samples in the contact phase and f_i is the magnitude of the i th force (assuming all forces are positive). High average force represents a forceful manipulation that may not be adequate when the task involves fragile virtual objects. Related performance criteria are the peak contact force and the force variance [Hannaford and Wood, 1989].

Another performance measure specific to haptic feedback is the cumulative contact force/torque. This is obtained by multiplying the force (or torque) magnitude by the sampling interval over the contact time, as in

$$N \text{ Cumulative force} = J \text{ At} \quad (7.2)$$

where At is the sampling interval and N is the number of data samples, as before. The higher the cumulative force/torque, the greater is the subject's muscle exertion. High cumulative forces can thus lead to user fatigue and high haptic interface energy consumption and wear.

Examples of task-specific performance measures are those associated with cognitive tasks. Such studies supplement data from the VR system by sampling electrodes placed over different areas of the subject's body. Their electrical signals (voltages, currents, or frequencies) can measure brain waves (through electroencephalography, EEG), heart rate (through electrocardiograms, EKGs), and muscle tone (through electromyography, EMG). These variables are used together with other measures (such as breathing rate or electrodermal activity, EDA) to determine the degree of a user's immersion (or presence) in the virtual world, mental workload, stress, phobias, and other cognitive variables.

7.1.2 Usability Engineering Methodology

A subclass of human factors research is formed by usability studies, conducted to determine the ease (or difficulty) of use of a given product. Within the scope of this book, we are interested in usability evaluations of VR interface devices or VR applications. Usability studies differ from general-purpose VR human factors studies, which are more theoretical and limited in scope to a particular feedback modality, interaction technique, or

system characteristic. By contrast, usability engineering is product-oriented, iterative in nature, and part of the product development cycle.

In view of the differences in scope and approach, it is understandable that the methodology followed by usability engineering studies is specific to them. Furthermore, VR usability engineering is more involved than methods used in classic usability evaluations, which deal with 2D (windows and menus) rather than 3D multimodal environments. Finally, the theory behind VE usability methodologies is an area of active research, and there are no clear standards.

Hix and Gabbard [2002] developed a methodology of conducting VR usability studies. As illustrated in Figure 7.4a, their methodology consists of four stages: user task analysis, expert guidelines-based evaluation, formative usability evaluation, and summative evaluation. In what follows we discuss this methodology and the way it was applied in the study of a military command and control application called Dragon.

User task analysis is the first stage of a usability study. It identifies and describes the tasks and compiles a list of user actions and system resources needed to accomplish those tasks. The techniques used involve questionnaires, interviews with typical users, direct observation, and analysis of technical documentation (technical specifications or previous systems). The results of the task analysis need to identify not only actions, but also interrelationships (dependences and order sequences) between such actions. Thus, proper analysis results in structured understanding of the task, including the user's information flow during the execution of that task. Poor (missing or incomplete) task analysis is a frequent cause of bad product design (a product that is difficult to use). In extreme cases products may be impossible to use if they place physical or cognitive demands that surpass the subject's capabilities.

The Dragon military command and control simulation involves a digital map that users (unit commanders) need to interact with during battle. As shown in Figure 7.4b, the map depicts a 3D view of the battlefield (in this case an amphibious assault), with 3D icons for airplanes, ships, tanks, trucks, etc. This differs from classic military maps, which are 2D and use

2D symbols of friendly and enemy units drawn on transparent (acetate) overlays. In the Dragon system such overlays are not necessary since the virtual map is dynamically updated by the computer running the simulation and the graphics feedback is presented on a workbench-type display. During battle, users need to be able to navigate (change view) and select certain symbols in order to manipulate them or receive text-based data (location, strength, readiness status, friend/foe, etc.). It is intuitive that the ease of use (degree of usability) of such an application is critical in view of the consequences of human error in lost lives. Furthermore, the situation is complicated by the stress under which the military commander operates.

Expert guidelines-based evaluation (sometimes called heuristic evaluation) is the second stage of usability studies. It aims at identifying potential usability problems early in the design cycle and indicate why the particular components/techniques are problematic. This is a pencil-and-paper comparison of the user's actions during the task to established guidelines. Several evaluators (experts) first inspect the design alone. Subsequently, they communicate as a group in order to determine where there is consensus and what differences exist in their individual findings.

The Dragon expert evaluation concentrated on the critical task of navigation. Ease of navigation was identified as key feature since it conditioned other tasks (such as object selection, query, and manipulation of 3D icons). Navigation here was characterized as either egocentric (the user is surrounded by the environment) or exocentric (the user is outside the environment, looking in). During the evaluation, experts interacted with the simulation using a flight stick, which is a wand with pushbuttons and an imbedded 3D tracker. They found problems with the system responsiveness (detailed later in this chapter) as well as poor functionality of the interface during exocentric navigation. Some of these problems were remedied before proceeding with the next stage of the evaluation.

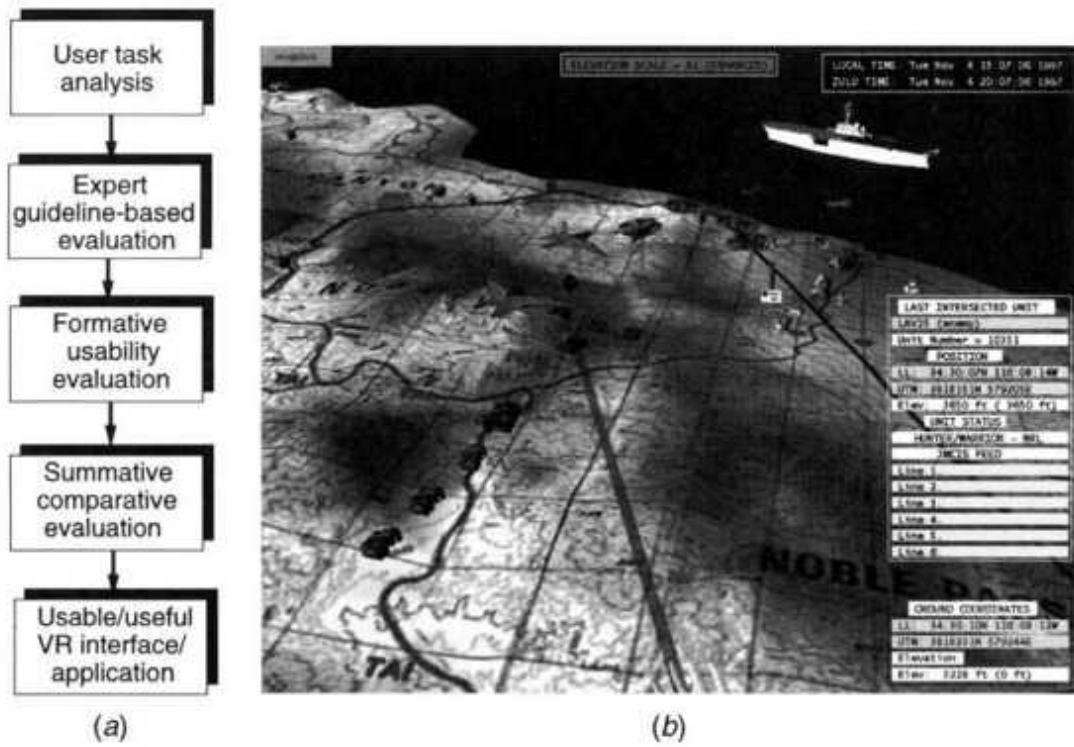


Fig. 7.4 Usability engineering methodology. (a) Main stages. From Hix and Gabbard [2002]. Reprinted by permission. (b) Military command and control application example. From Naval Research Laboratory [2002]. Reprinted by permission.

Formative usability evaluation is an observational evaluation method aimed at iteratively assessing and improving product usability. This is done by placing representative users in task-based scenarios while collecting qualitative and quantitative data (task completion time and error rates). These observations are then used in product redesign and the evaluation is repeated.

The formative evaluation of Dragon involved two stages of design. In the early design cycle, the evaluations were done to select the best interface for navigation. The three candidates were Pinch Gloves, voice recognition, and the wand. In a typical multiuser scenario, the Pinch Gloves proved to work well with right-hand-dominant subjects and produced time delays when transferring the glove from one subject to another. Voice recognition also proved ineffective (slow and unreliable) as an interface. Thus the wand was selected as the interface for navigation.

The second round of Dragon formative evaluations involved a large number of subjects and were done to refine the wand usage. Each subject had to perform six formative evaluations ranging from 20 minutes to more than 1 hour. During this time, two or three evaluators timed individual subtasks, and recorded errors. These errors were associated with moving the wand in the wrong direction, not being able to rotate the digital map, problems with navigation degrees of freedom, etc. In the end it was found that interactions needed a combination of egocentric and exocentric navigation, depending on the particular needs of the battlefield scenario. A large effort was spent on proper mapping of wand pushbuttons to navigation pan, zoom, rotate, heading, pitch, and roll. Pan and zoom were mapped to the wand trigger and pitch and heading to the left pushbutton, while exocentric rotate and zoom were coupled with the right wand pushbutton.

Summative evaluations are performed at the end of the product development cycle. They statistically compare the new product with similar systems in order to determine which is better (which assures more productivity at lower costs). Summative evaluation thus allows the selection of the best product (or simulation system) among several candidates using field trials and expert reviews.

The summative evaluation of Dragon involved the study of four parameters: navigation metaphor (egocentric or exocentric), gesture mapping (rate or position control of the viewpoint based on the user's hand movement), visual display device (workbench, desktop monitor, display wall, or CAVE), and graphics mode (stereoscopic or monoscopic). The study used a combination of between-subjects and within-subject design, as shown in Figure 7.5 [Naval Research Laboratory, 2002]. Thirty-two subjects participating in the study were divided into groups of four. Each group was assigned a different combination of stereo viewing (on, off), control movement, and navigation metaphor (exocentric or egocentric) conditions. The within-subject component of the design changed the graphics display presenting the digital map. Thus each subject had to repeat the navigation task on a map displayed either by a CAVE (GROTTO), a wall, a workbench, or a desk-top monitor. Users where fastest where using the monitor and slowest on the work bench [Swan et al., 2003].

Stereo viewing		On				Off			
Control movement		Rate		Position		Rate		Position	
Frame of reference		Ego	Exo	Ego	Exo	Ego	Exo	Ego	Exo
Between subjects	Within subject	GROTTO	Subjects 5–8	Subjects 9–12	Subjects 13–16	Subjects 17–20	Subjects 21–24	Subjects 25–28	Subjects 29–32
		Wall							
		Workbench							
		Desktop							
Computer platform		Subjects 1–4							

Fig. 7.5 Experimental design used in the summative evaluation of the Dragon military command and control simulation. From Navy Research Laboratory [2002]. Reprinted by permission.

7.2 USER PERFORMANCE STUDIES

Evaluating a subject's performance during interactions with virtual worlds is a complex endeavor due to its dependence on many factors. These include the particular virtual world simulated (its complexity), the user's characteristics (age, prior computer or task knowledge), system characteristics (graphics mode, latency, I/O devices used), as well as the task characteristics and interaction techniques used.

7.2.1 Testbed Evaluation of Universal VR Tasks

Researchers have proposed to use testbeds as a way to deal with these evaluation complexities [Lampton et al., 1994; Bowman et al., 2001]. VR testbeds are composed of a small number of universal VR tasks. These tasks, such as travel in an environment, object selection, and object manipulation, can be found in most VR applications. While more time-consuming and expensive than other types of human factors studies, testbeds provide a structured way to model subject performance. By constructing task taxonomies and analyzing performance at the subtask level, testbeds make it possible to predict a subject's performance in applications that incorporate those tasks, subtasks, and interaction techniques.

Figure 7.6a shows a scene from the testbed evaluation of the user's performance during navigation tasks developed by Bowman and his colleagues [2001]. The environment contained several types of obstacles (such as fences and trees) that could be placed randomly such that travel from start to target was not immediate. Similarly, the targets (flags placed in the environment) could be randomly changed. The 38 subjects who completed the study (32 men and 6 women) were divided into seven groups of at least 5 subjects each. Each subject group traveled using a different interaction technique. These were steering-based, manipulation-based, and target-specification techniques. The subjects in the three steering-based navigation groups used pointing, gaze tracking, or torso tracking, respectively. Manipulation-based travel used HOMER or go-go techniques. HOMER interaction involves ray casting to select an object and manipulation to reach it. The go-go technique allows subjects to stretch a virtual hand much further in the VR than natural reach (using nonlinear mapping), grasp an object, and pull the view of the virtual camera forward. Finally, targetspecification techniques allow subjects to be moved by the simulation to a specified target object. Such techniques used either ray casting or dragging of an icon over a digital map overimposed on the scene.

Figure 7.6b shows the variation in the group mean travel time during a first-time exploration of the environment. It can be seen that the fastest exploration (or shortest time to reach the target) corresponds to the use of gaze-directed or pointing interaction to change the virtual camera viewpoint of the simulation. The group that took longest to reach the target was that using the digital map dragging interaction. This may be explained by the indirect navigation method (map) and the higher cognitive demand of this technique. The gaze-directed interaction technique, while allowing four-times faster navigation, did produce eye strain and nausea in some subjects. This was due to the rapid change in the virtual camera viewpoint, coupled with the scene presentation on a low-resolution (VGA) monoscopic (binocular) HMD.

Figure 7.7a shows a scene from the testbed evaluation of the subject's performance during object selection and placement tasks [Bowman et al., 2001]. Subjects were asked to select a highlighted object (the darker box)

from a 3 x 3 array of similar objects and then place it between two targets (the textured boxes to the right). Forty-eight subjects (31 men and 17 women) were divided among nine different groups, using a between-subjects and within-subject experimental design. The between-subjects component used different interaction techniques for each group. One group used the go-go-type interaction previously described. The other eight groups had varied selection, attachment, and positioning techniques. Object selection was done either by ray casting (extending a ray from the subject to the selected object) or occlusion. Attachment to the object was done either by moving the virtual hand to the object or scaling the user so that his or her hand touched the selected object. Finally, object positioning used either a linear mapping between the subject's hand and the manipulated object motion or pushbuttons to move the object closer or further away. The scene was viewed through the same monoscopic HMD previously described. The withinsubject variables were the ratio of object size to target size (which was either 1.5 or 3.75 times larger), the number of DOF used during box manipulation (two or six), and the distance from the subject at which the object had to be placed (3, 6, and 10 m).

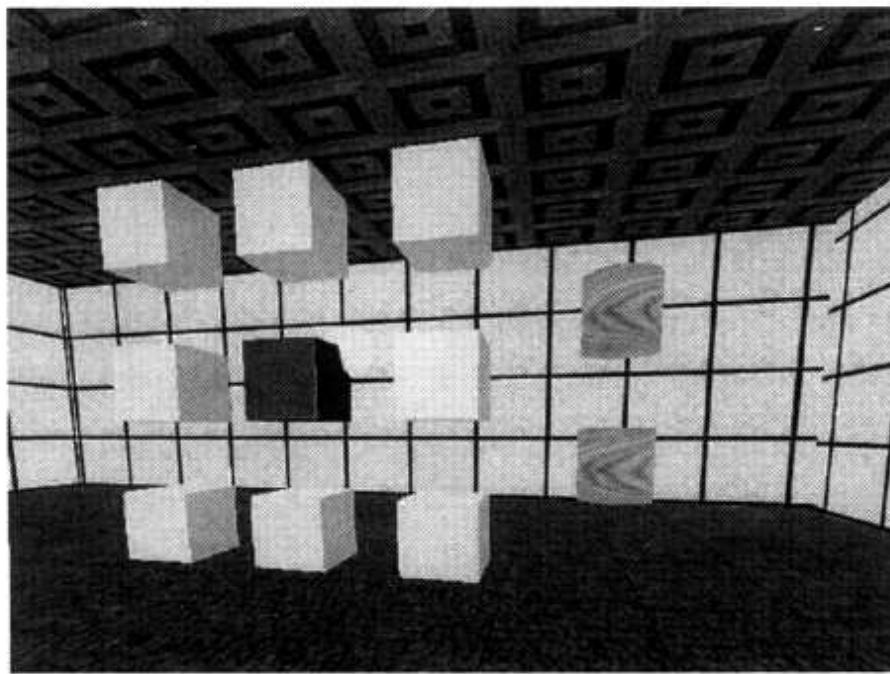


(a)

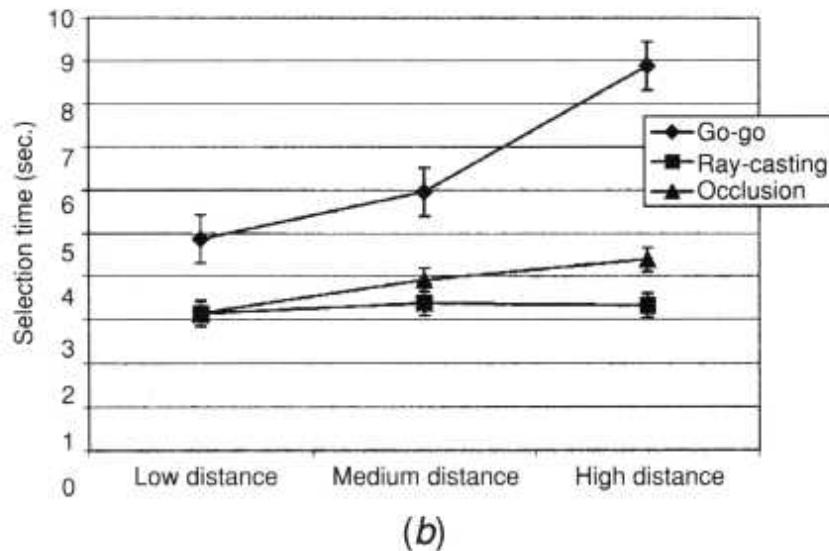
Technique	Think time	Travel time	Total time
Gaze-directed	2.16	18.28	20.44
Pointing	2.20	22.33	24.53
Torso-directed	2.77	27.00	29.77
HOMER	4.20	37.66	41.86
Map dragging	29.54	52.39	81.93
Ray-casting	1.86	34.95	36.81
Go-go	3.29	21.48	24.77

(b)

Fig. 7.6 Testbed evaluation of navigation and searching tasks: (a) graphics scene; (b) thinking and travel time (in sec.) as a function of interaction technique. From Bowman et al. [2001]. © 2001 Massachusetts Institute of Technology. Reprinted by permission.



(a)



(b)

Fig. 7.7 Testbed evaluation of object selection and placement tasks: (a) graphics scene; (b) selection time as a function of distance to object and interaction technique. From Bowman et al. [2001]. © 2001 Massachusetts Institute of Technology. Reprinted by permission.

Figure 7.7b shows the influence of the distance to the selected object and of the interaction technique on selection time. It can be seen that distant

objects were harder to select, the go-go technique being significantly slower than either ray casting or occlusion. Larger boxes were easier to select than smaller ones, regardless of selection technique, with the go-go technique benefiting most from an increase in selected object size. Results showed that subjects had difficulty completing the placement of the box within the target area when the manipulated object had six DOF. As a consequence, it took on average four times longer to complete the task under these conditions.

7.2.2 Influence of System Responsiveness on User Performance

One important characteristic influencing the user's performance is system responsiveness. The responsiveness of a VR system is inversely proportional to the time between the user's input and the simulation response to that input. The longer such a delay, the less responsive is the system and the more difficult it is to perform a VR task on that system.

System responsiveness (SR) [Watson et al., 1998] groups time delays due to hardware latency as well as the user's cognitive load. One important component of the hardware latency is the time taken to render a new image (or frame). As discussed in Chapter 5, a given graphics pipeline will take longer to render a complex scene than it would take to render a simpler environment. This is equivalent to a lower frame refresh rate for the complex scene, all else being equal. The problem that needs to be studied is the dependence of the image-understanding processes on this scene refresh rate. The movie industry relies on the fact that perceived continuity of visual motion requires a refresh rate larger than the eye's sampling rate (16 Hz). Above 25-30 images/sec the brain is fooled into perceiving a continuous motion. It is therefore interesting to see what happens to the user at low frame refresh rates (below 25 frames/sec).

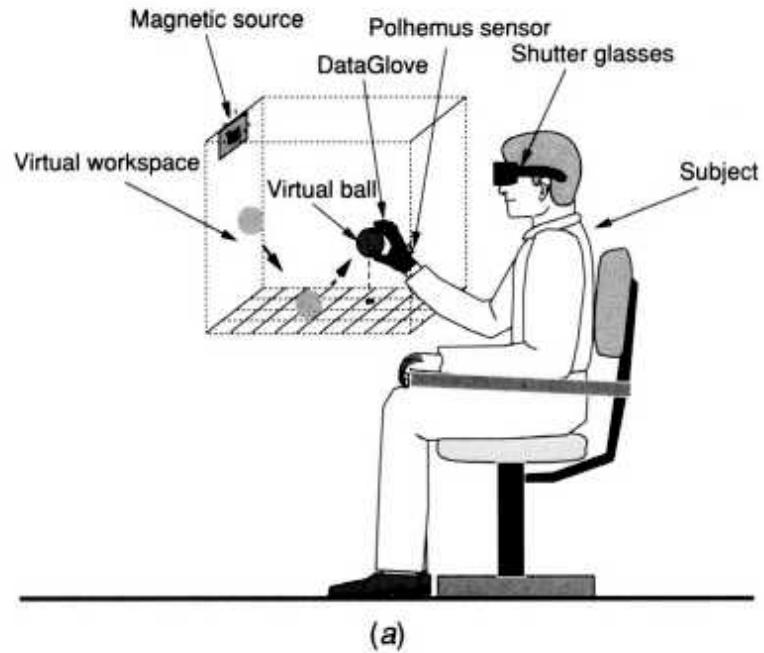
Ranadive [1979] studied the effect of frame refresh rate and image resolution on subject performance of a teleoperation task. Subjects used a seven-degree-of-freedom Argone E2 manipulator to perform remote tasks. The video feedback was assured by cameras placed at the remote site (no graphics rendering), and performance was measured as the subject's task completion time. It was observed that, as the video refresh dropped from 28 images/sec to 4 images/sec, the performance dropped also. The degradation

was considerable below 14 images/sec. Massimino and Sheridan [1989] confirmed this result for a block assembly task under direct and indirect (video) view. Subsequently, Piantanida and his colleagues [1993] discovered that the degradation was due to the subject confusing distances and the relative motion of various objects.

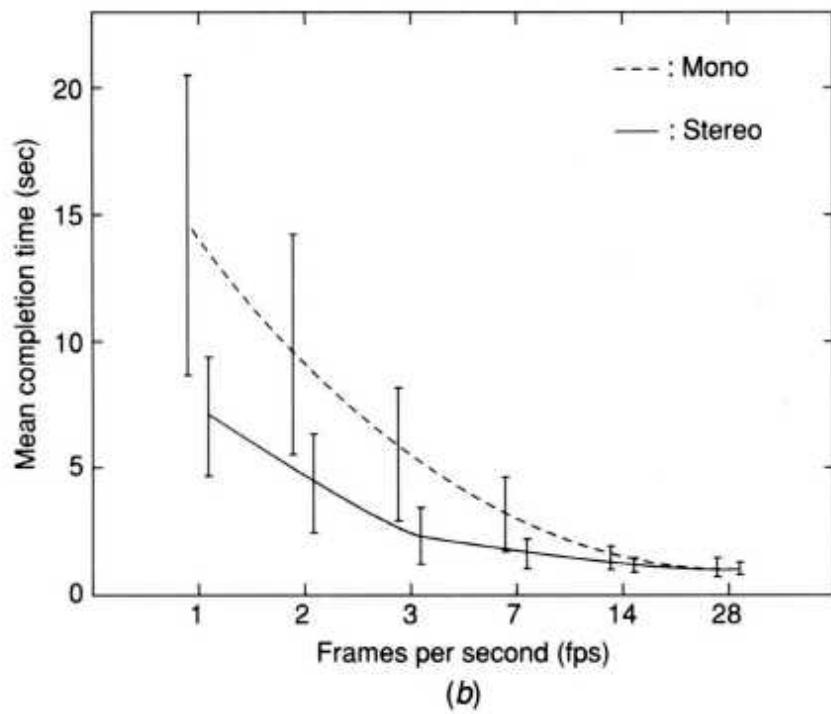
A type of task particularly sensitive to system responsiveness is the tracking and grasping (capturing) of a moving virtual object (target). Richard et al. [1996] studied the length of time taken by subjects to grasp a moving target in a 3D virtual world of fixed complexity as a function of frame refresh rate and graphics mode (monoscopic and stereoscopic). The scene consisted of a deformable virtual ball of 7 cm diameter (72 Gouraud-shaded polygons) bouncing inside a virtual room. As shown in Figure 7.8a, the walls were marked as meshes and prevented the ball from escaping outside this volume. The virtual ball had an X shadow projected on the floor to aid in depth perception. The target moved at 25 cm/sec with a random initial direction and bounced off walls without loss of energy. This dynamics model was necessary to make it more difficult for subjects to capture the ball. The scene included a virtual hand that moved in response to the operator, who wore a sensing glove and 3D tracker.

The experiments used a large number of subjects (42 male and 42 female) in order to obtain statistically significant data. The study used a between-subjects design, with subjects being assigned to look at either a monoscopic scene (on a CRT) or at a stereo scene (CRT and active glasses). For each graphics mode there were six groups of 7 subjects each. Groups had to capture the bouncing virtual ball when the frame refresh rate was set at 28, 14, 7, 3, 2, or 1 fps, respectively. Each subject performed 12 trials under one of these refresh rates. Trials were separated by 15-sec rest periods. Figure 7.8b shows subject performance in terms of task completion time. These results show that (for both monoscopic and stereoscopic graphics) there is little difference in performance when frame refresh rates were 14 frames/sec and above. However, once the frame refresh rate dropped to 7 fps or below, the performance was degraded and subjects took longer to capture the virtual ball. The effect of low refresh rates on capture time was less dramatic for the groups with the stereoscopic graphics condition. This

is certainly due to their ability to better perceive depth, regardless of frame refresh rate. Additionally, the standard deviation becomes larger and larger with the reduction in frame refresh rates, meaning that some subjects took much longer to capture the ball than their group average. There was less variability in the subjects who looked at a stereo scene, indicating better task performance uniformity among subjects in these groups.



(a)



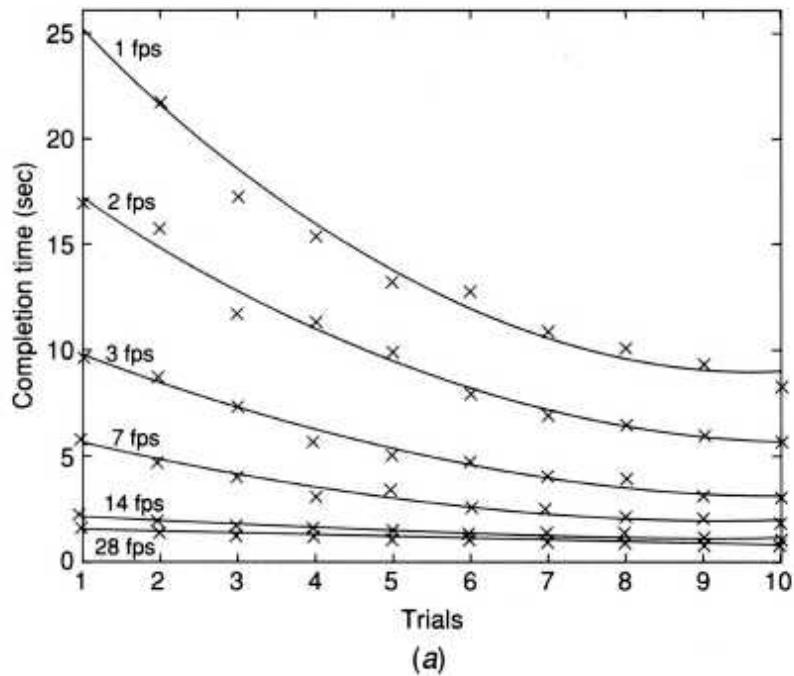
(b)

Fig. 7.8 Effect of frame refresh rate and graphics mode on the completion time of a bouncing ball capturing task: (a) experimental setup (shown here for the stereo condition); (b) experimental data for all subjects. From Richard et al. [1996]. © 1996 Massachusetts Institute of Technology. Reprinted by permission.

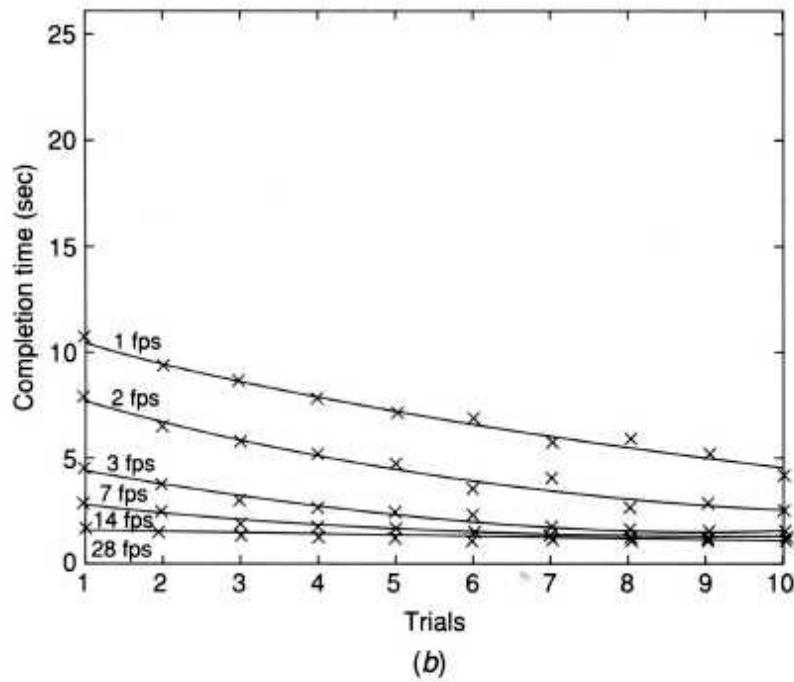
Figure 7.9 [Richard et al., 1996] shows the impact of frame refresh rate on another component of the tracking/capturing task performance, namely task learning time. It can be seen that subjects learned differently at 28 and 14 fps than they did at 1 fps. When the system responsiveness was poor and the graphics was monoscopic, subjects took 25 sec to capture the virtual ball when they first tried this. However, after 10 trials the time to capture the ball had dropped to only 8 sec. By contrast there was essentially no learning in the group that had 28-fps graphics (the learning curve is flat). Comparing the two graphs in Figure 7.9, we observe the impact of the graphics mode on task learning time. Here again stereo proved beneficial, as it reduced the amount of learning needed by all subjects. There was essentially no learning involved for frame refresh rates above 7 fps since stereo scenes looked more natural (and familiar) to the subjects. Finally, the frame refresh rate also influenced the strategy used by subjects to capture the virtual ball. Subjects in the groups with low frame refresh rates used prediction and tended to stalk the ball in a corner before capturing it. By contrast, subjects in the groups with high frame refresh rates tracked the ball continuously in a natural way.

Another type of task sensitive to the system responsiveness is the placement of a 3D object inside a target volume. Watson and his colleagues performed a series of studies aimed at determining the influence of system responsiveness and its variability (expressed as standard deviation of system responsiveness, SDSR) on object placement tasks. In the first study [Watson et al., 1998] subjects had to capture an object and subsequently place it on top of a target pedestal. They used a within-subject design in which participants received monoscopic graphics feedback at three levels of system responsiveness (frame refresh rates of 17, 25, and 33 fps, respectively). For each frame refresh rate, the variability in system responsiveness was set through percentage variation of frame refresh rate (5.6%, 22.2%, and 44.4%, respectively). Results showed that the subjects' performance (expressed as placement time and placement accuracy) was affected by both SR and SDSR. The variability of system responsiveness had a larger effect on placement tasks performed at low frame refresh rates. The worst performance corresponded to a mean frame refresh rate of 17 fps and a standard deviation in this rate of 44.4%. Under these conditions

subjects took on average 2.2 sec to place the grasped object in the target volume and had an accuracy of 86% in doing so. By comparison, the subjects completed the same task in only 1.72 sec when the frame refresh rate was 33 fps and had small variability (5.6%). Their placement accuracy improved to 90% under these conditions.



(a)



(b)

Fig. 7.9 Effect of frame refresh rate and graphics mode on task learning time: (a) groups seeing monoscopic graphics; (b) groups seeing stereoscopic graphics. From Richard et al. [1996]. © 1996 Massachusetts Institute of Technology. Reprinted by permission.

In the foregoing study the placement task difficulty (as expressed by the Fitts index [Fitts, 1954]) was kept constant with regard to the relative dimensions of the manipulated object and placement target area. In a subsequent study [Watson et al., 1999] the researchers looked at the dependence of task throughput on system responsiveness. Task throughput was expressed as the ratio of the index of difficulty and the time to place the grasped object in the target volume. The within-subject design changed the index of difficulty (from 1.5 to 6.5) and system responsiveness (from 215 to 345 msec). The study used 10 subjects (undergraduate students) with no prior VR experience. Results showed that it took longer to place the grasped object (a virtual ball) within the target volume with the increase in task index of difficulty (tighter tolerances). Figure 7.10a shows that, regardless of difficulty level, it took longer to place the object when the system was less responsive. However, the relative importance of system responsiveness (SR) grows with the task index on difficulty (ID). This is due to the higher demands placed by more difficult tasks on SR. Furthermore, the placement task throughput was also affected by system responsiveness, as seen in Figure 7.10b.

7.2.3 Influence of Feedback Multimodality

The studies described above kept constant the type of feedback provided to the user, but varied its quality (for example, by changing the frame refresh rate or the variability of this rate). We now look at the effect of the presence or absence of a given feedback type on task performance. We are interested especially in manipulation of virtual objects and assembly tasks under multimodal (graphics, sound, and haptic) feedback.

Boud and his colleagues at the University of Birmingham, United Kingdom, studied the influence of direct haptic feedback on the user's performance during an assembly task involving object manipulation and insertion [Boud et al., 2000]. The task chosen for this study was a "tower of

"Hanoi" assembly of three wooden disks. The stipulation that no larger disk be placed on top of a smaller one was satisfied by a unique sequence of intermediate steps, as illustrated in Figure 7.1 la. Therefore the task completion time was not influenced by the order of the intermediate steps, but only by the time it took to complete them.

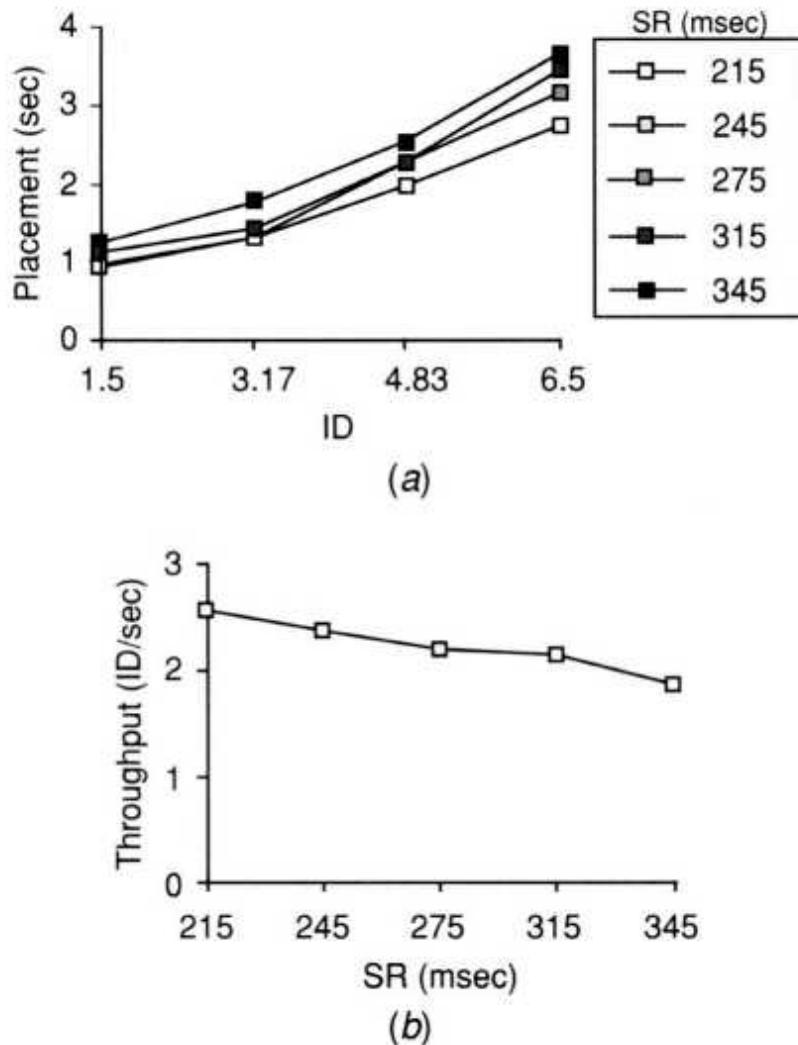
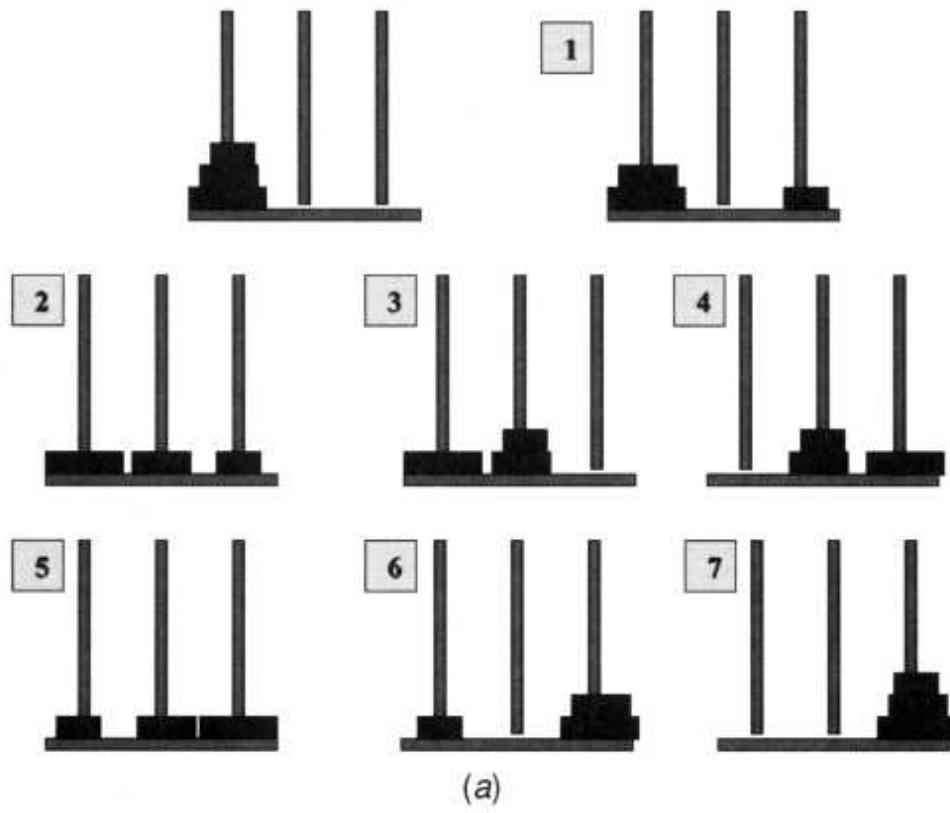


Fig. 7.10 Effect of system responsiveness (SR) and index of difficulty (ID) on an object placement task: (a) task completion time; (b) task throughput. From Watson et al. [1999]. © 1999 ACM Inc. Reprinted by permission.

The study used a within-subject design, such that participants had to perform the assembly under different feedback conditions. In the first condition subjects saw a virtual representation of the tower of Hanoi

displayed on an HMD and manipulated the 3D objects (disks) using a 3D mouse. In another condition they wore the HMD (showing a virtual representation of the task), but manipulated real wooden disks instrumented with 3D trackers (as shown in Fig. 7.11 b). Two other conditions involved the use of a monitor and 2D mouse (desk-top VR) or the manipulation of real objects under direct view. Each condition was presented randomly to four male subjects who had considerable prior VR experience. The subjects had to complete the tower of Hanoi assembly task 10 times for each condition. The subject's performance was measured as task completion time and its standard deviation. As expected, the shortest completion time corresponded to the performance of a real task (it took subjects on average 14 sec to do so). When no force feedback was present (the conditions that had 2D or 3D graphics feedback only) performance degraded significantly. Under these conditions the task completion time more than doubled (to 34 sec). However, when force feedback was provided (by using instrumented disks), the performance improved and subjects completed the task in 22 sec. The standard deviation was also smaller than for the previous two conditions.



(a)



(b)

Fig. 7.11 "Tower of Hanoi" assembly task: (a) task sequence; (b) subject manipulating instrumented objects. From Boud et al. [2000]. © Massachusetts Institute of Technology. Reprinted by permission.

Feedback multimodality involves the presence of a given feedback channel (such as force feedback in the foregoing example) as well as

sensorial substitution and sensorial redundancy.

Definition Sensorial substitution (or transposition) occurs whenever information that is usually in one sensorial domain is presented to the brain through another sensory system. Sensorial redundancy involves the use of several (at least two) sensorial domains to present the same information to the subject.

Sensorial substitution for visually impaired individuals is used when text is presented through haptics (Braille displays). Similarly, communication with the hearing impaired is done by replacing auditory feedback with visual feedback (sign language). An example of a task involving sensorial redundancy is pushing against a virtual wall. Force feedback is then provided directly through a haptic interface (such as the PHANToM) and visually by seeing the contact force rendered as a vector displayed in the graphics scene. In another example, if the simulation task is a peg-in-hole insertion, the user expects haptic feedback to his or her fingers. However, if the sensing glove does not have such features, then feedback of the contact forces may be visual or auditory. Table 7.1 summarizes the various sensorial transpositions that may be used in VR simulations, depending on the senses involved.

As seen from the table, we distinguish cases of simple (one-to-one) from complex transpositions (where several senses are involved) [Laurel, 1992]. In the first category fall the transpositions force-sound, force-image, and sound-image. The second category includes the transpositions force-(sound, image), force-(force, sound, image), sound-(force, image), and sound-(sound, force, image).

Sensorial transposition has, in fact, another interesting aspect, that of sensorial redundancy [Fukui and Shimojo, 1992]. In this case, information is provided in excess in order to help the subject to perform the task better. One such case would be force-(force, image) transposition. Here the subject receives force information, but also supplemental information about the magnitude of the force (either as a vector superimposed in the scene or a numerical value). This should allow more precise control in the simulation,

especially in complex tasks. At the same time one should be aware that too much redundancy is not desirable owing to the danger of sensorial overload.

Studies were done to determine the role of force feedback sensorial substitution and redundancy during manipulation of deformable virtual objects [Richard et al., 1996; Fabiani et al., 1996]. As illustrated in Figure 7.12a, subjects had to pick a virtual ball at location A and place it at location C by passing over location B. The task had to be accomplished in less than 15 sec. During manipulation the virtual ball had to be deformed not more than 10% of its radius, otherwise an error was recorded. The experimental system consisted of a partially immersive virtual environment, similar to the one shown in Figure 7.8a. The difference was that subjects now wore headphones and donned a Rutgers Master I (RM-I) force feedback glove on their right hand [Burdea et al., 1992]. This glove used off-the-shelf small pneumatic actuators to provide force feedback to an otherwise open-loop VPL DataGlove.

The study used a between-subjects design in which groups of subjects had to manipulate the virtual ball under different force feedback conditions. There were 84 subjects (42 male and 42 female) assigned to groups of 14 subjects each. All subjects were presented with graphics feedback. Group N received no force feedback (the glove actuators were not activated) and had to gauge deformation based solely on graphics information. The glove actuators were also deactivated for the subjects in Group V1, who were presented with force feedback sensorial substitution. This was realized through the LED visual display of the Rutgers Master I control box. The higher the modeled grasping force (thus the higher the surface deformation, assuming Hooke's law), the more LEDs were turned on. Group A received audio feedback through headphones, but no direct force feedback. This sensorial substitution technique changed the frequency of the sound in proportion to the modeled contact force. Two additional subject groups had to perform the task with sensorial redundancy. They were provided with contact force information either on the Rutgers Master I and LED displays (Group H-V) or through the Rutgers Master I and headphones (Group H-A). Figure 7.12b [Fabiani et al., 1996] shows the corresponding experimental results in terms of percentage ball deformation as the curve RM-I. It can be

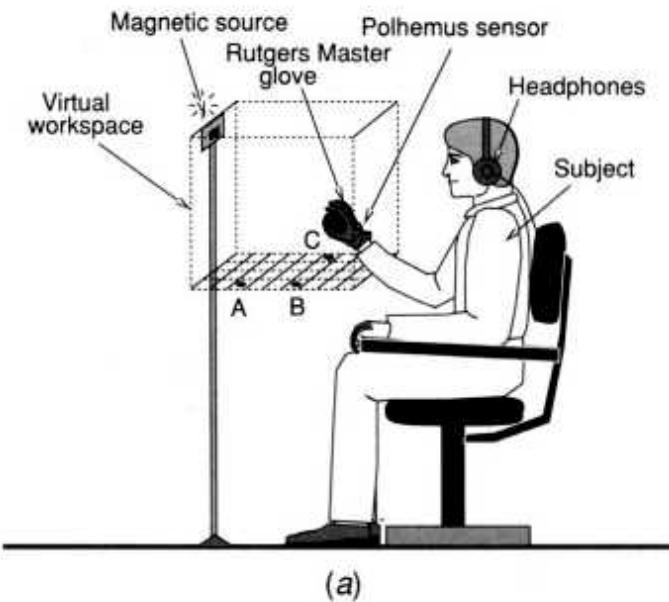
seen that among groups with nonredundant feedback modalities, the force feedback (Group H) assured the best subject performance (the smallest ball surface deformation). This was probably due to the richer force information provided by the Rutgers Master I than by any of the sensorial substitution conditions. The addition of redundant visual and especially auditory force feedback information improved performance further. Auditory feedback gave subjects a strong cue of initial contact (to supplement direct force feedback). The modeled contact forces were small in that instance and they were filtered by the static friction in the glove force feedback actuators. Thus performance suffered when no auditory cue was present. As a consequence, the amount of ball surface deformation was larger for Group H than for Group H-A.

TABLE 7.1. Human Sensorial Transposition Affecting the Simulation Loop

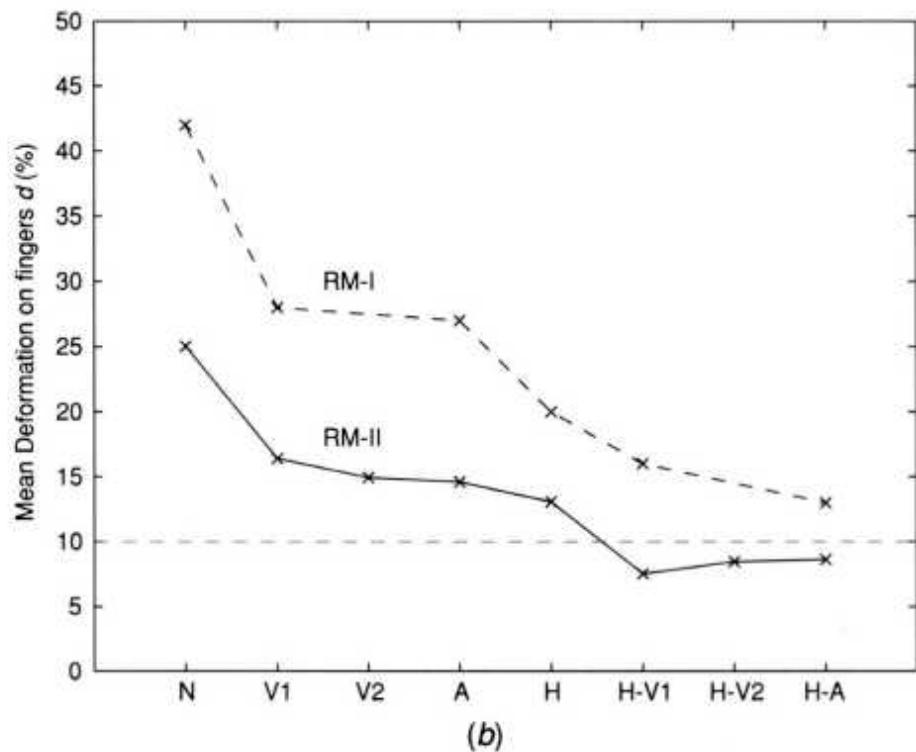
Input	Feedback	Can be Transposed To ^a
Auditory	Auditory	—
	Visual	Visual
	Haptic	Haptic
	Auditory/visual	(Visual)
	Auditory/haptic	(Haptic)
	Visual/haptic	Visual/haptic
	Auditory/visual/haptic	(Auditory)/(haptic)
Haptic	Auditory	Auditory
	Visual	Visual
	Haptic	—
	Auditory/visual	(Auditory)
	Auditory/haptic	Auditory/haptic
	Visual/haptic	(Haptic)
	Auditory/visual/haptic	(Auditory)/(haptic)
Visual	Visual	Auditory
		Haptic
		(Auditory)
		(Haptic)
		Auditory/haptic
		(Auditory)/(haptic)

^a Parentheses indicate a partial transposition.

Source: From Burdea and Coiffet [1993]. © Editions Hermès. Reprinted by permission.



(a)



(b)

Fig. 7.12 Effect of force feedback modality during manipulation of a deformable virtual ball: (a) experimental configuration; (b) ball deformation

for different feedback modalitiesgraphics only (N), visual (V 1, V2), auditory (A), H (haptic). From Fabiani et al. [1996]. © 1996 IEEE. Reprinted by permission.

Since friction proved a problem, the study was repeated 2 years later, this time using a redesigned haptic glove (the Rutgers Master II). This improved glove design incorporated custom glass-graphite actuators such that friction was reduced substantially. The ergonomics of the interface was also improved by integrating position sensors inside the actuators. This eliminated the need for a separate VPL DataGlove, while increasing the range of motion. The only other modification to the study design was the addition of the V2 condition, in which forces were visualized on the screen. This helped task performance since, unlike subjects in Group V 1, subjects in Group V2 did not have to look away from the screen to receive information on the magnitude of contact forces. New subjects were recruited and the virtual ball deformation measurements were repeated. As can be seen from the curve RM-II in Figure 7.12b, the performance improved for all subjects. Sensorial redundancy (for groups H-A, H-V1, and H-V2) resulted in ball surface deformations that were below the goal of 10% of its radius. Furthermore, the distribution of forces among fingers (not shown) became more uniform.

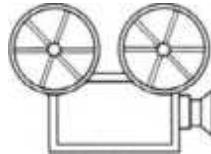
In this study we saw how vision was used as a redundant feedback modality to complement haptic feedback during manipulation of a compliant virtual object. There are cases where such cross-modal enhancement is used to supplement a weak haptic feedback or to create an illusion of haptic feedback when the interface does not provide it. An example is the study conducted by Biocca and colleagues in which subjects were asked to manipulate various virtual objects (anatomical organs or graphics primitives) using a pair of tracked Pinch Gloves [Biocca et al., 2001]. Subjects received visual feedback through tracked stereo HMDs and no sound feedback was provided. A visual cue depicting a virtual spring appeared whenever an object was to be manipulated, as shown in Figure 7.13 [Biocca et al., 2001]. The subjects had to stretch the virtual spring first before the object snapped into the palm. Immediately after the simulation, subjects had to complete a subjective evaluation questionnaire in which they

were asked whether they felt the weight or inertia of the grasped objects. Interestingly, of the 74 participants in the study, approximately 30% reported that they sometimes or always felt the weight of the objects they manipulated. As mentioned in Chapter 2, the Pinch Glove does not have haptic feedback at the wrist, which would be needed to simulate weight or inertia. Thus the dominant visual sensorial channel enhanced the proprioceptive and tactile feedback of the Pinch Glove (when fingers are touched) to create the illusion of weight.

An opposite type of cross-modal enhancement is sensorial conflict, in which information from one sensorial channel contradicts that received by another sensorial channel. Several effects can thus be created, ranging from illusions (moderate conflict) to simulation sickness (in case of severe conflict). Lecuyer and his colleagues conducted a study to determine the boundary of illusion created by contradictory visual and haptic cues [Lecuyer et al., 2001]. As illustrated in Figure 7.14a, the task consisted of compressing two pistons with virtual springs inside. One piston was used as reference, and its visual displacement was equal to that of a PHANToM interface used to provide force feedback to the subjects. Thus the modeled resistance of the reference virtual spring was that felt by the subject through the PHANToM handle. The other piston was a pseudo-haptic one, which had either the same stiffness as the reference virtual spring or up to 186% higher stiffness. Visually, however, its stiffness could be either higher (up to 40%) or lower (up to -70%) than the reference spring.

The study used a within-subject design in which each subject took repeated turns pressing the two virtual springs and deciding which was stiffer. Of interest here is the portion of the experiment in which the pseudo-haptic piston was haptically stiffer, but visually less stiff, than the reference spring. Thus the virtual spring would have a larger visual compression than the displacement of the PHANToM handle. Researchers found that the visual feedback compensated for the conflicting haptic feedback, such that the two springs were judged subjectively to be equally stiff. Figure 7.14b depicts the variation in the average point of subjective equality (over the subject population) as a function of the increase in the stiffness of the pseudo-haptic piston. As can be seen, a spring that felt visually 24% less stiff than the

reference spring was perceived as being equally stiff even when haptically it felt 86% stiffer than the reference virtual spring. Thus the graphics feedback was able to compensate for a much larger discrepancy in haptic information in order to make subjects perceive the two springs as haptically equivalent. Researchers state that there was great variability among subjects, some being more and some less prone to such sensorial illusions. This is consistent with Biocca's findings where only 30% of the subjects perceived the haptic illusion of weight. In a similar way there is great variability of responses to severe sensorial conflicts. These can induce cybersickness in some users, as discussed next.



VC 7.1

7.3 VR HEALTH AND SAFETY ISSUES

The numerous studies described in this chapter show how interaction techniques, system characteristics, system responsiveness, and multimodality have a direct effect on a subject's performance during VR simulations. The same factors play an important role in the effect simulations have on the user's health and safety. Viirre and Bush [2002] classify these as either direct effects or indirect effects.

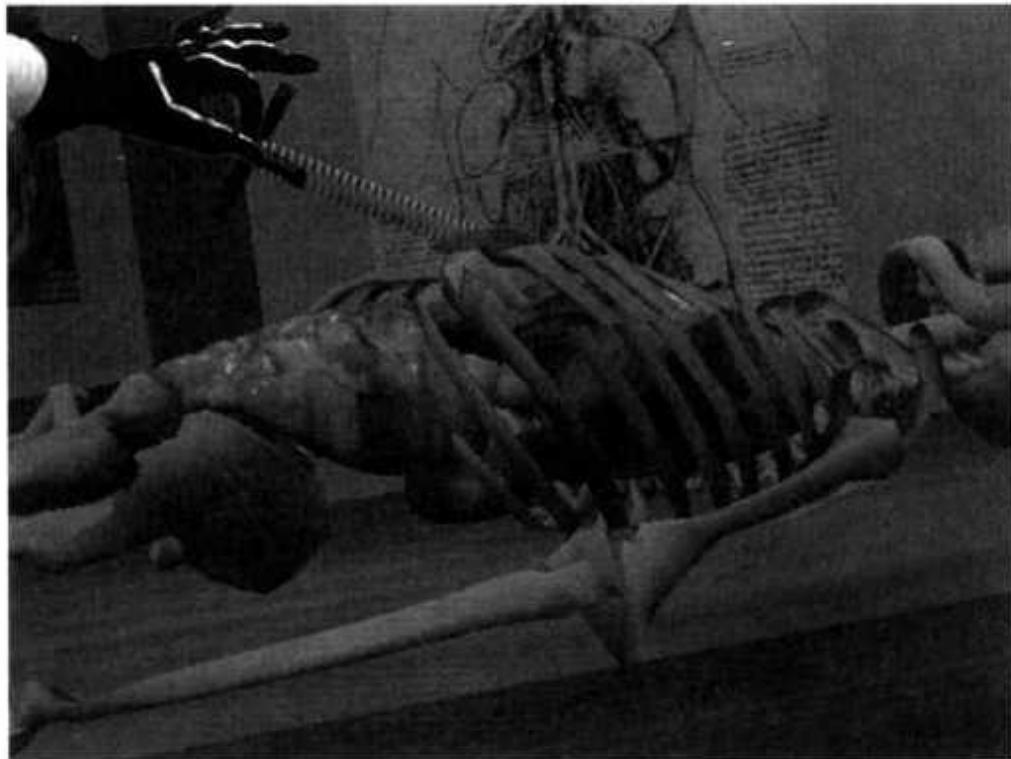
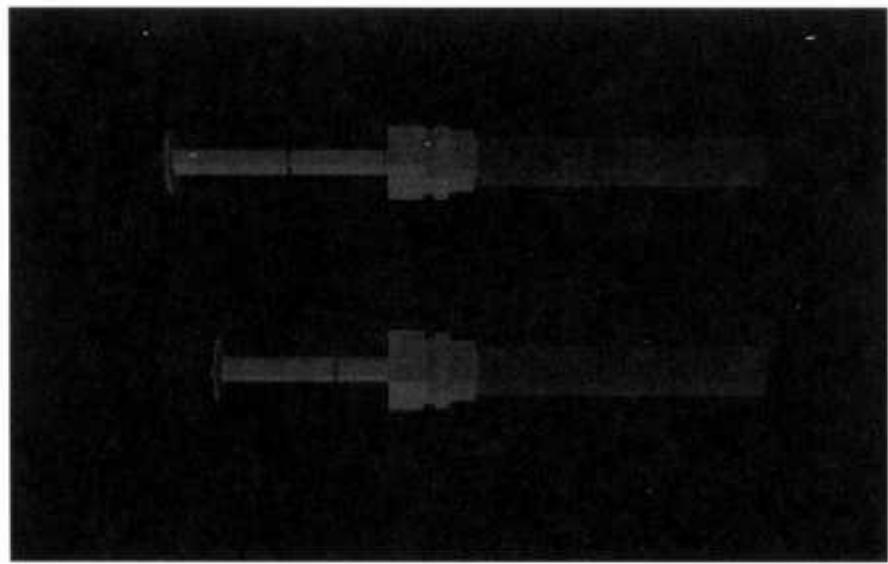


Fig. 7.13 Virtual spring used to created the illusion of weight. From Biocca et al. [2001]. © 2001 Massachusetts Institute of Technology. Reprinted by permission.

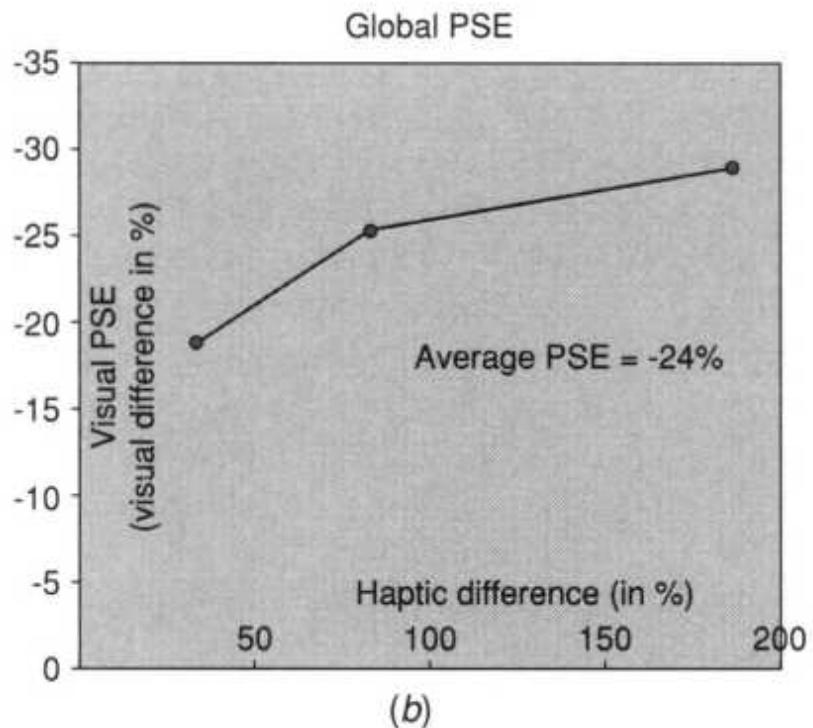
Definition Direct effects of VR simulations on the user involve energy transfer at the tissue level and are potentially hazardous. Indirect effects are neurological, psychological, sociological, or cybersickness and affect the user at a higher functional level.

7.3.1 Direct Effects of VR Simulations on Users

Direct effects of VR simulations affect mainly the user's visual system (as interactions are visually dominated), but also the user's auditory, skin, and musculoskeletal systems.



(a)



(b)

Fig. 7.14 Effect of sensorial conflict in producing a haptic illusion: (a) graphics scene depicting reference and pseudo-haptic virtual springs; (b) point of subjective equality (PSE) between the reference spring and a haptically stiffer and visually more compliant pseudo-haptic spring. From Lecuyer et al. [2001]. © 2001 IEEE. Reprinted by permission.

7.3.1.1 Direct Effects on the Visual System. These occur when a user is subjected to high-intensity light directed at his or her eyes. The combination of light intensity and duration of exposure exceeding tolerable limits will result in corneal burns, retinal burns, and other injuries. One example of potential damage is the laser used in miniature wearable displays, which directly illuminates the retina. Continuous viewing without hazard is only possible with Class 1 lasers of 400 nanowatts or less. By comparison, the simple laser pointer belongs to Class 3a and if directed at the retina, will cause damage [Kestenbaum, 2000].

Another type of light source directed to the eye is the infrared LED used in eye tracking systems. It measures the gaze direction of the user's eye by imaging the reflected IR signal using a miniature camera placed inside the HMD. High-output IR LEDs produce signals that can induce cataracts. This hazard is compounded by the fact that, unlike lasers, infrared light is invisible to the user. To avoid injury to the eye, the IR signal needs to be kept extremely low, 7.6×10^{-5} watts according to the U.S. Food and Drug Administration [2000].

The activity of the vision centers in the brain can also be directly affected by VR exposure. This typically happens by immersing the user in a scene where bright lights are pulsed at low frequency (1-10 Hz). This visual flow triggers repeated firings on the corresponding neural circuitry in a chain reaction. This in turn results in an induced "absence" state in which the user is nonresponsive to his or her surroundings. Such seizures, if repeated, can lead to brain injury. To avoid this it is necessary to control both the scene content and the frame refresh rate. Highly complex scenes containing bright virtual light sources and which are rendered at low frame refresh rates may become hazardous. Bright lights coupled with loud pulsing sounds and longer simulations may also result in migraines. Thus, users prone to migraines (20% of women and 10% of men) should avoid immersion in such environments [Viirre and Bush, 2002].

7.3.1.2 Direct Effects on the Auditory System. These affect the user if the simulation noise level is too high. According to the U.S. Occupational Safety and Health Administration, people should not hear sounds of 115 dB for

more than 15 minutes/day. If the intensity is reduced to 105 dB, exposure duration increases up to 1 hour/day. In VR simulations that have 3D sound sources, the perceived intensity depends on the user's position. If the user navigates too close to a very loud simulated sound source, then the recommended limits of exposure may be exceeded and damage to the auditory system will result. The same direct effects are present if the user is stationary and the sound source is navigating in the environment. This is the case in military training where the simulation involves virtual airplanes which periodically fly over the user (trainee).

7.3.1.3 Direct Effects on the Musculoskeletal System. These effects relate to the use of haptic interfaces which can apply high level of forces or push the user's limbs beyond anatomical range limits. These hazards are present in force feedback gloves that pull fingers backward or in motion platforms that move ankles beyond normal rotation angles. These hazards may result in tendonitis, muscle pain, and other orthopedic problems. Fractures from fall may also result due to tripping over cables and tethers so prevalent in today's VR systems.

Another potential impact is the increased risk of skin disease when VR interfaces (such as HMDs and sensing gloves) are shared between several users. This is typically the case in research laboratories or at industrial trade shows. To minimize the risk, vendors sometimes resort to disposable gloves worn by users under the actual device. An alternative is to redesign interfaces to avoid using porous materials, which can easily become host to bacteria and viruses.

7.3.2 Cybersickness

Perhaps the most troublesome effect of VR simulation on the user is the onset of cybersickness. The term is used here in a more restrictive way than motion sickness and simulation sickness. Motion sickness may be induced without immersion in a simulation by simply pivoting with the eyes closed or riding a roller-coaster in an amusement park. Simulation sickness is a byproduct of interacting with a simulation, which need not involve exposure to virtual environments [Lathan, 2001].

Definition Cybersickness is a form of motion sickness that results from interaction with or immersion in virtual environments. Its main symptoms are eye strain, disorientation, postural instability, sweating, pallor, drowsiness, nausea, and (in rare cases) vomiting.

The present state of VR technology and our limited understanding of human sensorimotor response explain the current high incidence of cybersickness among users. For example, 78% of users experience some form of oculomotor problems, 70% get nauseated, and 67% are disoriented following VR interactions [Stanney et al., 2002]. For some users these symptoms are so severe that they have to exit the simulation. These staggering statistics point to the need to understand the causes of cybersickness and consider probable aggravating factors (such as exposure intensity and duration). This knowledge will help us develop ways to reduce or eliminate the (negative) aftereffects of VR exposure.

7.3.2.1 Neural Conflict. This occurs when information from several sensorial channels is not in agreement, as in the case of purposely induced sensorial illusions. In other instances the sensorial conflict is due to VR's technological limitations and may trigger the onset of cybersickness, as illustrated in Figure 7.15.

The exact physiology and neural pathways involved in cybersickness are extremely complex and not fully understood [Harm, 2002]. However, it is widely believed that the conflicting multimodal information needs to involve the vestibular sensors in the inner ear canal. The two principal structures providing information on the user's head rotation and translation are the semicircular canals and the otolith organs, respectively [Stoffregen et al., 2002]. Feedback from these sensors is transmitted to the central nervous system, where it is compared with sensory data received from the visual and haptic/proprioceptive systems. Cybersickness is triggered when these multiple sensorial systems conflict.

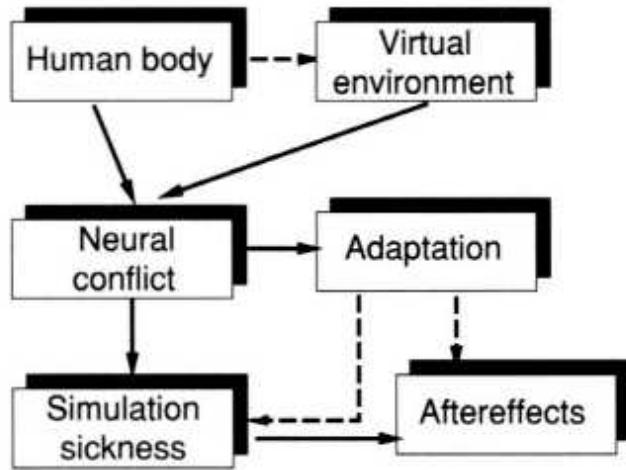


Fig. 7.15 A proposed model for cybersickness.

A universal VR task is navigation, as mentioned earlier in this chapter. Neural conflicts associated with VR navigation involve either conflicts between sensorial data from the vestibular and visual systems or the absence of such data. A flight simulator that does not have a motion platform will only provide visual feedback, but vestibular sensors will not be excited if the head is kept still. Conversely, a properly functioning motion platform may cause sensorial conflict when the simulation involves flying through fog, with the visual feedback remaining unchanged.

Stoffregen and his colleagues [2002] proposed a way to classify the motion dynamics of VR navigation, accounting for the user's head motion and the control of the vehicle. As illustrated in Table 7.2, the researchers distinguish between vehicle movements that are actively controlled and those that are passively experienced (as a passenger and not the driver/pilot). Of course head movement, whether rotation or translation, is always actively controlled. Each cell in the table has three possible visual-vestibular couplings, which are either equivalent, altered, or distinct from real-world experiences. An altered coupling is not equivalent to its real-world counterpart due to technical factors such as system latencies, optical distortions, or poorly calibrated trackers. A distinct coupling differs grossly from reality and involves the absence of either the vestibular or visual stimulation, as previously described. For example, a user of a domed flight

simulator with a fixed base (no motion platform) will have an equivalent head motion (there is no tracker and no mapping gain since the graphics are projected on a dome). However, the airplane's motion dynamics is distinct since there is no motion platform to induce inertial forces that would be sensed by the vestibular system. By contrast, consider a user of the same flight simulator, but this time providing graphics feedback through an HMD with an orientation-only tracker. Since the tracker does not measure translation, the simulation does not respond to head translations and there is a conflict between visual data and that provided by the vestibular system. Even the rotation motion has altered dynamics due to tracker latency and inaccuracies. The airplane's movement maintains a distinct dynamics as in the domed simulator case since there is no feedback of inertial forces (no motion platform).

7.3.2.2 Influence of System Characteristics. System characteristics such as tracking errors can affect the visual-vestibular coupling of users wearing an HMD. Such errors also affect the coupling between the motion of avatars shown in the scene and the user's proprioceptive system providing sensory information on limb motion.

TABLE 7.2. Proposed VR Motion Dynamics Classification System^a

	Head Movement	Vehicular Movement
Rotation	Active (equivalent, altered, distinct)	Active or passive (equivalent, altered, distinct)
Translation	Active (equivalent, altered, distinct)	Active or passive (equivalent, altered, distinct)

^aBased on Stoffregen et al. [2002]. Reprinted by permission.

Visual feedback characteristics are also important and potentially aggravating factors of cybersickness. Eye strain is induced by poor resolution or images presented through misaligned HMD optics. Large-FOV displays (such as display walls) can also induce a form of cybersickness calledvection. This is a compelling sensation of selfmotion experienced when sitting still in an auditorium and watching rapidly changing realistic scenes projected on a large display. This is in no doubt due to the dominance

of the visual sensorial channel (so-called visual capture) over the proprioceptive one.

There are other system characteristics that induce sensorial conflicts which can lead to cybersickness. One important factor is system lag due to latencies in tracking, communications, rendering, and scene display. System latencies above 100 msec degrade performance (as previously discussed in this chapter). Latencies are especially problematic for tracked HMDs, where the scene will appear as lagging (or floating) behind head motion. Large latencies cannot be compensated by the eye image-tracking mechanism (the so-called vestibular-ocular reflex), which controls visual stability. As a consequence, the user will overexert the eye muscles (resulting in eye strain) and may subsequently become nauseated.

7.3.2.3 The Influence of User Characteristics. The response to sensorial conflict differs from person to person. In other words, we are not all created equal when it comes to cybersickness. Some people exhibit cybersickness symptoms after very little time on a simulator, while other users of the same system are more "resistant," and exhibit fewer or no symptoms. Thus it is important to consider the role the user's characteristics play in making some people more prone than others to cybersickness.

It is widely believed that susceptibility to motion sickness (and thus to cybersickness) is age-dependent [Stanney et al., 2002]. Susceptibility is greatest between the ages of 2 and 12 years and declines thereafter, such that at age 25 we are half as susceptible as we were at age 18. Perhaps the reason for this age-induced resistance relates to the level of neurohormonal substances in the body, since some hormones improve the user's ability to adapt to visuovestibular sensorial conflicts [Harm, 2002]. Generally speaking, female users are approximately three times more susceptible to cybersickness than male users. Their susceptibility is also affected by such factors as pregnancy and the menstrual cycle, factors that are known to affect the level of hormones in the body.

Regardless of age or gender, users become more prone to cybersickness after taking medication that affects sensorial and neural-processing centers.

Alcohol and drugs are also known to affect the same neural mechanisms involved in cybersickness and thus to lower a user's resistance to it.

7.3.2.4 Influence of a User's Degree of Interactivity. This factor, expressed as a user's navigation techniques, also influences cybersickness. Recall that in Table 7.2 we distinguished between passively experienced navigation and navigation under the user's active control. When users interact with the simulator through an interface, they exercise their muscles and move their limbs, and thus the proprioceptive/haptic system provides additional sensorial data to the central nervous system. Such data play a positive role in reducing the effects of cybersickness through accelerated adaptation (as will be discussed later).

Stanney and Hash [1998] performed a study to determine the influence of the user's degree of interactivity on the onset and severity of cybersickness. The task involved navigation through a virtual maze that required passage through windows or doors or the use of elevators. Twenty-nine subjects (14 male and 15 female college students) wearing active glasses navigated through the maze, which was displayed in stereo on a CRT. The study used a between-subjects design, such that the subjects were assigned to either an active control, a passive control, or an active-passive control experimental group. Subjects in the active control group navigated through the maze using a six-DOF joystick. By contrast, subjects in the passive-control group could not initiate any input to the viewpoint change and were taken on a ride. The subjects in the third experimental group had help in navigating obstacles by adaptively removing unnecessary degrees of freedom from their input. For example, during passage of a doorway the }, translation as well as roll and pitch rotations were disabled. Each subject was immersed in the simulation for about 30 minutes and immediately thereafter had to fill out a subjective evaluation questionnaire. This questionnaire, called the simulation sickness questionnaire (SSQ) [Kennedy et al., 1993], scored the severity of a number of cybersickness symptoms (as none, slight, moderate, or severe). The individual scores were then weighted in order to obtain a total severity score (TSS). As can be seen from Figure 7.16 [Stanney and Hash, 1998], the highest severity of cybersickness was experienced by subjects in the passive-control group (gray). The subjects in the active-passive navigation

control group were the least affected by cybersickness. Stanney and Hash believed this was due to the reduction in navigation degrees of freedom when passing obstacles, which in turn reduced the amount of unnecessary motion. Since less motion meant less sensorial conflict, the severity of reported nausea, oculomotor distortion, and disorientation was smallest for this experimental group.

7.3.2.5 Temporal Aspects of Cybersickness. These include the amount of time the subject is immersed in the VR simulation as well as the distribution of repeated exposures over time. Kennedy and his colleagues [2000] estimate that such temporal aspects have a significant (20-50%) impact on the outcome of cybersickness.

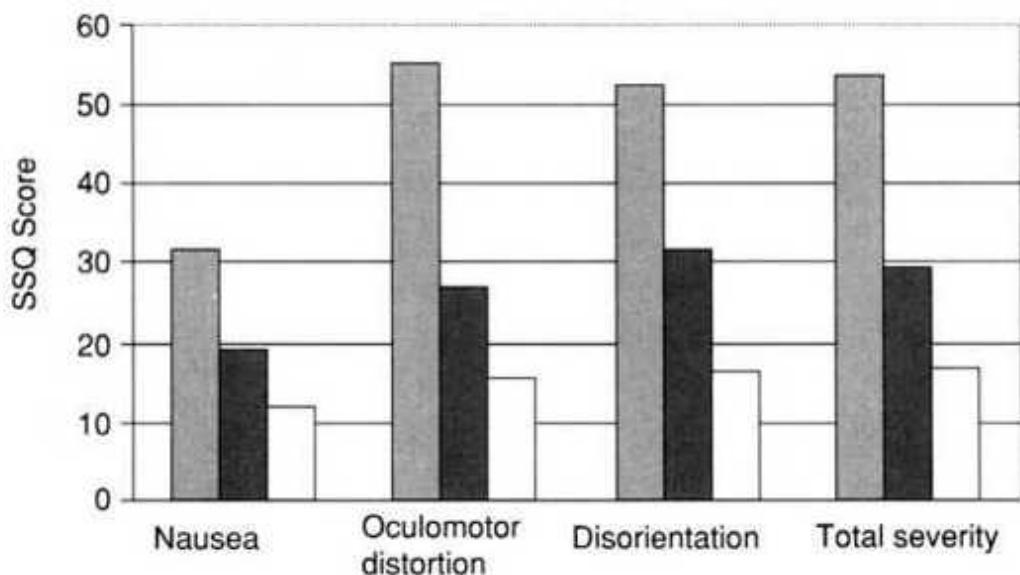
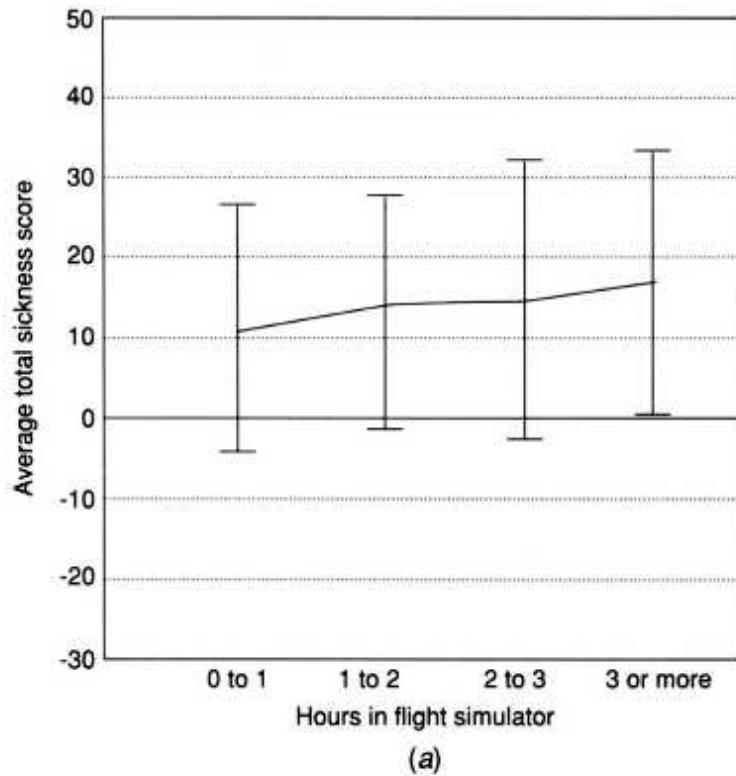
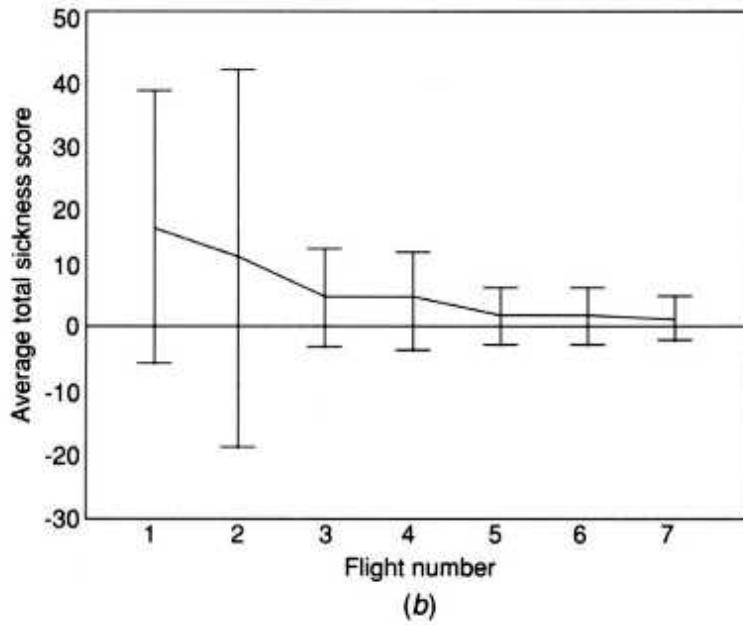


Fig. 7.16 Influence of user degree of interactivity on cybersickness severity: gray, passive control group, dark, active control group, white, active-passive control group. From Stanney and Hash [1998]. © 1998 Massachusetts Institute of Technology. Reprinted by permission.



(a)



(b)

Fig. 7.17 Severity of cybersickness as a function of: (a) VR exposure duration; (b) repeated VR exposures over time. From Kennedy et al. [2000]. © 2000 Massachusetts Institute of Technology. Reprinted by permission.

It is believed that a user immersed for the first time in a VR simulation will have symptoms of cybersickness that depend on the magnitude of the

sensorial conflict. For a given sensorial conflict magnitude and a given subject, the severity of cybersickness grows in proportion to the time spent in the simulator. Figure 7.17a shows the total severity score (reported on SSQ questionnaires) as a function of hours spent by pilots in helicopter flight simulators [Kennedy et al., 2000]. These data were collected from hundreds of subjects and achieve statistical significance. It can be seen that the cybersickness severity was 70% higher for sessions lasting 3 hours or more versus those lasting at most 1 hour (TSS mean of 17.48 vs. 10.78 for the shorter sessions).

Figure 7.17b [Kennedy et al. 2000] graphs the variation in TSS data for repeated sessions on a helicopter flight simulator. The data were compiled from 53 series of questionnaires filled out over seven consecutive simulation sessions. There was a clear and dramatic reduction in average severity scores between the first flight session (TSS of 15.53) and the last one (TSS of 1.55). Furthermore, the standard deviation among the scores reported by individual pilots exhibited a similar significant reduction (from 22.03 to 4.11). Thus, not only did the pilots experience significantly less cybersickness following repeated simulation sessions, but they did so uniformly as a group. It is important to note that the zero point for this graph corresponds to an elite group of subjects (military pilots). Thus their perceived acceptance level or tendency to underreport cybersickness severity may not generalize to the vast majority of (civilian) VR users.

7.3.3 Adaptation and After effects

In the second study conducted by Kennedy and colleagues [2000], the helicopter flight simulator system as well as the training scenario remained unchanged over repeated sessions. Nor were subjects replaced over these seven training sessions. Thus the system and user characteristics as well as the degree of interactivity remained the same throughout. What then could be the cause of the drastic reduction in cybersickness total severity scores plotted in Figure 7.17b?

During interaction with virtual environments a user is subject to sensorial conflict or rearrangement since the relationship between several sensorial

systems differs from normal [Welch, 2002]. The brain response to this rearrangement is adaptation, which is defined by Welch [1978] as follows:

Definition Adaptation to sensory rearrangement is a semi-permanent change of perception and/or perceptual-motor coordination that serves to reduce or eliminate a registered discrepancy between, or within, sensory modalities, or the errors in behavior induced by this discrepancy.

In other words, over repeated VR exposures, the brain learns to consider abnormal multimodal sensorial relationships as normal, within context. Therefore sensorial conflicts are no longer perceived as such by the brain, and the severity of cybersickness is reduced. Over time the user develops a compensatory response to the conflicting sensorial information and is said to be adapted to the simulation. Adaptation is facilitated by active interaction, error-corrective and immediate feedback, incremental VR exposure to the simulation (and thus to the sensory rearrangement it generates), and the use of distributed practice [Welch, 2002].

The basic premise for adaptation is that the sensory rearrangement is unchanged over time. An example would be a system with constant latency or a visual feedback with a constant shift or magnification. If, however, system latencies, visual shifts, and the presence or absence of a given sensorial modality all vary asynchronously during the simulation, then the brain is subject to sensory disarrangement. Under these conditions it is more difficult or impossible to adapt over repeated exposures to VR. Adaptation does not occur because the brain cannot develop a compensatory rule that remains constant over time.

Adaptation to VR occurs whether cybersickness is present or not. An instance where adaptation is not coupled with cybersickness is the remapping of the internal proprioceptive system during a manipulation task where the visual feedback is distorted. Groen and Werkhoven [1998] performed an experiment to measure the effect of purposely induced lateral shifts in tracking data on hand-eye coordination. Prior to VR immersion, subjects were asked to reach for a visual target (a die) while their whole arm was hidden from view. This was realized by holding a tracker receiver between the fingers of the right hand while moving the arm under a half-

mirror on which the visual target was projected. As illustrated in Figure 7.18a, the subjects' targeting was accurate since their proprioceptive system calibration was correct. In other words, the subjects correctly estimated where their arm would be, based solely on their proprioceptive sensing. Subsequently, subjects donned a low-resolution tracked HMD and a tracked VPL DataGlove. This time they saw a scene depicting a die and a virtual hand. The subjects were instructed to again reach for the target, in the presence of a 10-cm lateral shift in the glove tracking data (Fig. 7.18b). Due to this grossly incorrect tracking, subjects saw the virtual hand overshoot the target, while their (unseen) real hand was correctly mapped on it. After repeated trials the subjects adapted, such that their virtual hand now reached the target correctly. Their real hand position was, however, offset to compensate for the translation shift in tracker data (Fig. 7.18c). This was done unconsciously due to the dominance of vision over the arm's open-loop proprioceptive control. At the end of their VR immersion the subjects were asked to keep their eyes closed and place their arm under the half-mirror apparatus. Then they opened their eyes and had to reach for a real visual target, similarly to the test performed prior to VR immersion. Again the subjects estimated the hand location based only on their proprioceptive sensorial channel. Interestingly, the subjects stopped their motion prematurely, at about 2 cm from target, in the opposite direction of the tracking shift induced during VR immersion (Fig. 7.18d).

This study shows not only that there was adaptation in the absence of cybersickness, but, just as importantly, that the effects of sensorial adaptation lasted once the VR immersion ended. These aftereffects are not permanent and generally subside over time. Once the adapted subjects were able to see their arm touch an object, the aftereffect disappeared. It is as if the visual system was again correctly calibrated to the proprioceptive one.

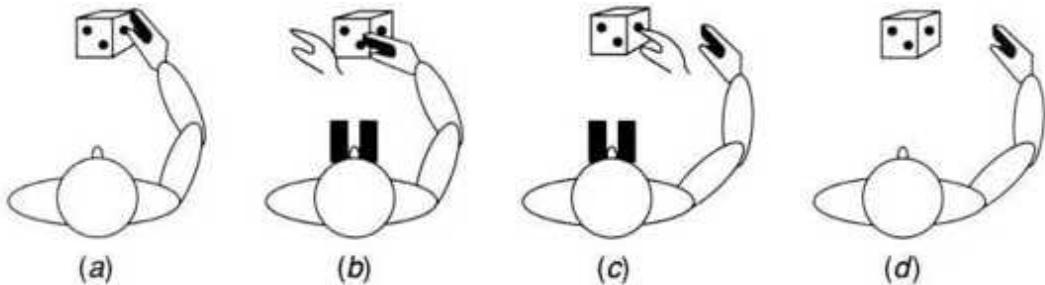


Fig. 7.18 Hand-eye coordination adaptation: (a) before VR exposure; (b) initial mapping through artificial offset; (c) adapted grasping; (d) aftereffects. From Groen and Werkhoven [1998]. © 1998 Massachusetts Institute of Technology. Reprinted by permission.

DiZio and Lackner [2002] reviewed proprioceptive adaptation and its aftereffects. They concluded that an adapted user will experience aftereffects upon return to the real environment in the form of deviated body movements, erroneous estimation of external forces on the body, and even auditory mislocalization. However, aftereffects are by no means only linked to proprioceptive mapping, since visual [Wann and MonWilliams, 2002] and vestibular aftereffects [Stoffregen et al., 2002] have also been observed.

An important class of aftereffects includes those associated with cybersickness, which manifest themselves as flashbacks, sensation of self-motion, head spinning, headaches, and diminished hand-eye coordination. These aftereffects last hours, even days, following the end of VR exposure and affect the user's performance on real tasks and even his or her safety. As a result, the military has grounding policies following sessions on flight simulators. Such policies ban driving, biking, roof repair, or operation of certain machinery for many hours after a simulator session ends. This gives the pilot trainee time to readapt to the sensorially correct surroundings without taking unnecessary risks.

An alternative approach to waiting a certain amount of time after VR exposure was proposed by Welch [2002]. He considered the case of training simulators for oil tankers, which introduce large time delays between the user's actions and the corresponding change in visual feedback. This time delay, while correctly modeling the large inertia exhibited by oil tankers, may become problematic due to aftereffects. User adaptation could pose safety risks if the user attempts to drive immediately following the end of such a training simulation. Welch proposed to replace the customary waiting time with another session on the VR trainer done immediately after the regular session. During this readaptation session the simulation would be changed by eliminating the time delays, such that the dynamics will resemble those of the car the user will subsequently drive. This relearning approach, which uses the simulator itself to eliminate the aftereffects, could

presumably be extended to other types of simulation. Of course, this calls for further human factors research, but the idea seems intriguing.

7.3.4 Guidelines for Proper VR Usage

We conclude our discussion on the effects of VR exposure by compiling a set of guidelines meant to minimize the onset and severity of cybersickness. These guidelines, outlined in Table 7.3, are largely qualitative owing to the variability in hardware performance, application domains, and interaction techniques encountered in today's VR systems. The guidelines are based largely on the usage protocols developed at the University of Central Florida [Stanney et al., 2002], on the results of the human factors studies previously presented, and on common sense. They are neither all-encompassing nor static. Indeed, as VR technology and interaction techniques improve, some of the guidelines presented here will become obsolete. We may very well reach a point where all that has to be controlled is immersion time, primarily to prevent VR overdose. This raises higher level, primarily sociological, issues of VR, which are discussed next.

TABLE 7.3. VR Design and Usage Guidelines Aimed at Minimizing Cybersickness^a

During system development
Minimize latencies and make them stable
Avoid pulsating light sources of low frequency
Reduce spatial frequency content in large displays
Ensure that HMDs have properly aligned optics and sufficient resolution
Reduce intensity and duration of loud 3D sound sources
Use accurate trackers and remove sources of interference
Assure consistency in multimodal displays
Before immersion
Screen users whenever possible for susceptibility to cybersickness
Place warning labels and educate users about potential adverse effects from VR exposure
Limit exposure to users who are free from drugs and alcohol consumption
Encourage users to be well rested before exposure
Discourage VR usage by those with cold, flu, binocular anomalies, or susceptibility to migraines or photic seizures
During immersion
Provide proper airflow and comfortable air temperature (preferably below 70°F)
Ensure equipment fits users comfortably through necessary adjustments
Minimize initial exposure time for strong stimuli (10 minutes or less)
Monitor users for signs of cybersickness
Inform users they can/should discontinue the simulation if they so wish
After immersion
Measure user hand–eye coordination and postural stability
Introduce a time period immediately after VR exposure in which users are not allowed to perform high-risk activities (driving, piloting, biking, etc.)
Possibly reimmerse users in a readaptation simulation
If necessary, follow up with users to monitor prolonged aftereffects
Introduce intersession periods of 3–5 days.

^a Adapted in part from Stanney et al. [2002]. Reprinted by permission.

7.4 VR AND SOCIETY

The authors' background is neither sociology nor psychology. Still, we feel it is useful to include aspects of general interest related to the societal impact of large-scale use of VR. There is already abundant (though unstructured) literature on the subject. Authors concentrate either on Internet usage issues, science-fiction (with little regard to present technological limitations), or the sex industry [Lowe, 1992; Epley, 1993; Faxon, 1993]. We think that even

the virtual world should be treated with realism by considering certain eternal human characteristics in relation to particular features of this technology.

Mankind's understanding of the importance of technical progress has made it a continuous, ever-accelerating process (especially in the era of global communication and economic integration). If a product is technically feasible and meets a certain consumer need, then it will proliferate inexorably. It happened with computers, with the Internet, and it is safe to say that the same will apply to VR systems. What is difficult to predict is the time it will take, whether it will be 5, 10 or 30 years. Let us remember that in 1954 the creators of "artificial intelligence" were convinced that in 10 years they would have a computer that "thinks." Unfortunately, 50 years later they are far from their target.

Another aspect of technological progress is that in itself it is neutral, that is, it is up to us to apply the discoveries for mankind's benefit or its detriment. Examples are abundant, but the dual use of atomic energy seems to be very persuasive. This is yet another motivation for analyzing our responsibilities with respect to the new VR technology. The following chapters give a number of examples of VR applications (present and emerging). The sociological impact of these applications will be felt in three areas: professional life, private life, and public life.

7.4.1 Impact on Professional Life

In almost any field, the major activities of a profession consist in learning, design, analysis, realization, and communication. VR may benefit all these activities and increase overall individual productivity. At the same time, its networking capability (through distributed simulations) will allow teamwork and easy expert consultation, thus improving the quality of the end product.

Much is said these days of the work-at-home alternative to the long commuting, traffic jams, pollution, and stress we face daily. Here the Internet and the networking ability of VR will allow more people to perform complex work while remaining at home. This last aspect will have a

positive impact on family life, especially with respect to the problems that professionals face in raising their children.

7.4.2 Impact On Private Life

While the positive impact on professional life is clear, the exaggerated use of VR in private life may accentuate some already existing societal problems. In principle, VR is a formidable communication modality, but it can also lead to unprecedented individual isolation. The present abuse of television and video games, so widespread in the young and not so young, could be further exacerbated by the new technology. One can spend a large part of one's time in the pleasure and easiness of a simulation and thus evade the realities and effort necessary for education.

Since the technology is new, there is not enough statistical evidence to sustain the isolationist hypothesis, nor is there enough evidence to sustain the idea that the Internet and VR will foster stronger societal interaction. In fact studies [Franzen, 2000; Calvert, 2002] have found conflicting results. For example, the argument that online shopping and entertainment will diminish face-to-face contact and interaction is balanced by the fact that this will save travel time to stores, time that can be spent with family and friends. Franzen's results on Internet usage in Switzerland show that there was no adverse effect on social interaction, whereas time spent on TV viewing decreased.

VR could be a positive substitute for drugs in pathological cases, and is already used clinically to alleviate symptoms of phobia, anxiety, and other psychiatric disorders [Hodges et al., 2001], as discussed in Chapter 8. Conversely, VR could also create a whole new class of depersonalized addicts to this new "drug." Calvert [2002] talked of addicts to "Multi-user domains, dungeons and dragons" (MUDs), persons who spend as much as 40 hours every week playing this online, text-based game. Of course, games, entertainment, and relaxation are a necessary part of adult life, and even more so for children. The new VR games could be very beneficial if enjoyed over normal and not disproportionate amounts of time. It is thus a question of "dosage." This is mainly left to the individual (or the family), as it is hard to imagine another way to control its use.

One aspect which has given rise to a great amount of attention, ink, and controversy is the use of VR as a companion to or substitute for (real) sex. The literature has coined the term "virtual sex" (here mainly regarding the male experience, which is an oversimplification [Epley, 1993]). It is fair to say that sex is an important (and necessary) part of life. Human nature is such that the sex industry (legal or illegal) has always made a great deal of money through prostitution, direct or indirect. In doing so the industry has used the most effective communication media (the Internet, TV, magazines, video, etc.). It is thus expected that the sex industry will invest heavily in this new medium, especially in view of its multisensorial nature. Here a number of questions are raised. "Will VR prevent the spread of AIDS"? "Is it immoral if a married man has an affair with a virtual rather than a real mistress?" [Faxon, 1993]. "Is it immoral if VR users deceive other users by assuming a persona of opposite sex, then engaging in virtual encounters of an intimate nature?" [Calvert, 2002].

These questions seem premature simply because the underlying technology is far from allowing us to build a "virtual" man or woman that has all the physical, sensorial, and psychological properties of a real person. The models are too complex for today's computers, the feedback miniature actuators do not exist, etc. Thus we do not see an impact on the sex industry (in either the numerical or pathological sense) in the immediate future.

7.4.3 Impact On Public Life

We have already insisted on the unmatched communication medium that VR represents. Communication is by definition information transfer, and information is at the foundation of our social life. Thus the effects of VR on public life may be significant.

First, VR may have a beneficial effect on governing bodies by making administrators more efficient and bringing them closer to their constituents. Time delays may be shortened, personnel files made more accurate, communication improved, etc. At the same time, there are risks of depersonalized judicial decisions and invasion of privacy. In the worst case VR could allow governments to become the feared Big Brother, and thus very strict information access laws need to be enacted. VR may also become

a perfect disinformation tool in the hands of governments or large economic interests [N. Adams, Adams Consulting Group, Western Springs, IL, personal communication, 1993].

An interesting phenomenon that emerged at the end of the 20th century is the appearance of online societies, the most popular being the Alphaworld [Schroeder, 1997]. This online society is populated by hundreds of thousands of users, some being "tourists," some long-term "inhabitants." As shown in Figure 7.19, users chose an avatar to represent them and communicate through text (displayed at the bottom of the window). Users move around (by mouse navigation) and can build objects that are added to the world. Thus the virtual world is not static, rather it is evolving, and becomes a fertile place for social interaction (including pro- and antisocial behaviors). After spending 10 hours in Alphaworld, Schroeder [1997] observed that the degree of engagement with other users and with the virtual world is higher than that experienced in text-based MUDs or in 2D worlds with limited interaction.



Fig. 7.19 View from Alphaworld. From Schroeder [1997]. Reprinted by permission.

We are convinced that present VR developments have potential consequences for public, private, and professional life. The question is whether the change will be incremental or revolutionary. We do not know the answer for at least two reasons. First, the technology and market are in an initial stage of development. Second, it is highly improbable that one can foresee all possible technological barriers that may bring promising applications to a dead end. Finally, we hope that mankind will know how to seize the chance of virtual reality without becoming its victim.

7.5 CONCLUSION

This chapter reviewed the large spectrum of human factors research done to measure user performance during VR interactions as well as the effects such interactions have on the individual and on society in general. We showed how system characteristics, interaction techniques, and multimodality all play a role in task performance. These same factors together with the user's characteristics and simulation dosage issues play important roles in the effects of VR exposure on the user. We gave qualitative guidelines for proper use of VR systems in order to reduce the effects of cybersickness. The awareness gained here should better prepare the reader for the study of current and emerging VR applications, which are the subject of the remainder of the book.

7.6 REVIEW QUESTIONS

1. What are human factors studies, why are they important to VR, and what are their main stages?
2. What variables are measured in a typical human factors study, and why? What is the meaning of standard deviation?
3. In the context of a human factors study, what is a subject? What is the difference between experimental and control groups? What is the difference between a case study and a control study? What is an institutional review board (IRB)? What are the variables set by an experimental protocol?

4. What is the meaning of a small standard deviation versus a large one for a given trial? What is the meaning of a decreasing average in a task completion time during repeated trials versus a constant average over repeated trials? Make a drawing and explain.
5. What are the variables of interest for human factors tests of tasks with force feedback? Why?
6. What differences exist between human factors engineering and usability studies? What methodology is followed by usability evaluations?
7. What universal VE tasks were studied on testbeds and how?
8. What is the effect of graphics view mode (stereo/mono) at low frame rates on a tracking/capture task?
9. What is the influence of frame refresh rate on standard deviation for a subject population? What is the influence of graphics mode (stereo/mono) on task completion time?
10. What is the influence of system responsiveness on a user's performance? What about the magnitude of SDSR?
11. What are sensorial substitution and sensorial redundancy?
12. What were the results of RM-I/RM-II ball deformation tasks? Explain the effect of friction on the results.
13. What did the Rutgers 1996 study show about the influence of graphics refresh rate on task completion rate? What about the effect of graphics mode (stereo vs. mono)? Make a drawing and explain.
14. What was the task chosen in the U.K. study of the influence of force feedback mode on task completion time and why? What were the four conditions chosen for the experimental groups? Which was fastest and why?

15. In the subsequent Rutgers study on sensorial redundancy, transposition and influence of glove dynamic rate, what modality (or modalities) ensured that the plastically deformed ball had less than 10% deformation during manipulation? Make a drawing and comment?
16. What is sensorial illusion and what boundary was uncovered by the study of virtual springs?
17. What direct effects can VE immersion have on users?
18. Describe cybersickness and its causes. Make a drawing and explain.
19. What is the relation between system characteristics and cybersickness?
20. Do the user's characteristics (age, health, gender, etc.) play a role in the onset of cybersickness? Why yes, or why no?
21. What is the influence of the degree of user control on the severity of cybersickness?
22. What is adaptation? Explain the findings of the proprioception adaptation study.
23. What are aftereffects? How can a user readapt following VR exposure?
24. What are the social implications of VR?

REFERENCES

- Biocca, F., J. Kim, and Y. Choi, 2001, "Visual Touch in Virtual Environments: An Exploratory Study of Presence, Multimodal Interfaces, and Cross-Modal Sensory Illusions," *Presence*, Vol. 10(3), pp. 247-265.
- Boud, A., C. Baber, and S. Steiner, 2000, "Virtual Reality: A Tool for Assembly?" *Presence*, Vol. 9(5), pp. 486-496.
- Bowman, D., D. Johnson, and L. Hodges, 2001, "Testbed Evaluation of Virtual Environment Interaction Techniques," *Presence*, Vol. 10(1), pp. 75-

- Burdea, G., and P. Coiffet, 1993, *La Réalité Virtuelle*, Hermès, Paris.
- Burdea, G., J. Zhuang, E. Roskos, D. Silver, and N. Langrana, 1992, "A Portable Dextrous Master with Force Feedback," *Presence*, Vol. 1(1), pp. 18-28.
- Calvert, S., 2002, "The Social Impact of Virtual Environment Technology," in K. Stanney (Ed.), *Handbook of Virtual Environments: Design, Implementation and Applications*, Erlbaum, Mahwah, NJ, pp. 663-680.
- Das, H., H. Zak, W. Kim, A. Bejczy, and P. Schenker, 1992, "Operator Performance with Alternative Manual Control Modes in Teleoperation," *Presence*, Vol. 1(2), pp. 201-218.
- DiZio, P., and J. Lackner, 2002, "Proprioceptive Adaptation and Aftereffects," in K. Stanney (Ed.), *Handbook of Virtual Environments: Design, Implementation and Applications*, Erlbaum, Mahwah, NJ, pp. 751-771.
- Epley, S., 1993, "Sex in Virtual Reality," *CyberEdge Journal*, 1993 (May/June), pp. 11-13.
- Fabiani, L., G. Burdea, N. Langrana, and D. Gomez, 1996, "Human Performance Using the Rutgers Master II Force Feedback Interface," in *Proceedings of the IEEE International Symposium on Virtual Reality and Applications (VRAIS '96)*, Santa Clara, CA, pp. 54-59.
- Faxon, D., 1993, "When Worlds Collide: Cybersex and Societal Values," *Pixelation*, Vol. 11(3), pp. 5-6.
- Fitts, P., 1954, "The information capacity of the human motor system in controlling the amplitude of movement," *Journal of Experimental Psychology*, Vol. 47(6), pp. 381-391.
- Franzen, A., 2000, "Does Internet Make Us Lonely?" *European Sociological Review*, Vol. 16(4), pp. 427-438.

Fukui, Y., and M. Shimojo, 1992, "Difference in Recognition of Optical Illusion Using Visual and Tactual Sense," *Journal of Robotics and Mechatronics*, Vol. 4(1), pp. 58-62.

Groen, J., and P. Werkhoven, 1998, "Visuomotor Adaptation to Virtual Hand Position in Interactive Virtual Environments," *Presence*, Vol. 7(5), pp. 429-446.

Hannaford, B., and L. Wood, 1989, "Performance Evaluation of a 6 Axis High Fidelity Generalized Force Reflecting Teleoperator," in *Proceedings of NASA Conference on Space Telerobotics*, Pasadena, CA, Vol. II, pp. 87-96.

Harm, D., 2002, "Motion Sickness Neurophysiology, Physiological Correlates and Treatment," in K. Stanney (Ed.), *Handbook of Virtual Environments: Design, Implementation and Applications*, Erlbaum, Mahwah, NJ, pp. 637-661.

Hix, D., and J. Gabbard, 2002, "Usability Engineering of Virtual Environments," in K. Stanney (Ed.), *Handbook of Virtual Environments: Design, Implementation and Applications*, Erlbaum, Mahwah, NJ, pp. 681-699.

L. Hodges, P. Anderson, G. Burdea, H. Hoffman, and B. Rothbaum, 2001, "Treating Psychological and Physical Disorders with VR," *IEEE Computer Graphics and Applications*, 2001 (November/December), pp. 25-33.

Howe, R., and D. Kontarinis, 1992, "Task Performance with a Dextrous Teleoperated Hand System," in *Proceedings of SPIE*, Vol. 1833, Boston, pp. 199-207.

Kennedy, R., N. Lane, K. Berbaum, and M. Lilienthal, "Simulation Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness," *International Journal of Aviation Psychology*, Vol. 3(3), pp. 203-220.

Kennedy, R., K. Stanney, and W. Dunlap, 2000, "Duration and Exposure to Virtual Environments: Sickness Curves During and Across Sessions,"

Presence, Vol. 9(5), pp. 463-472.

Kestenbaum, A., 2000, "New Revision of ANSI Z136.1 (Laser Safety Standards)," LIA Today, Vol. 8(3), online at www.hps.org/hpspublications/articles/ansiz136.I.html.

Lampton, D., B. Knerr, S. Goldberg, J. Bliss, M. Moshell, and B. Blau, 1994, "The Virtual Environment Performance Assessment Battery (VEPAB): Development and Evaluation," Presence, Vol. 3(2), pp. 145-157.

Lampton, D., J. Bliss, and C. Morris, 2002, "Human Performance Measurement in Virtual Environments," in K. Stanney (Ed.), Handbook of Virtual Environments: Design, Implementation and Applications, Erlbaum, Mahwah, NJ, pp. 701-720.

Laurel, B., 1992, "On Finger-Flying & Other Faulty Motions," in Cyberarts Conference, Pasadena, CA, pp. 286-291.

Latham, R., 2001, "Tutorial: A Brief Introduction to Simulation Sickness and Motion Programming," Real Time Graphics, Vol. 9(10), pp. 3-5.

Lecuyer, A., J.-M. Burkhardt, S. Coquillart, and P. Coiffet, 2001, "Boundary of Illusion: An Experiment of Sensory Integration with a Pseudo-Haptic System," in Proceedings of IEEE Virtual Reality 2001, pp. 115-122.

Lowe, W., 1992, "Virtual Sex," Playboy, p. 164.

Massimino, M., and T. Sheridan, 1989, "Variable Force and Visual Feedback Effects on Teleoperator Man/Machine Performance," in Proceedings of NASA Conference on Space Telerobotics, Pasadena, CA, Vol. 1, pp. 89-98.

Naval Research Laboratory, 2002, "Evaluation of Virtual Environment Navigation," online at www.ait.nrl.navy.mil/vrlab/projects/NavEval/NavEvalOverview.pdf.

Piantanida, T., D. Bowman, and J. Gille, 1993, "Human Perceptual Issues and Virtual Reality," Virtual Reality Systems, Vol. 1(1), pp. 43-52.

Ranadive, V., 1979, "Video Resolution, Frame Rate and Grayscale Tradeoffs under limited Bandwidth for Undersea Teleoperation," SM Thesis, Massachusetts Institute of Technology, Cambridge, MA.

Richard, P., G. Birebent, P. Coiffet, G. Burdea, D. Gomez, and N. Langrana, 1996, "Effect of Frame Rate and Force Feedback on Virtual Object Manipulation," *Presence*, Vol. 5(1), pp. 95-108.

Schroeder, R., 1997, "Networked Worlds: Social Aspects of Multi-User Virtual Reality Technology," *Sociological Research Online*, Vol. 2(4), at www.socresonline.org.uk/socresonline/2/4/5.html.

Stanney, K., and P. Hash, 1998, "Locus of User-Initiated Control in Virtual Environments: Influence on Cybersickness," *Presence*, Vol. 7(5), pp. 447-459.

Stanney, K., R. Mourant, and R. Kennedy, 1998, "Human Factors Issues in Virtual Environments: A Review of the Literature," *Presence*, Vol. 7(4), pp. 327-351.

Stanney, K., R. Kennedy, and K. Kingdon, 2002, "Virtual Reality Usage Protocols," in K. Stanney (Ed.), *Handbook of Virtual Environments: Design, Implementation and Applications*, Erlbaum, Mahwah, NJ, pp. 721-730.

Stockburger, D., 1996, *Introductory Statistics: Concepts, Models, and Applications*, online at www.psychstat.smsu.edu/introbook/sbkOO.htm.

Stoffregen, T., M. Draper, R. Kennedy, and D. Compton, 2002, "Vestibular Adaptation and Aftereffects," in K. Stanney (Ed.), *Handbook of Virtual Environments: Design, Implementation and Applications*, Erlbaum, Mahwah, NJ, pp. 773-790.

Swan, E., J. Gabbard, D. Hix, R. Schulman, and K. P. Kim, 2003, A Comparative Study of User Performance in a Map-Based Virtual Environment," Proc. of IEEE VR 2003, Los Angeles, CA, pp. 259-266.

U.S. Food and Drug Administration, 2000, "Guidance on Electronic Products which Emit Radiation," Center for Devices and Radiological Health, online at www.fda.gov/cdrh/radhlth/.

Viirre, E., and D. Bush, 2002, "Direct Effects of Virtual Environments on Users," in K. Stanney (Ed.), *Handbook of Virtual Environments: Design, Implementation and Applications*, Erlbaum, Mahwah, NJ, pp. 581-588.

Wann, J., and M. Mon-Williams, 2002, "Measurement of Visual Aftereffects Following Virtual Environment Exposure," in K. Stanney (Ed.), *Handbook of Virtual Environments: Design, Implementation and Applications*, Erlbaum, Mahwah, NJ, pp. 731-749.

Watson, B., N. Walker, W. Ribarsky, and V. Spaulding, 1998, "Effects of Variation in System Responsiveness on User Performance in Virtual Environments," *Human Factors*, Vol. 40(3), pp. 403-414.

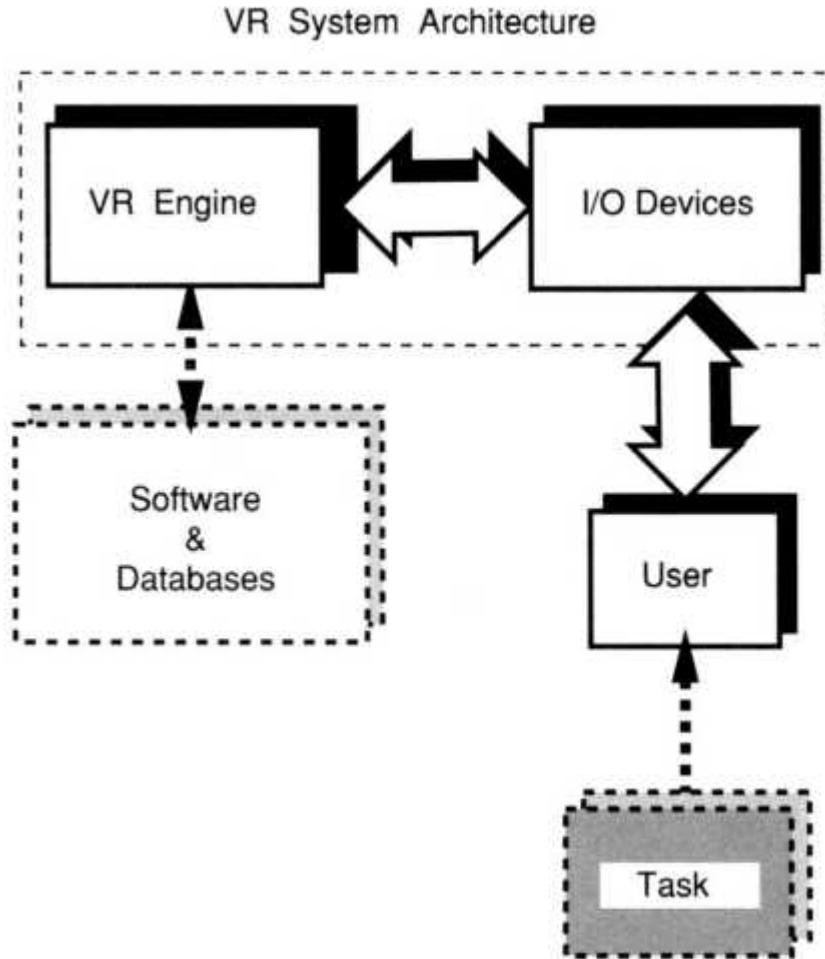
Watson, B., N. Walker, W. Ribarsky, and V. Spaulding, 1999, "Managing Temporal Detail in Virtual Environments: Relating System Responsiveness to Feedback," in *Proceedings of ACM Computer Human Interaction'99 Conference*, pp. 280-281.

Welch, R., 1978, *Perceptual Modification: Adapting to Altered Sensory Environments*, Academic Press, New York.

Welch, R., 2002, "Adapting to Virtual Environments," in K. Stanney (Ed.), *Handbook of Virtual Environments: Design, Implementation and Applications*, Erlbaum, Mahwah, NJ, pp. 619-636.

CHAPTER 8

TRADITIONAL VR APPLICATIONS



The previous chapters showed how immersion and interactivity improve task learning, while hardware and software modularity bring economy of scale. Additionally, multiuser networking allows team collaboration for both application development and simulation. Thus, in principle, VR is ideal in applications ranging from military simulations to arts and education, and from medicine to manufacturing. All these areas share a

need for flexibility, realism, and group participation at reduced simulation costs.

In practice, however, adoption of VR has been slowed by technology shortcomings, a lack of "killer" applications, and marketplace conservatism. Many valuable ideas are still in the prototype and pilot phases and have not progressed to full commercial use. VR applications can thus be classified as a function of their progression from concept to market. Brooks [1999], looking at the degree of maturity of VR applications, classified them as either demonstrations, in pilot production, or in production. A VR application in the production stage is in the hands of the end user, doing real work, and without further testing.

In 1991 there were no commercial VR applications since dedicated hardware was still being developed. A worldwide VR market survey conducted in 1993 identified 805 projects [Helsel and Doherty, 1993]. The survey statistics showed the dominant application sector to be entertainment, the arts (audiovisual), and education, with a total of 172 projects. Another important sector was the military and aerospace (102 sites), where VR can produce tremendous cost savings. A field with fewer projects, but with higher importance in terms of saving human lives, was medicine (49 sites).

Almost a decade later, another survey grouping visualization and VR systems estimated that about 2700 people worldwide were end users of such technology [CyberEdge Information Services Inc. 2000]. The number doubled if one counted the developers of visualization and VR technology. By 2002 this number reached hundreds of thousands [CyberEdge Information Services Inc. 2002]. In view of the widespread use of computers in our society, these numbers are small. While the survey identified users on six continents, the market is dominated by North America and Europe, as seen in Figure 8.1 [CyberEdge Information Services, 2002]. The largest application domain in 2002 was education and training (which included medical education, military training, and university education). Other significant uses of VR were research (not

surprisingly), but also manufacturing, entertainment and business services (such as marketing, trade shows, and architectural planning).

The selection of VR applications discussed in this book gives preference to applications in the production phase. Pertinent studies on the results of VR use are included whenever possible. Also included are projects in the pilot phase that illustrate promising application domains. Our goal is to educate the reader on what VR solutions exist commercially, what advantages can be gained from their use, and what other applications are likely to follow. Traditional VR application areas, such as medicine, education/arts/entertainment, and the military, are discussed in this chapter. Emerging VR applications in manufacturing, robotics, and data visualization make up the subject of Chapter 9.

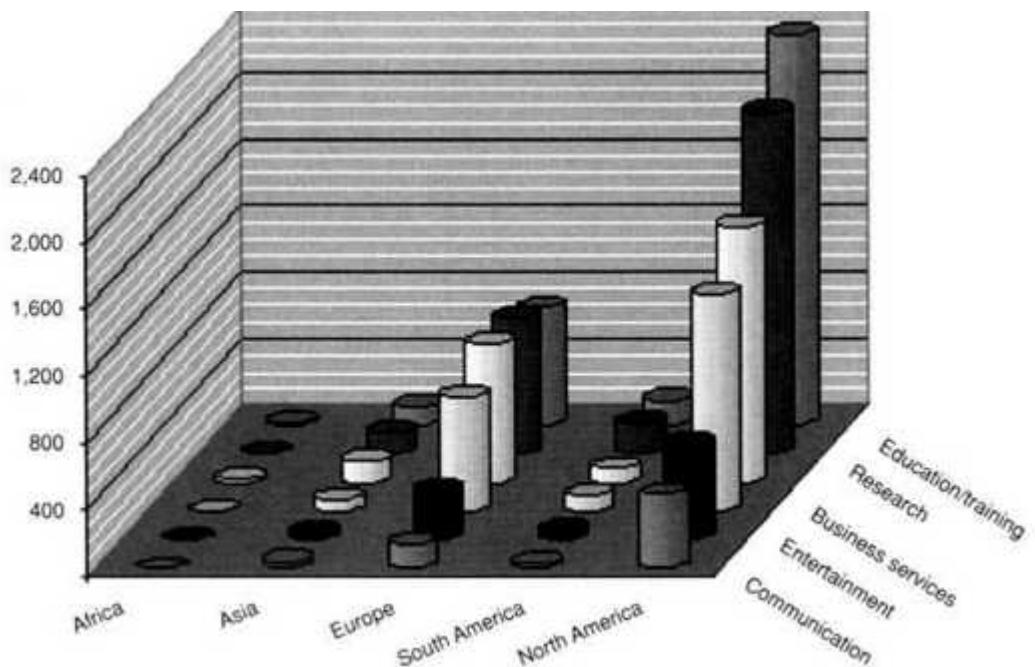


Fig. 8.1 Sectors of visualizationNR applications according to geographic area. From CyberEdge Information Services 2002. © 2002 CyberEdge Information Services. Reprinted by permission.

8.1 MEDICAL APPLICATIONS OF VR

In recent years increased computer and Internet usage in medicine has started to change the way health care is delivered. The power of computing allows online medical education, patient databases, presurgery simulation, use of robotics, remote consultation, digital radiography, expert systems, and so on [Taylor et al., 1996]. The resulting increase in the quality of medical procedures makes up for increased equipment costs. Additionally, Internet computing facilitates patients' access to medical care, either locally or in remote locations where specialists are usually not available.

Medical VR is a form of medical informatics, which has seen a steady, albeit slow growth over the past decade [Satava and Jones, 2002]. The process has been hampered by a lack of standards in data format and validation measures, expensive VR technology, and the requirement to show cost savings put in place by managed healthcare organizations. That is why, of the multibillion dollars of medical informatics expenditures made in 1996, only \$20 million was spent on VR [Anon, 1996]. Despite these limitations, VR brings numerous advantages to the medical community. These include improved medical training (errors made on virtual, rather than real patients; modeling of unusual and rare cases), more realistic certification procedures (objective measures of surgical skill, for example), and more enjoyable treatment (in the case of virtual rehabilitation). The examples given here in the areas of human anatomy, diagnosis, surgical skill training, and virtual rehabilitation should make these advantages more apparent.

8.1.1 Virtual Anatomy

We start our discussion with anatomy teaching, as it is the basis of medical education throughout the world. Until now, the art of teaching medical students has not kept pace with advances in knowledge and technology. The majority of current anatomy courses use textbooks (reminiscent of the atlases introduced in the 16th century) and cadavers for dissection. Unfortunately for the student, textbooks are becoming larger and more expensive, while access to cadavers is harder to get.

Earlier efforts to improve teaching efficiency used digitized photographs and 2D (nonimmersive) anatomy models on CD-ROM [Schwartz, 1993]. By the mid 1990s a much more detailed anatomical database, called the Visible Human, became available. This standard anatomy was created through a collaboration between the National Library of Medicine and the University of Colorado, and was made available without cost on the Web [National Library of Medicine, 2001]. A male and a female cadaver were imaged (Computed tomography, CT; magnetic resonance imaging, MRI) then frozen, imaged again, and sliced from head to toe. Each slice was in turn photographed and digitized, resulting in a multigigabit-size database of 2D images.

With the creation of the Visible Human, researchers and educators throughout the world had a common database on which to test 3D shape extraction and animation algorithms necessary in the development of realistic anatomy course material. Lorensen [2001] at the General Electric Research and Development Center developed a marching-cubes algorithm for extracting polygonal shapes from the Visible Human volumetric datasets. Input to the algorithm consisted of the 522 slices of the CT images taken on the male cadaver. These slices were indexed, then a connectivity algorithm and a marching-cubes algorithm were used to extract skin, organ, and skeletal structures. Subsequently, the large number of surface triangles was decimated in order to make the model more manageable. Figure 8.2 shows the resulting models [Lorensen, 2001]. The ridges visible on the skin are plastic tubes inserted to aid in the registration and connectivity of the image slices. Note that the elbow portion of the arms is missing due to CT artifacts (the male cadaver was larger than the CT imaging workspace!). These models were then used to create a flythrough key frame animation through the body. Obviously, flying through a body is impossible on real cadavers. Nevertheless, this visually rich VR experience helps understand the complex 3D nature of the human body. Many other researchers have since extracted various organs from the National Library of Medicine database such that a growing online gallery exists [Gold Standard Multimedia, 2000]. These organs or organ systems can be downloaded as QuickTime movies and viewed locally.

Creating animations of 3D organ models based on the Visible Human database is a precursor to developing a computerized anatomy teaching curriculum. An example is the Anatomic VisualizeR created at the University of California at San Diego and now being used by the Uniformed Services University of the Health Sciences (Bethesda, MD) [Rigamonti et al., 2000]. The Anatomic VisualizeR provides a virtual dissection room in which students can explore various aspects of anatomy organized into teaching modules (skull, ear, thorax, abdomen, and neuroscience). Each lesson allows a student wearing active glasses to view an interactive anatomical model rendered in stereo by an SGI workstation. Students can create and manipulate crosssection views, measure and identify structures, change the opacity of organs (for better view of hidden components), draw lines, link modules, and so on. Figure 8.3 is a view of the ear module, showing a larger-than-life model of the intricate inner ear structures. In real life these tiny nerves and blood vessels are difficult to visualize through cadaver dissections. Furthermore, once a cut is made, the anatomy is disrupted, and if another cut is necessary to gain a different view, the problem is compounded. With the Anatomic VisualizeR, such digital slices can be made many times, allowing the student to learn at his or her own pace. Individual bones or vessels can be highlighted (flagged) and enlarged (zoomed in) to help in the development of a 3D mental image of the anatomy. Researchers are currently porting the system to PCs, no doubt in order to reduce cost and to allow more systems to be used simultaneously.

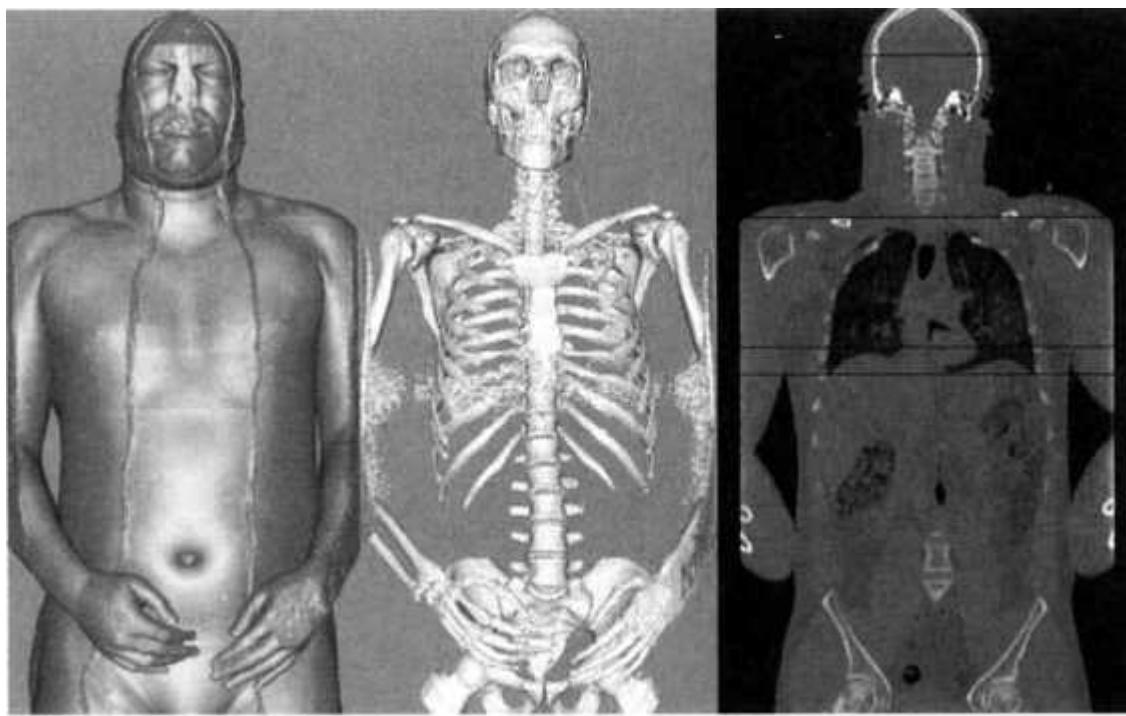


Fig. 8.2 Anatomical reconstruction of the Visible Man. Adapted from Lorensen [2001]. Reprinted by permission.

An alternative way to reduce cost and increase access for students (and others) is to place such teaching modules on the Web. This approach was taken in the Digital Anatomist project undertaken at the University of Washington [Brinkley et al., 1999]. The data are organized as symbolic information and spatial information. The symbolic information contains metadata, textual description, and labels of structures, while the spatial information contains annotated models, 3D images, and animations. The symbolic and spatial information resides on structural information servers, which make it available over the Web. Web clients, using reference, tutoring, or consultant interfaces, can then view anatomical structures remotely at little or no cost. Interestingly, the architectural framework includes an intermediate layer of Digital Anatomist or Digital Neuroscientist intelligent agent. These agents know available data resources and customize them to fit a particular user type and expected use. The researchers have developed a Digital Anatomist Interactive Atlas made of structurally annotated images, which are viewable remotely [University of Washington, 2001].

8.1.2 Triage and Diagnostics

Once anatomical models are developed, they can be modified to add pathology such that students learn to diagnose abnormalities. A good diagnostician is an experienced practitioner who has learned from his or her mistakes (on real patients). Whether diagnostic skill is linked to simple palpation of a patient or to more complex endoscopic exploration, it requires both repetition and access to a sufficient number and variety of patients. Virtual reality can help by allowing repetition under monitored conditions as well as the creation of a large variety of pathologies (including rare cases).

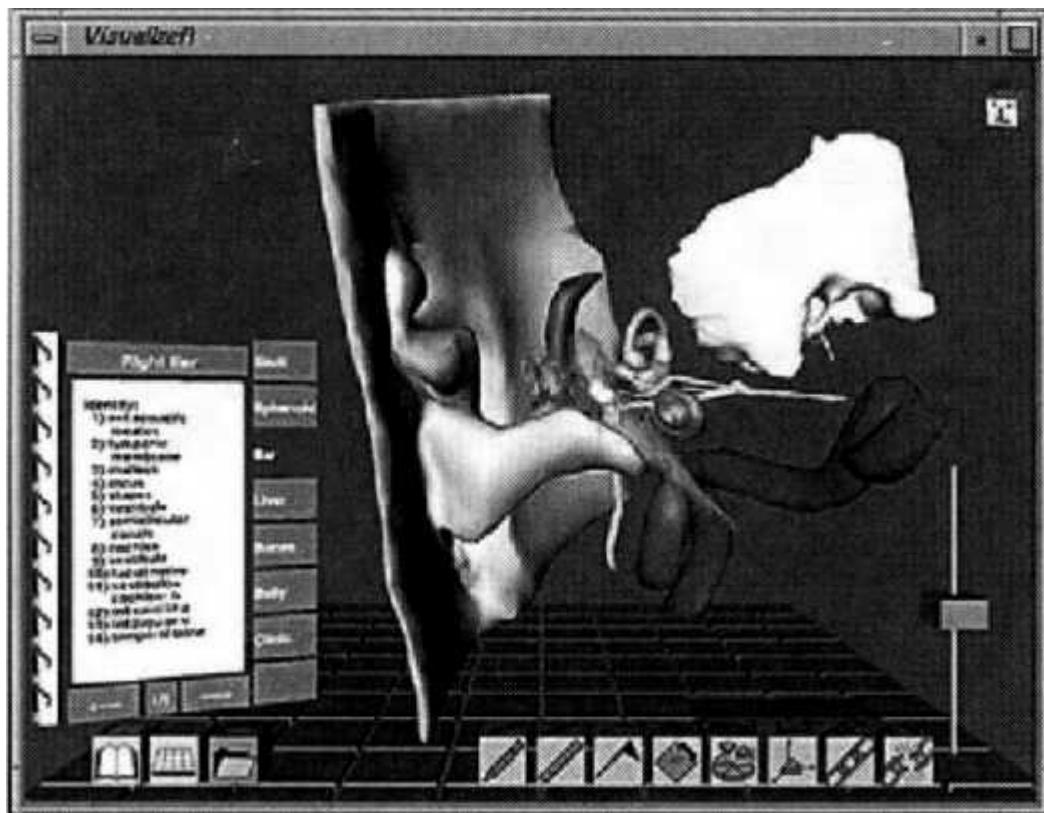


Fig. 8.3 Anatomy teaching module for the inner ear based on the Visible Man database. From Rigamonti et al. [2000]. Reprinted by permission.

Diagnosis is also required for triage, a highly stressful process done on the battlefield or in emergency rooms as a step preceding administration of treatment. In an emergency medical situation tasks need to be done not only

correctly, but also fast. An extreme scenario is training to handle a large number of casualties due to a germ warfare attack.

8.1.2.1 Emergency Medical Response to Bioterrorism. Such training is required in most large population areas such that first responders learn to diagnose illnesses, administer first aid, or send patients to specialized medical facilities. The current live training methods are inadequate due to high cost and reduced realism. A case in point is the simulated casualty (a volunteer), who, unlike real bioterrorism situations, never dies. Virtual reality can improve training since casualty avatars can be programmed to exhibit realistic physiological responses and if untreated, die. Furthermore, costs are reduced since VR training can be done locally and emergency response personnel need not travel away from their base.

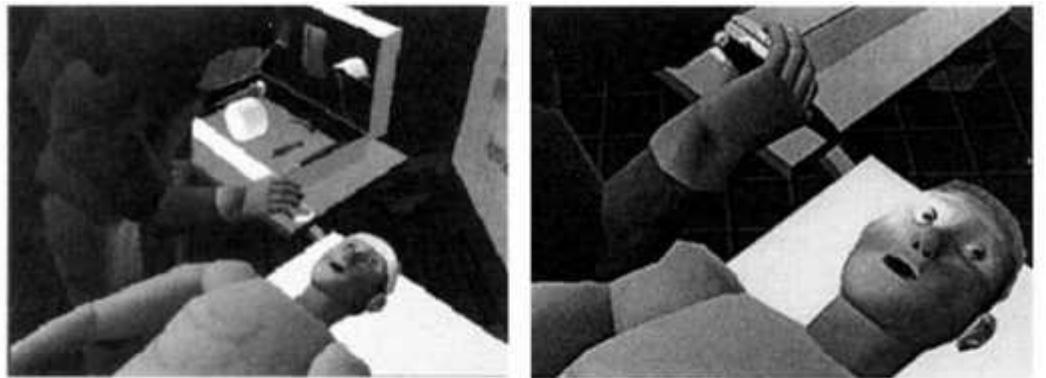
Stansfield and her colleagues at the Sandia National Laboratories [Stansfield et al., 2000] developed a VR training system for first responders in a biowarfare attack called BioSimMER. Trainees put on HMDs and trackers (worn on the palms and head) and are immersed in a scenario depicting a small airport. The trainees are mapped to avatars clad in protective gear, as illustrated in Figure 8.4a, and interact with the simulation through objects or speech. These objects (such as a flashlight, scissors, or syringe) have intelligence. Since the trainee lacks dexterity (they do not have a sensing glove), the objects are attracted to the avatar's palm. Furthermore, they are queried by the simulation such that data are provided on their status (flashlight turned on/off, for example). The virtual patients are driven by a state machine (finite-state automata) in order to increase realism. They change state based on the progression of a disease or in response to the user's input. The change of state is also specific to the disease; for example, eye reflex (tested by the trainee in Fig. 8.4b) is absent in case of head trauma. Conversely, if the casualty has a collapsed lung (pneumothorax), then the avatar's skin changes color, turning progressively blue.

Figure 8.4c shows the computing hardware architecture that supports the simulation. The various computers are connected on an LAN and use multicast communication. The trainee tracker data are sent to a workstation,

which renders the scene on the trainee's HMD. Voice input from the trainee is sent to a voice recognition module, which then feeds the casualty simulation module driving the casualty state machine. A second user (instructor) is similarly immersed in the simulation in order to monitor its progress.

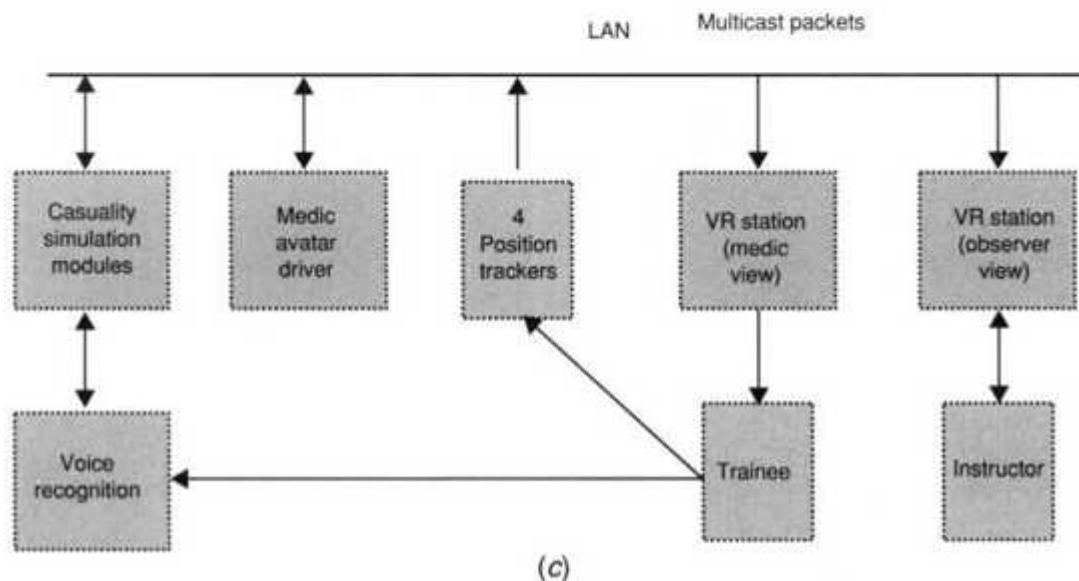
The researchers reported on an evaluation study of BioSimMER performed in 1999 using 23 emergency medical technicians (EMTs). Of these, 20 subjects were paramedics and 3 were instructors, 13 subjects had no prior experience with VR/flight simulators. The work experience was less than 10 years for 13 subjects, the remainder having more than 10 years of EMT experience. They were asked to perform a subjective evaluation of the simulation by filling out a system assessment questionnaire. Results showed that the subjects were very satisfied with the vital signs and the patient's appearance (4.5 point of a maximum 5). The lowest scores were given to the handling of objects (3.2) and the simulation level of difficulty (2.5). The study thus showed the need for a more natural way of handling objects and a more complex, in-depth scenario in order to provide an optimal training experience. Encouragingly, however, the subjects thought that the training and scenario were very applicable to their responsibilities as first responders (4.2 points average).

8.1.2.2 Prostate Palpation. This is a standard diagnostic tool used to detect malignancies in the prostate. The procedure, called digital rectal examination (DRE), is performed on men to determine the state of their prostate. There are four possible diagnoses: normal, enlarged (benign), incipient malignancy (single nodule), and advanced malignancy (cluster of nodules). Since prostate cancer is the second-leading cause of death among men (25% of patients die from the disease), DRE skills and diagnosis accuracy are obviously important. Nevertheless, the procedure is taught in medical schools in primitive ways (using rubber models, limited training on patients, no online data on student proficiency, and no confidence for the graduating student).



(a)

(b)



(c)

Fig. 8.4 Training in emergency medical response to bioterrorism. (a) Trainee's avatar in protective suit; (b) Detail of a virtual casualty. From Stansfield et al. [2000]. © 2000 Massachusetts Institute of Technology. Reprinted by permission. (c) System architecture. Adapted from Stansfield et al. [1998]. © 1998 IEEE. Reprinted by permission.

Researchers at Rutgers University and the University of Medicine and Dentistry of New Jersey (UMDNJ) developed a VR-based DRE training system [Burdea et al., 1999]. It consisted of a PHANToM interface, an SGI workstation, and a motionrestricting mechanical board. The board had a circular orifice through which the trainee had to put his or her index finger prior to insertion in the PHANToM thimble. A lower abdomen model was purchased from a commercial database and modified to include four prostate 3D models. A low-resolution (200 vertices) semicircular upper

surface of the prostate was created using OpenGL. This surface had a medial indentation dividing the model into east-west lobes. This represented the anatomical landmark used by physicians to mentally register the position of their index fingertip on the prostate surface during DREs. Malignancies were modeled as spheres and placed close to the surface, as illustrated in Figures 8.5a and 8.5.b. They could be placed randomly on the north-south or east-west lobes of the prostate. Force feedback during palpation was programmed in GHOST with an optimized vertex search collision detection approach that took advantage of the prostate's circular geometry.

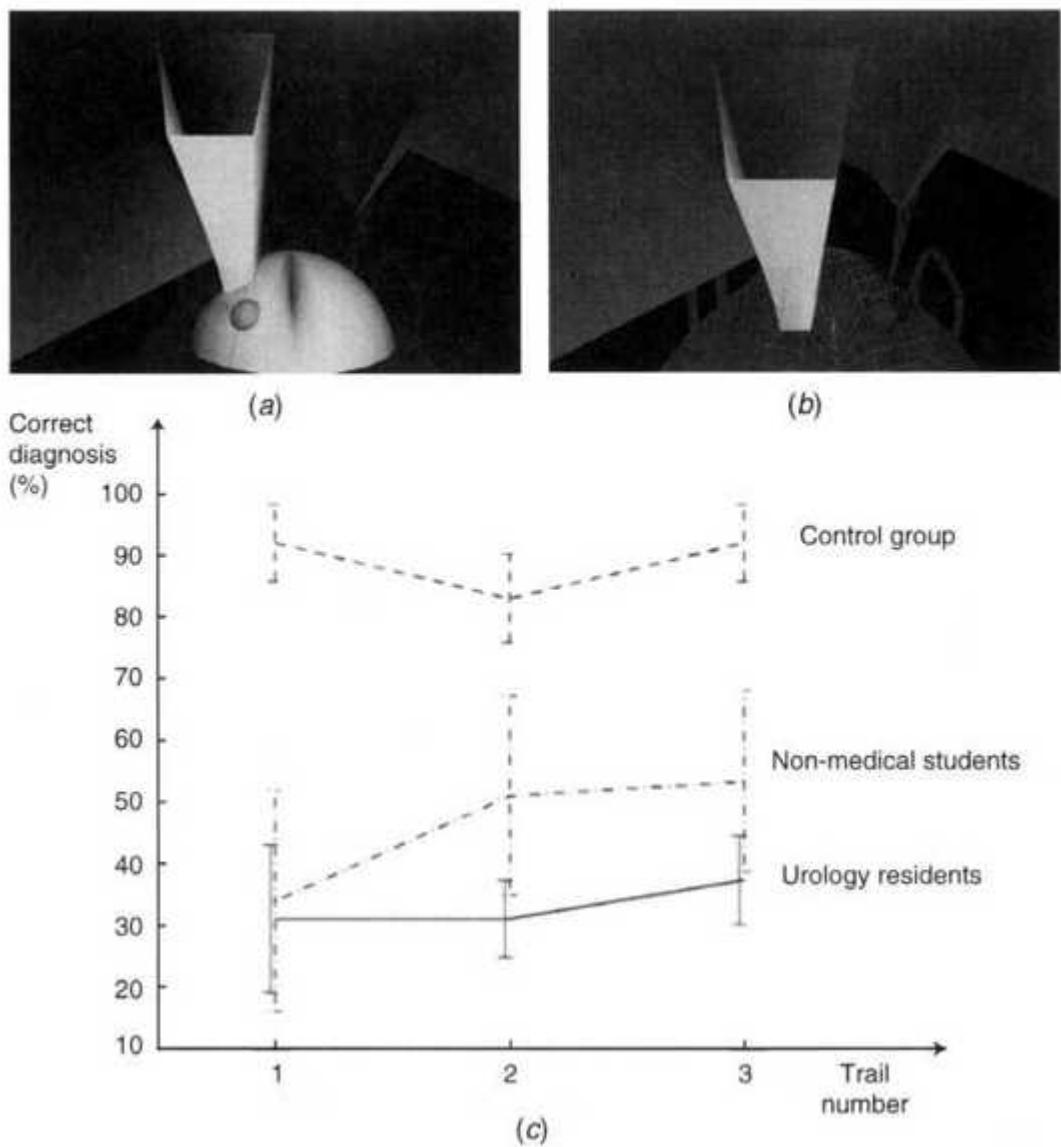


Fig. 8.5 Training in prostate palpation: (a) prostate model with a single nodule; (b) transparent prostate model showing a cluster of nodules; (c) human factors study showing diagnostic task learning. From Burdea et al. [1999]. © 1999 IEEE. Reprinted by permission.

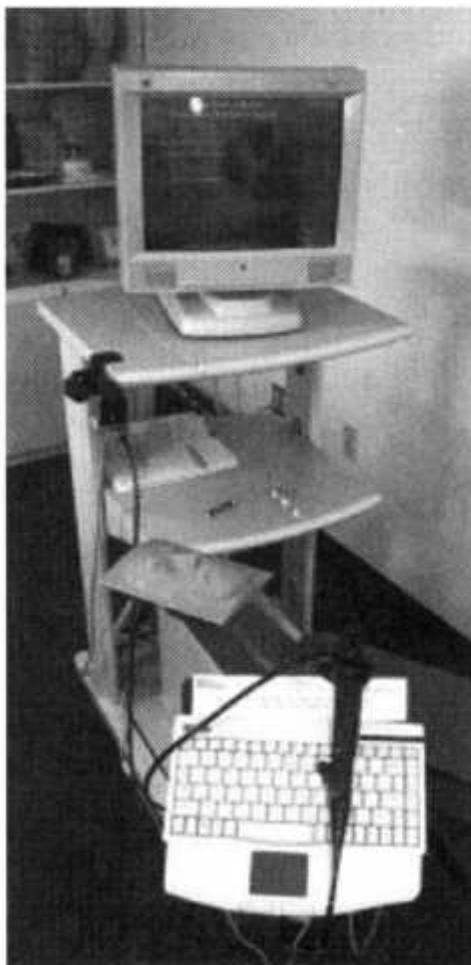
A human factors study was performed in 1999 to determine the efficacy of the DRE simulator in training for the procedure. The study's experimental design used three groups of subjects. The first experimental group consisted of 22 novice users (Rutgers undergraduate students) and the second group consisted of four experts (UMDNJ urology residents). A third group, used as control, was made up of experts (other urology residents) performing the traditional exam on rubber models. Both experimental groups used the PHANToM interface, but only the experts had the motion restriction board attached making the task more difficult. The subjects in the experimental groups first trained for 5 minutes on the use of the PHANToM and then they palpated virtual prostates that were visible on the screen. During this initial learning phase the prostate was rendered in wireframe mode to help trainees locate the malignancies. Subsequently, the simulation switched to an examination mode with a blank screen such that diagnosis was based solely on haptics (as in real DREs). Each subject in the experimental groups was given 12 randomized cases to diagnose, and the variables measured were time to diagnose (seconds), the diagnosis given (normal, enlarged, incipient, and advanced malignancy), and the correct diagnostic. Results showed that both experimental groups were able to distinguish between malignant and nonmalignant cases (novice 67% and experts 56%). Unsurprisingly, the experts were uniformly faster to diagnose, being fastest in detecting incipient malignancies (39 sec on average). Both groups were, however, significantly slower than the control group, who were able to diagnose about five times faster, and more accurately. This is again not surprising since these experts were using a mechanical system that did not randomize the position of the nodules on the prostate rubber model and provided richer haptic feedback (the PHANToM used in the study did not output torques). Perhaps as interesting is the way subjects learned the task, a process illustrated by Figure 8.5c. It can be seen that there was significant learning for the novice subjects, especially between the first and the second trial. However, the urology residents using the PHANToM had no learning

between the first and the second trial, but did improve their diagnosis on their third trial. These experts did not learn the task, rather they learned the system (its limitations). Having understood the simplified model and imperfect haptics, they compensated and adapted. There was also a reduction in their group standard deviation between the first and the last trial, signifying more uniformity in task performance.

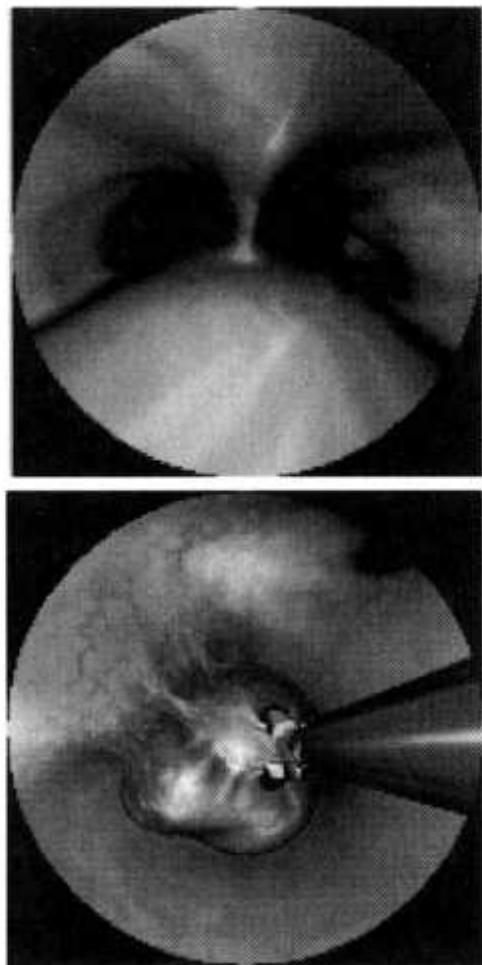
8.1.2.3 Endoscopic Examinations. These are routinely done to detect cancer or diagnose other diseases affecting various parts of the body. The procedures are performed by inserting a flexible viewing device (the endoscope) in body cavities and viewing the images of the body interior displayed on an adjacent monitor. Depending on the region of interest, such examinations are called colonoscopies (for the lower intestine), bronchoscopies (for the airway-nasal area, vocal chords, trachea, and bronchioles), and angioplasties (for the blood circulatory system).

Regardless of their name, endoscopic examinations are invasive (since the endoscope is inserted into the patient), uncomfortable (requiring patient anesthesia), and can lead to injury and complications (even death) if the endoscope scrapes or perforates the cavity wall. Thus the procedure requires good reasoning and manual skills (to maneuver the end of the endoscope from the point of entry to the organ inspected). Such skills are currently taught on mechanical models of the anatomy of interest, on animals, or on patients. Mechanical models are simplistic and cannot faithfully reproduce the patient's reactions nor adequately log errors made by the physician-in-training. Animals have different anatomy, and their use for training is controversial. As a consequence, patients are the ones who are unnecessarily injured (sometimes killed) in order for the physicians to gain the necessary endoscopy skills. Later, if they do not perform a certain number of such procedures every year, they lose these skills. A report published by the American Heart Association found that doctors who perform fewer than 70 angioplasties every year have a 69% higher rate of complications in their patients than doctors who perform more than 270 such procedures every year [Hirshfeld et al., 1998]. This applies to other endoscopic procedures as well, and it is also true for surgery (discussed later in this chapter).

The foregoing discussion highlights the need for medical simulators that would allow physicians to make errors without affecting real patients, would allow physicians to resharpen their skills periodically, and could be used in medical certification. In 1998 HT Medical Systems Inc. (now part of Immersion Co.) introduced the PreOp bronchoscopy simulator, shown in Figure 8.6a [HT Medical Systems Inc., 1998a]. The system consists of a real fiber-optics flexible endoscope introduced in a robotic interface that provides haptic feedback, and a PC running the VR simulation. The graphics consists of a textured anatomical model of the airway (based on the Visible Man database), which responds dynamically to trainee's actions (see Fig. 8.6b). For example, the tissue deforms and the "patient" coughs if the tip of the virtual endoscope touches the airway wall. At the same time, the trainee feels resistance due to the actuators inside the robotic interface in which the endoscope was inserted. Other scenarios, such as bleeding, obstructed view due to airway secretions, or the presence of malignancies, are also simulated. Software running on the PC measures the trainee's performance in terms of time taken for the procedure, errors (hitting the wall or wedging), number of airway segments left uninspected, and so on. These are compared with benchmarks obtained from experts performing the procedure and help quantify the trainee's learning and skill level.

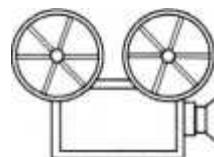


(a)



(b)

Fig. 8.6 Virtual bronchoscopy: (a) simulator system; (b) views of the tracheobronchial anatomy. From HT Medical Systems Inc. [1998a]. © Immersion Co. Reprinted by permission.



VC 8.1

Colt and his colleagues [2001] performed a human factors study to determine the efficacy of the PreOp simulator in training novice physicians. They used an experimental group of five physicians-in-training with minimal exposure to flexible fiberoptic bronchoscopy (FFB) and a control

group of four experienced physicians. These control subjects had performed over 200 FFB procedures each prior to the study. The subjects in the experimental group had an 8-hour training on FFB using the PreOp system, which included 4 hours of individual practice. The experimental task was a thorough inspection of the airway of a normal virtual patient. The measured variables were speed (duration of bronchoscopy), dexterity (number of contacts with the bronchial wall), and accuracy (segments missed). Results showed that the trainees' dexterity and accuracy improved significantly following VR training. There was no change in the duration of the procedure. Subsequently their performance was compared with the control group, who had 30 minutes of familiarization training on the PreOp simulator. The experts performed the FFB procedure faster, no doubt a result of their years of prior training. However, they missed significantly more segments (29% vs. 4.5% for the experimental group). Thus their accuracy was poorer than the VR-trained novices! The researchers then asked both groups to perform the same procedure on a mannequin, measuring speed and accuracy. This time the trainees and the experts had about the same speed in performing the procedure (5 minutes 47 seconds vs. 5 minutes 11 seconds, respectively). Again the accuracy of the VR-trained novices exceeded that of the experts (no segments missed vs. 17% misses, respectively).

Improving physician's skills in performing invasive endoscopic procedures is one way of reducing the patient's discomfort and complications. A more radical way would be to make the procedure noninvasive. A case in point is video colonoscopy, which is done over two million times in the United States every year [Satava and Jones, 2002]. This procedure is performed to detect polyps on the colon wall and is literally a lifesaver because colon cancer detected early is 90% treatable. However, optical colonoscopy requires anesthesia, is expensive, lengthy (2 hours or more), and requires bowel cleansing (enema). As a consequence, colonoscopy is largely avoided by the population. Thus, until recently, there was no mass-screening mechanism available for colon cancer.

Viatronix Inc., a startup company formed by researchers at the State University of New York (SUNY) at Stony Brook, developed a virtual

colonoscopy procedure which is entirely noninvasive [Wax, 2002]. The patient takes a contrasting substance and undergoes a normal helical CT scan, which can image the entire colon in seconds. Subsequently, the 2D slices are assembled in a 3D model of the colon using a segmentation algorithm developed at SUNY. Unlike earlier manual segmentation, this automatic segmentation algorithm extracts the colon wall contour through electronic cleansing. This process digitally removes the liquids and stool residue from the colon interior and applies intensity filters to improve image quality [Sato et al., 2000]. This process takes approximately 15 minutes on a PC. Subsequently, a navigation algorithm allows a virtual camera to travel the whole length of the colon (something often not possible in optical colonoscopy). This algorithm automatically extracts the view trajectory and presents it on a graphical user interface, such as the one shown in Figure 8.7 [Wax, 2002]. The 3D model of the colon is presented in the center window together with various buttons to measure polyps, change their transparency ("virtual biopsy"), volumetrically render the surrounding tissue behind the colon wall, etc. Additional 2D images on the right side show 2D sections in the original CT data that correspond to the current camera position inside the virtual colon. A 3D model of the entire colon is rendered in the left side of the graphical user interface, showing a top view that specifies the current position and view direction.

Thousands of virtual colonoscopies done without anesthesia have been performed since its commercial introduction in 2001 [Marchese, 2002]. The initial clinical trials showed that the detection rate for virtual colonoscopy is the same as for optical colonoscopy for polyps larger than 5 mm in diameter. However, virtual colonoscopy was able to find some 3-mm polyps that were missed in the initial optical colonoscopy examinations. At the time of this writing, the American Cancer Society is evaluating the ongoing clinical studies in order to devise standards and eventually add virtual colonoscopy to its list of recommended colon cancer screening procedures.

8.1.3 Surgery

If the diagnostic procedures described in the foregoing lead to the conclusion that the patient has a tumor, he or she will most probably undergo surgery. The use of VR in surgery affects a number of distinct areas. These include training anesthesiologists and nurses in intravenous procedures, open and minimally invasive procedure training for new surgeons, surgical planning of complex procedures, navigational and informational aids during surgery, and prediction of operational outcomes [Rosen et al., 1996].



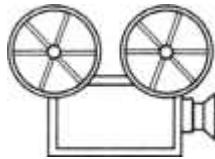
Fig. 8.7 Virtual colonoscopy graphical user interface. From Wax [2002]. Reprinted by permission.

A novice surgeon training on real cadavers cannot repeat a given procedure in case of a mistake, as the body organs have been altered. The learning curve of a surgeon thus continues long after graduation. An internationally known expert in eye surgery has told the authors that "it takes thousands of procedures to become really proficient." Who would

want to be one of the first hundreds of cases? What is needed are simulators that allow surgeons to learn by repetition, the way airline pilots do, while at the same time reducing the risks for patients [Satava, 1993]. Such surgical simulators have started to become commercially available and to be gradually incorporated in teaching curricula. We now describe available simulators for intravenous procedures, open surgery, and minimally invasive surgery (MIS).

8.1.3.1 Intravenous (IV) Procedures. These procedures are performed on approximately 80% of patients admitted to hospitals. They range from simple needle insertion to draw blood for laboratory testing, to catheterization (insertion of a catheter tube in the vein), to lumbar anesthesia. The classic teaching methods involve sticking a needle in an orange or a plastic model, or (most of the time) learning on patients. This poor approach, coupled with the high rate of change in nursing staff and the difficulty in verifying the skill of new hires, has led to an increase in hospital-acquired infections and related deaths [HT Medical Systems Inc., 1998b].

In order to address the need to improve nurses' IV skills, Immersion Medical developed the CathSim intravascular catheterization simulator [Ursino et al., 1999]. As seen in Figure 8.8 [Amato, 2001], the system consists of a special-purpose haptic interface called AccuTouch, a PC, and a desk-top monitor. The trainee holds the device in both hands, as done in a real procedure. His left hand holds a sensorized handle that mimics the patient's anatomy in the area of needle insertion (such as the forearm). The right hand holds a real syringe/catheter assembly, which is mapped to a virtual needle on the screen. When the trainee pushes the mechanical sled inside AccuTouch, the virtual needle presses against the skin of the virtual patient, which is texture-mapped and deforms (for realism). A cross-sectional view of the skin and vein is also displayed in order to help the trainee orient himself. Additional input to the simulation is provided by the AccuTouch handle, which detects the traction action of the thumb to stretch the skin and thus stabilize the vein prior to needle insertion.



VC 8.2

The small resistance at the moment of contact with the skin is fed back to the trainee through a cable and actuator assembly in the AccuTouch box. The actuator is a magnetic particle brake, which applies resistive torques under PC control. The needle can be pushed further into the anatomy of the virtual patient such that the vein is penetrated and virtual blood appears in the needle holder. The physical modeling is based on a threshold that depends on the particular needle gauge selected as well as on the force applied by the trainee. Once the force threshold is exceeded, the needle penetrates the vein and the trainee feels a haptic "pop" (a sudden reduction in resistance). If excessive force is applied, the opposite wall of the vein is traversed, and blood spills out to form a hematoma. This tell-tale sign of a badly executed IV procedure appears on the skin of the virtual patient (as happens to real patients many times!). At that point the trainee gets additional auditory feedback on the mistake since the virtual patient starts to complain. A database records simultaneously the position of the needle/catheter assembly relative to the skin, the position of the needle relative to the catheter, the thumb action on the AccuTouch handle, and the patient's reactions.

Immersion Medical integrated this simulator into teaching modules that simulate various types of patients, from a newborn to normal adults, and from obese patients to drug addicts and geriatric virtual patients. The type of patient dictates the location of choice for the needle insertion as well as the physical anatomical characteristics. The insertion location is the cranium for newborns, as their arm veins are too fragile. Conversely, a geriatric patient will have an increased risk of bleeding, and a drug addict will have veins altered due to frequent chemical IV insertion. In order to further increase the educational value of CathSim, the VR simulation is complemented by preprocedure, site preparation, and postprocedure modules. These use text, video, or multimedia to test the trainee's ability to

select the proper insertion site, properly disinfect it, and apply protective bandage after the IV procedure is finished. Also part of the postprocedure module is feedback on the trainee's performance, including the possibility to replay the simulation. Although no validation studies were available at the time of this writing, Immersion Medical reports satisfaction with their product, which may lead to increased self-confidence among nursing staff and thus increased use by hospitals.

8.1.3.2 Open Surgery. Open surgery is performed whenever a large organ (or portion of one) needs to be removed. The advantages for the surgeon are direct sight of the surgical area (the surgeon looks directly at his or her hands) and good haptic feedback. For the patient, however, open surgery is more invasive than MIS and there is a longer recovery time postsurgery.

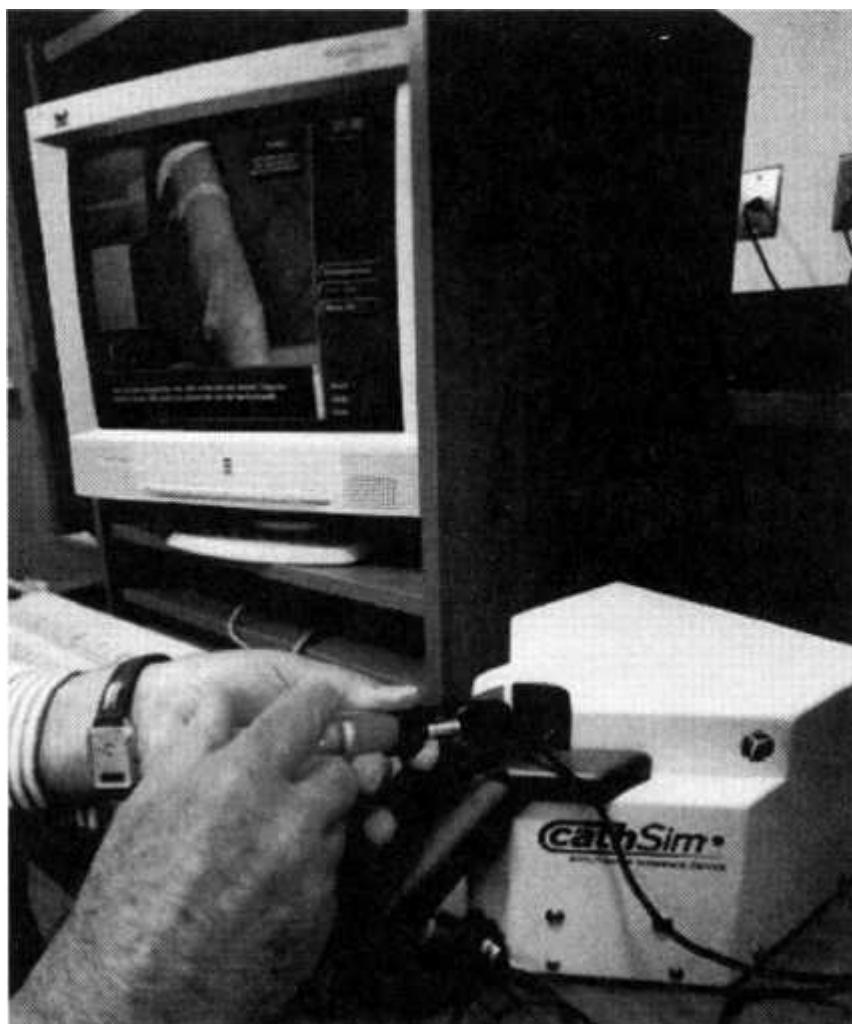


Fig. 8.8 CathSim system view showing the AccuTouch haptic interface and the graphics display of a virtual forearm. From Amato [2001]. © 2001 Massachusetts Institute of Technology. Reprinted by permission.

Examples of open surgery are abdominal procedures, where vessels have to be cut and then reattached after organ removal. Since open surgery predates MIS, it is not surprising that one of the earliest surgical simulators was intended for training in abdominal surgery [Satava, 1993]. The user (surgeon) interacted with a virtual abdomen through two VPL DataGloves, seeing the scene displayed on a stereo HMD and hearing 3D sound. The simulated surgical procedure was a coupling of the ends of a dissected bowel (also called anastomosis). The virtual world included various abdominal organs, the surgical instruments (bowel coupler, syringe, purse string, clamps, needles), as well as the operating table and lights. Interactivity with the simulated environment was very good, as the surgeon picked up the instruments in a very natural way. The frame refresh rate was also high (30 frames/sec). However, limitations in the computing hardware available in the early 1990s led to tradeoffs in the scene complexity (organs appeared cartoonish), simplified organ behavior when the organ was deformed, and lack of haptic feedback.

Boston Dynamics subsequently developed the anastomosis simulator shown in Figure 8.9a [O'Toole et al., 1997]. The availability of commercial haptic interfaces allowed the integration of two PHANTOM arms retrofitted with surgical instruments (tweezers, needle holder, needle, etc.). This increased the tactile realism since the trainee was able to manipulate real surgical instruments. An innovative approach concealed the PHANToM arms from view by using a half-mirror to cover them and placing the PC upside down on top of the simulator (not seen in the figure). In this way a trainee looking down at the procedure area was mapped cognitively to the virtual instruments he or she controlled (see Fig. 8.9b). An increase in simulation realism was also obtained through graphics texturing, by modeling the tubes' deformation (through splines), and by adding a dynamic response to the trainee's actions. For example, if the curved needle used to thread the vessel wall applied too much tangential force, then the vessel would be damaged and blood appeared. As with the

CathSim simulator previously described, the Boston Dynamics anastomosis simulator recorded trainee's actions and scored their performance.

A human factors study was subsequently performed to determine if the anastomosis simulator was sensitive enough to detect differences in surgical skill and whether it was effective as a teaching approach [O'Toole et al., 1998]. The study used two experimental groups of significantly different surgical skill. The first group consisted of 12 novice subjects (medical students) and the second group was composed of 9 expert subjects (vascular surgeons averaging thousands of previously done anastomosis procedures). The two groups were asked to perform an identical task, namely the insertion of a curved needle at four locations on a large vessel wall. The variables measured included task completion time, tissue damage, accuracy in placing the needle, angular error (vs. ideal trajectory), and peak applied force. As expected, the simulator's objective measures showed that there was a statistically significant difference in skill between the two groups. Figure 8.9c graphs the average tissue damage versus trial number. The surgeons had consistently less tissue damage occurrences throughout the trials and performed more uniformly as a group (smaller standard deviation). The subjects who were medical students had a dramatic reduction in the amount of tissue damage over the eight trials, indicating substantial task learning. They also improved significantly in other variables (accuracy, completion time, peak force). There was some learning in the expert group, too, over the first two trials, but this was learning the system, not the task.

8.1.3.3 Minimally Invasive Surgery. This represents a newer way of performing certain types of abdominal, knee, nasal, or intestinal surgery. Patients prefer MIS since it needs smaller incisions than open surgery, thus allowing faster recovery and less chances for complications. For example, a patient having a gall bladder removed through MIS can be discharged from hospital on the same day. The same procedure done in open surgery would require a 1-week stay in the hospital [Anon, 1998]. In 1991 there were 200,000 MIS procedures performed in the United States [Daventry Associates Inc., 1993]. Tens of millions of such procedures are performed today.

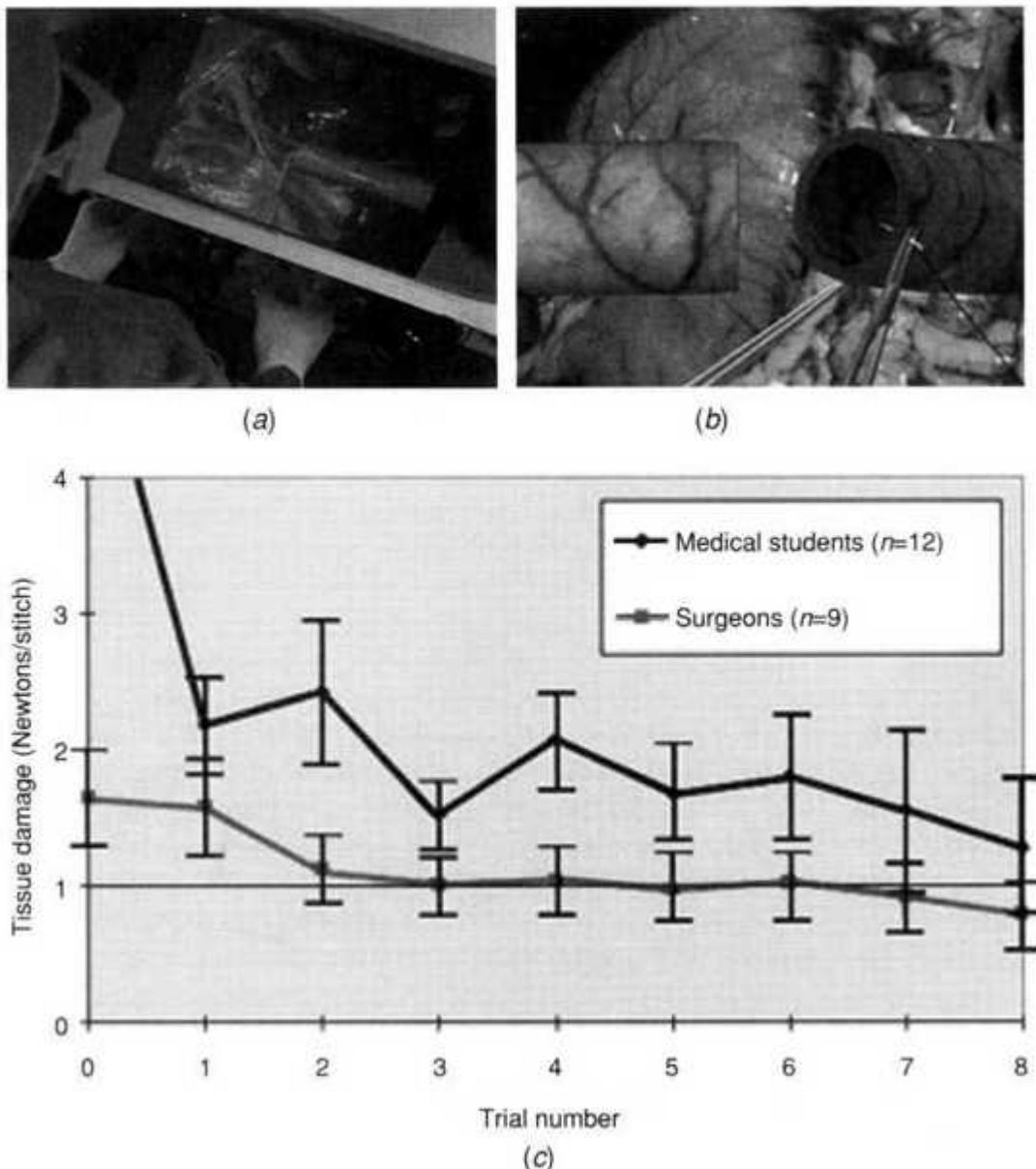


Fig. 8.9 Anastomosis simulator: (a) system view; (b) graphics feedback showing virtual blood vessels and surgical instruments; Images courtesy of Boston Dynamics. (c) subjects' task learning exemplified by tissue damage as a function of trial number. From O'Toole et al. [1997]. Reprinted by permission.

The two major MIS techniques are endoscopic surgery, performed with a flexible endoscope, and laparoscopic surgery, which is done using a rigid assembly of handle, stick, and cutting scissors, called a laparoscope. Both use tiny cameras inserted in the patient's body in order to allow the surgeon

to view the cutting area. During an MIS procedure the surgeon is thus forced to watch a TV monitor and look away from his or her hands, contrary to what happens during open surgery. Furthermore, laparoscopic surgery suffers from the fulcrum effect, which reverses the motion of the cutting element inside the patient's body relative to that of the surgeon's hand holding the handle outside the body. Finally, there is a filtering of haptic feedback due to the laparoscope itself, which distorts the forces felt by the surgeon. These drawbacks require significant surgical dexterity and excellent hand-eye coordination, which are obtained only through intensive training.

The large size of the MIS market, together with an obvious need for training, resulted in a gold rush to produce new simulators. Faced with inadequate training mechanisms in the early 1990s, many surgeons resorted to driving cars in reverse for hours in order to automate their left-right reversal proprioceptive mapping required in laparoscopy. The situation improved somewhat with the introduction of mechanical closed boxes with holes allowing the insertion of laparoscopes and cameras presenting the interior view on a monitor. These boxes allow trainees to practice cutting grapes, foam, chicken tissue, and such. While useful in acquiring basic motor skills, these mechanical boxes are not sensorized, do not measure the duration and economy of motion, nor do they store data in a database. Thus they cannot serve as an objective way to assess a surgeon's MIS skills, either locally or remotely.

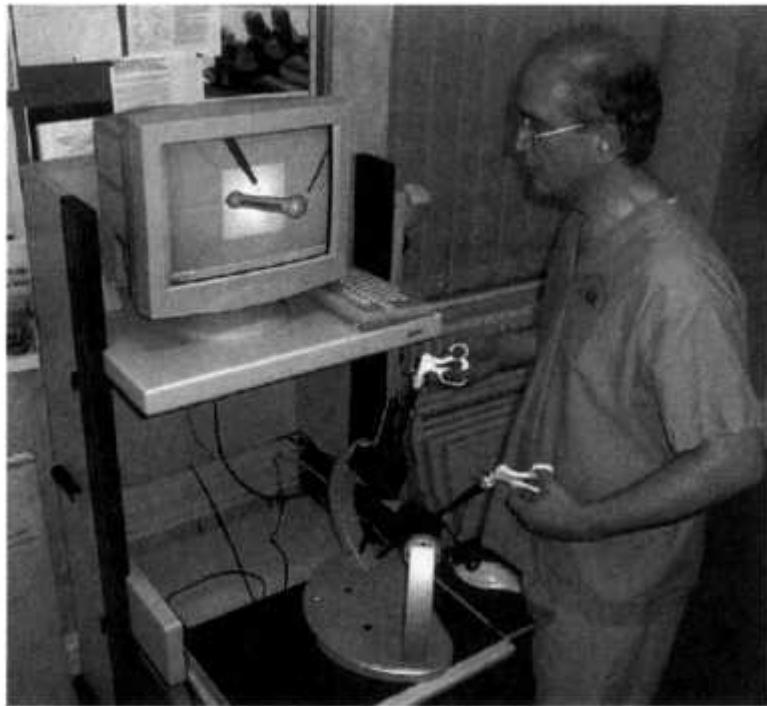
In the mid 1990s, a small company called Virtual Presence Ltd. (Manchester, U.K.) introduced one of the first VR-based MIS trainers [Stone, 1999]. The system, called the Minimally Invasive Surgical Trainer (MIST-VR), is currently marketed by Mentice Medical Simulations (Gothenburg, Sweden). As illustrated in Figure 8.10 [Stone, 2001], the system consists of a special-purpose computer interface, a PC, a monitor, and specialized software. The trainee holds a virtual laparoscopic interface, which has two laparoscopes on a sensorized support. The interface incorporates optical sensors to track the motion of each laparoscope in five degrees of freedom at 1000 Hz. Two rotary axes allow each surgical instrument to pivot about its insertion point, while a third

degree of freedom measures the in-and-out translation of the device. The last two degrees of freedom are rotation about the laparoscope insertion axis and the open/close position of the grip (handle). Communication with the PC is done over a standard RS232 line, allowing very small system latency (1 msec).

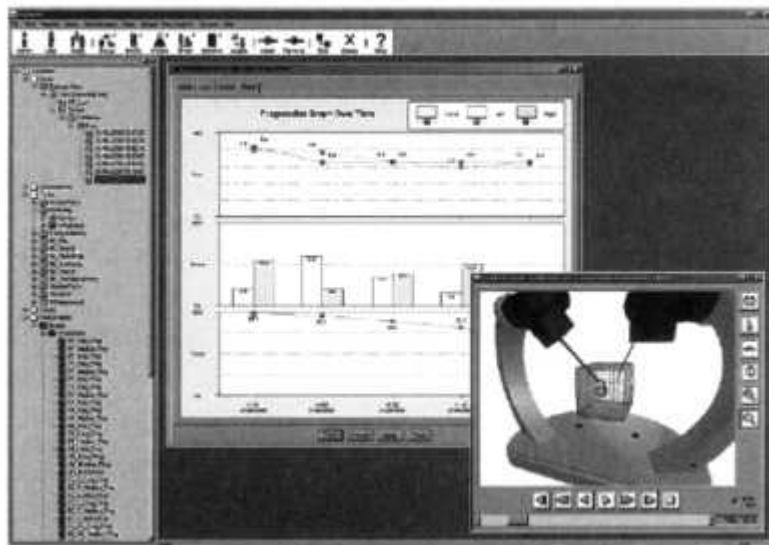
The MIST-VR 3D graphics feedback shows virtual models of the laparoscopes as well as target objects inside a cubic operating area. Training consists in performing six tasks of increasing complexity, which were selected based on an analysis of gall bladder surgery procedures. These tasks are target acquisition, object transfer and placement, object traversal (using both laparoscopes), instrument withdrawal and reinsertion, target diathermy (tissue burning), and finally target manipulation and diathermy. During task execution the system stores objective variables such as total task completion time, subtask completion time, number of errors (movements away from target), and efficiency (economy of motion). These performance data are provided in a report available to the trainee and to the examiner (during examinations) and allow comparative statistics within a group of trainees.

A number of validation studies have been performed in order to determine whether (1) MIST-VR is sensitive enough to be used as a skill assessment tool (can it detect skill differences between novice and expert subjects?) and (2) it is efficacious as a training system (i.e., does it improve subject's performance?). Taffinder and his colleagues at the Imperial College of Medicine in London performed a study using three experimental groups of 10 subjects each [Taffinder et al., 1998]. The first group was composed of experienced surgeons with over 100 laparoscopic procedures done before the study. The second group consisted of trainee surgeons who had not performed laparoscopies, and the third group was composed of nonsurgeons. Each group trained on the MIST-VR system performing the first five tasks. The last task, requiring two-hand manipulation and use of a foot pedal (for diathermy), was subsequently used to assess their surgical skill. Results showed that the experienced surgeons obtained significantly better scores in the MIST-VR test than the two other groups. Specifically, they averaged 4.8 errors compared with 6.3

for trainee surgeons and 7.3 for nonsurgeons. Furthermore, the expert subjects' task completion time of 13.7 sec was significantly shorter than the time taken by trainee surgeons (19.3 sec) and nonsurgeons (21.9 sec). Thus, while all subjects had to perform exactly the same task, their MIST-VR scores mapped to skill. The value of this finding relates to the reproducibility of the test and its use as objective measure of performance.



(a)



(b)

Fig. 8.10 The MIST-VR simulator: (a) virtual laparoscopic interface; (b) graphical user interface. From Stone [2001]. Reprinted by permission.

Subsequently, Gallager and colleagues at Queen's University (Belfast, Northern Ireland) performed a study using two groups of novice surgeons

[Gallager et al., 1999]. Each group consisted of eight subjects. The experimental group underwent training on MIST-VR, performing two complete sessions (sets of six tasks each) within 24 hours. The control group did not train on MIST-VR. Subsequently both groups were evaluated on a real task consisting of cutting precise line patterns on a piece of paper using laparoscopes. The paper was marked with lines spaced at 1-cm intervals and the subjects had to stay within these intervals, otherwise an error was recorded. The paper was placed inside a mechanical training box and the subjects had a view of the paper projected on a monitor. Each trial consisted of 26 cuts and the subjects had to perform 10 such trials. Figure 8.11 [Gallager et al., 1999] shows the number of correct cuts averaged by each group for the first and last trials on the mechanical training box. As can be seen, the group that received MIST-VR training exhibited significant better motor skill than the control group. They were consistently better than the subjects with no MIST-VR training and approached the upper score limit (26 correct incisions). Their standard deviation was also smaller than that of the control group (1.99 vs. 3.16 in trial 10), indicating more uniformity in performance among the group. The subjects in the control group also improved as they learned through repetition, although they did not match the experimental group performance even after 10 trials.

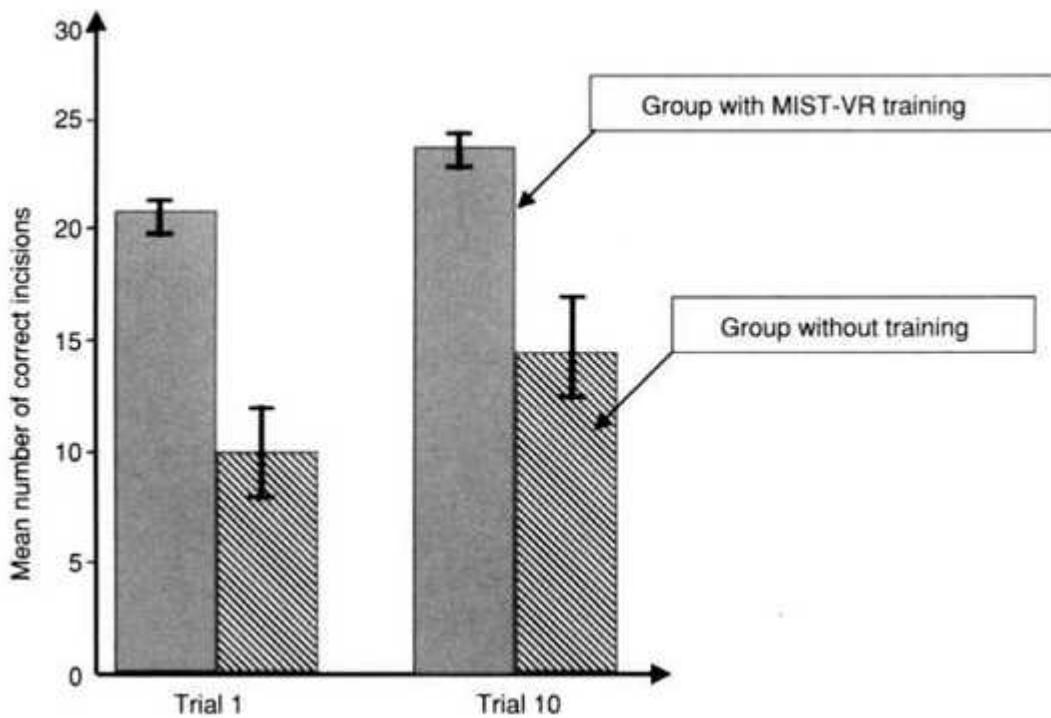


Fig. 8.11 MIST-VR simulator efficacy in skill training reflected in number of correct incisions made by subjects during a 2-minute period. Adapted from Gallager et al. [1999]. Reprinted by permission of Thieme New York.

The MIST-VR system is in use at the European Surgical Institute (Hamburg, Germany). More than 10,000 trainees took classes at the Institute in 2000 [Burger, 2000]. These participants are reported to have improved their basic skills up to 30% on the MIST-VR trainer, and can further their skills through advanced courses offered by the Institute. In a survey conducted among 274 participants, more than half considered that haptics was a missing feature in MIST-VR. This situation has changed with the recent introduction of the Laparoscopic Surgical Workstation that incorporates force feedback for insertion, pitch, yaw, handle twist and grip [Immersion Co., 2002].

8.1.4 Rehabilitation

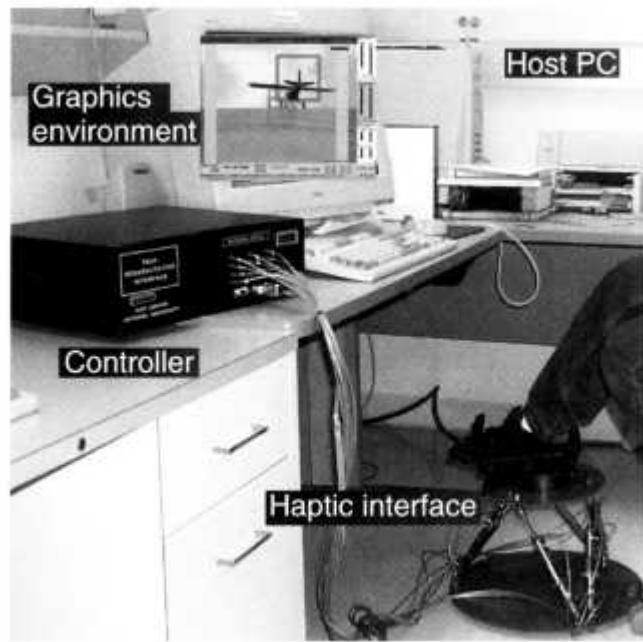
A patient discharged from a hospital following surgery usually has to undergo rehabilitation in order to regain strength, range of motion, walking ability, and so on. Rehabilitation is also prescribed for people who had

fractures or a stroke or who have cognitive or psychological deficits. Thus, depending on the patient population involved, rehabilitation can be classified as orthopedic, neurological (post-stroke or traumatic brain injury), and cognitive. Virtual reality is starting to be used in all these areas, even though it is currently mostly in the pilot or clinical trial phases. By adding virtual reality, the rehabilitation becomes VR-augmented (when used in addition to classical therapy) or VR-based (when replacing classic therapy altogether).

Before discussing particular areas of VR use in rehabilitation, it is useful to consider what advantages it has over traditional therapeutic methods. Patient rehabilitation consists of treatment in outpatient clinics and unsupervised exercises at home. Whether done at home or at a clinic, rehabilitation exercises are very repetitive and thus boring. With no supervision, there is little incentive for the patient to exercise at home. Clinics, on the other hand, are not available in rural or remote areas, and patients have to travel a long distance to get treatment. More than 50 million Americans live in rural areas, but only 10% of therapists practice there [National Rural Health Association, 1999]. Virtual reality can address these limitations by creating an interactive, entertaining environment (through the use of 3D graphics and video games) and allow supervised rehabilitation at home. This type of rehabilitation, called telerehabilitation, gives a remote therapist the possibility of monitoring (in real time or not) the actions of a home-bound patient using the Internet. Other advantages associated with VR-augmented and VR-based rehabilitation are the ability to simulate tasks similar to activities of daily living and to store fine-grained data in online databases. Conversely, VR can depart from reality, thus allowing patients to perform tasks they could otherwise not do and thus improve their morale (a very important component in healing). Finally, virtual rehabilitation can bring economy of scale and reduce costs by allowing a single therapist to monitor several remote patients through multiplexed telerehabilitation.

8.1.4.1 Orthopedic Rehabilitation. This form of rehabilitation follows a fracture, ligament sprain, or hand or knee surgery. One of the most injured parts of the body is the ankle. In 1998 almost one million Americans visited

emergency rooms because of ankle problems [American Association of Orthopedic Surgeons, 2002]. Girone and his colleagues at Rutgers University developed a VR-based rehabilitation system called the Rutgers Ankle [Girone et al., 1999]. As shown in Figure 8.12a, the system consists of a compact robotic platform, an electropneumatic controller, and a PC running the simulation. The robot is of parallel design (Stewart platform), which gives it high strength, and uses double-acting pneumatic actuators to pull-push the foot in all its degrees of freedom. A sensor placed under the foot support measures the forces and torques applied by the patient, which are stored in a database together with the corresponding foot position/orientation.



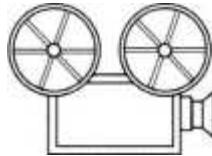
(a)



(b)

Fig. 8.12 Orthopedic rehabilitation of the ankle: (a) System view. From Girone et al. [1999]. © 1999 ASME. Reprinted by permission. (b) Screen view of the virtual airplane piloting exercise. From Deutsch et al. [2001b]. © Rutgers University. Reprinted by permission.

The rehabilitation exercise initially developed for this system involves the piloting of an airplane through 3D hoops (see Fig. 8.12b [Deutsch et al., 2001b]). By aligning the hoops horizontally only, or vertically, or in combination, the therapist can select what particular degree of freedom of the ankle is exercised. The difficulty of a particular exercise is varied by selecting the airplane speed as well as the resistance level provided by the Rutgers Ankle. The bar graphs on the side of the screen visualize in real time the degree of bending of the ankle and associated torques. These graphs change color to alert the therapist whenever the patient surpasses his or her initial abilities.



VC 8.3

In summer 2000 this system underwent a series of pilot trials in an outpatient clinic in New Jersey [Deutsch et al., 2001a]. Three orthopedic patients exercised in sitting position, piloting the airplane with the Rutgers Ankle. They practiced three times a week for 2 weeks, each session lasting approximately 30 minutes. Two of the subjects also underwent classic physical therapy. All patients' range of motion, torque, and coordination were measured at the start and the end of therapy. They all improved in these measures and reported enjoying the VR therapy. Subsequently, the system was upgraded to a two-platform configuration [Boian et al., 2003] and a remote monitoring capability was added [Lewis et al., 2003] prior to further tests in clinics and patients' homes.

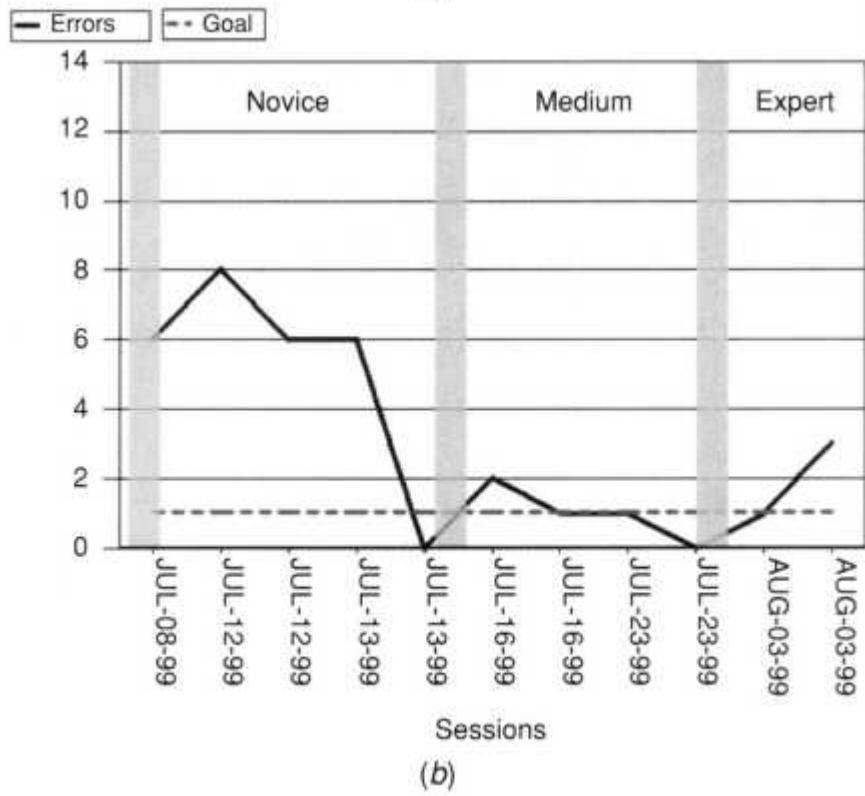
A VR-based orthopedic telerehabilitation system was developed at Rutgers University in collaboration with Stanford Medical School [Popescu et al., 2000; Burdea et al., 2000]. The system consists of a home station and a therapist station connected over the Internet in a client-server architecture. The home station (client), shown in Figure 8.13a, consists of a Rutgers Master II glove, an electropneumatic controller, a host PC, a hands-free microphone array, and a digital camera. The glove measures the real-time position of the thumb, index, middle, and ring fingertips in relation to

the palm and applies forces to resist flexion. Position data provided by the glove and a Polhemus tracker are used to drive a virtual hand rendered by the PC.

A library of modular rehabilitation exercises (programmed in WTK) allows patients to exercise at the impairment or functional levels. Impairments affect range of motion as well as finger grasping force. Their rehabilitation is done through exercises depicting elastic virtual balls, power putty, or a virtual DigiKey (a standard hand rehabilitation device that is similar to a trumpet keyboard with springs, used to improve finger strength). Functional rehabilitation is more complex, and improves hand-eye coordination through exercises such as pegboard and handball. A handball exercise asks the patient to catch and throw a bouncing virtual ball of varying velocity against a wall. The patient is scored (hits and misses) and needs to exercise in a set amount of time. In a pegboard exercise (shown in Fig. 8.13a) the patient needs to pick up nine cylindrical pegs and place them in the holes of a board shown inside a virtual rehabilitation room. In order to help the patient with the perception of depth, the floor of the room has a checkered pattern and the peg projects a shadow. The patient is also helped by auditory and visual cues (change in peg color when it is directly above the hole, and then again when it is correctly inserted in the hole). The remote therapist can vary the exercise difficulty at novice, medium, and expert levels. These correspond to progressively shorter amounts of time allowed for filling the board with pegs and progressively tighter tolerances between the virtual pegs and the holes. An Oracle database is used to store patient performance data locally and then upload it to the therapist station upon demand. Video conferencing is available when the patient needs it or when the therapist wants to supervise the patient's exercise.



(a)



(b)

Fig. 8.13 Orthopedic telerehabilitation. (a) System view. From Popescu et al. [2000]. (b) Case study results for pegboard exercise. From Burdea et al. [2000]. © 2000 IEEE. Reprinted by permission.

The system was tested in summer 1999 on a patient following surgery to his right hand. The patient underwent VR-based rehabilitation at Stanford University (located on the West Coast) and was monitored remotely from Rutgers University (located on the East Coast). During approximately 1 month of therapy the patient exercised on the system during 12 sessions, some with the assistance of a local therapist and some independently. He also performed some manual exercises at home, but never went to a regular rehabilitation clinic. As can be seen from Figure 8.13b, his hand-eye coordination improved such that he made progressively fewer errors on the pegboard exercise. Errors increased temporarily when the exercise difficulty was also increased, which shows the sensitivity of the system. Similar improvements were observed in his finger strength, which went up and surpassed the therapist's set goal [Burdea et al., 2000]. Currently the system is being upgraded to a new haptic glove, the Rutgers Master 11-ND [Bouzit et al., 2002], prior to further patient tests. Communication has also improved by migrating to the higher throughput Internet2. This affects primarily the teleconferencing capability, not the rehabilitation data upload, which is not bandwidth-intensive.

8.1.4.2 Rehabilitation of Post-Stroke Patients. This is a lengthy process that follows a hemorrhage or a blood clot in the brain. Such circulatory accidents cause stroke, which is a leading cause of long-term disability. There are 500,000 new cases each year in the United States [American Stroke Association, 2002]. Patients who survive a stroke can be left with cognitive deficits as well as motor impairments affecting half of their body. Thus either their left or right limbs have a degraded (or totally lacking) ability to move and degraded proprioception. Classic stroke rehabilitation therapy is performed in the acute (first month or so after stroke) and the subacute (5 months thereafter) stages of the disease. Patients who are in the subsequent chronic stage do not receive therapy, and have to endure a life of disability. Recent medical research has shown that the brain adapts and can repair itself years after stroke (so-called brain plasticity) as long as prolonged and intensive therapy is administered [Kopp et al., 1999]. However, the medical community is ill prepared for the associated cost and specialists needed to treat potentially millions of such patients.

Virtual reality can potentially be a lifesaver to the millions of stroke survivors due to its ability to deliver controlled, repetitive, and intensive therapy. Furthermore, VR-based or VR-augmented rehabilitation is very engaging and easily adaptable to individual patient's abilities. Finally, its use of computerized technology and multiplexing makes VR a force amplifier for the current army of therapists. It is thus not surprising that several research groups have started working on VR therapies for chronic post-stroke patients.

Holden and her colleagues at the Massachusetts Institute of Technology developed a system for motor training of arm reach motions of chronic post-stroke patients [Holden and Todorov, 2002]. Their approach, illustrated in Figure 8.14, is based on the principle of teaching by imitation. A "teacher" object performs a simple motion, sampled from the actions of an expert. In this case the motion is a frontal reach, something that stroke patients have difficulty with. The required motion trajectory is visualized by a sequence of wireframe-rendered cubes that passes through a virtual doughnut. This requires precision, similar to functional requirements associated with the task of reaching and placing an envelope through a mailbox slot. A second object (the white cube at the end of the virtual arm) attempts to imitate this trajectory under the patient's control. Another white cube visualizes the start of motion, and the real-time motion of the patient's arm is tracked using a Polhemus tracker. The system scores the patient's performance, taking into consideration both positional and temporal errors.

This simulation was integrated into a motor rehabilitation exercise depicting a virtual mailbox as well as virtual teacher and patient-controlled envelopes. The orientation of the mailbox slot (horizontal or vertical) as well as the placement of the slot in the virtual scene determine the configuration of the arm when extended (palm horizontal or vertical, near or far reach, etc.). These factors, together with the speed of the animation showing the teacher object trajectory, determine the exercise difficulty. Holden and Todorov [2002] and Holden [2002] reported on pilot trials of the system on nine patients. The patients were between 6 months and more than 5 years after the occurrence of their stroke. They underwent 30 rehabilitation sessions, each lasting for one hour, three times every week,

and their clinical measures were taken before and after treatment. Results showed significantly higher shoulder flexion and grip strength post VR therapy. The system is undergoing further trials and is being extended to include a telerehabilitation capability.

Researchers at Rutgers University, in collaboration with those at the University of Medicine and Dentistry of New Jersey and at the New Jersey Institute of Technology, are developing a multiplexed post-stroke telerehabilitation system of the hand [Jack et al., 2001; Boian et al., 2002]. The approach differs from that taken by Holden, in that rehabilitation is based on gamelike exercises, is done at the finger level, and incorporates haptic feedback. Using video game-like graphics is more engaging than the simple graphics used in the teaching-by-imitation method previously described. However, games could potentially pose a problem for stroke patients due to the increased cognitive load and richer visual feedback, which could confuse them.

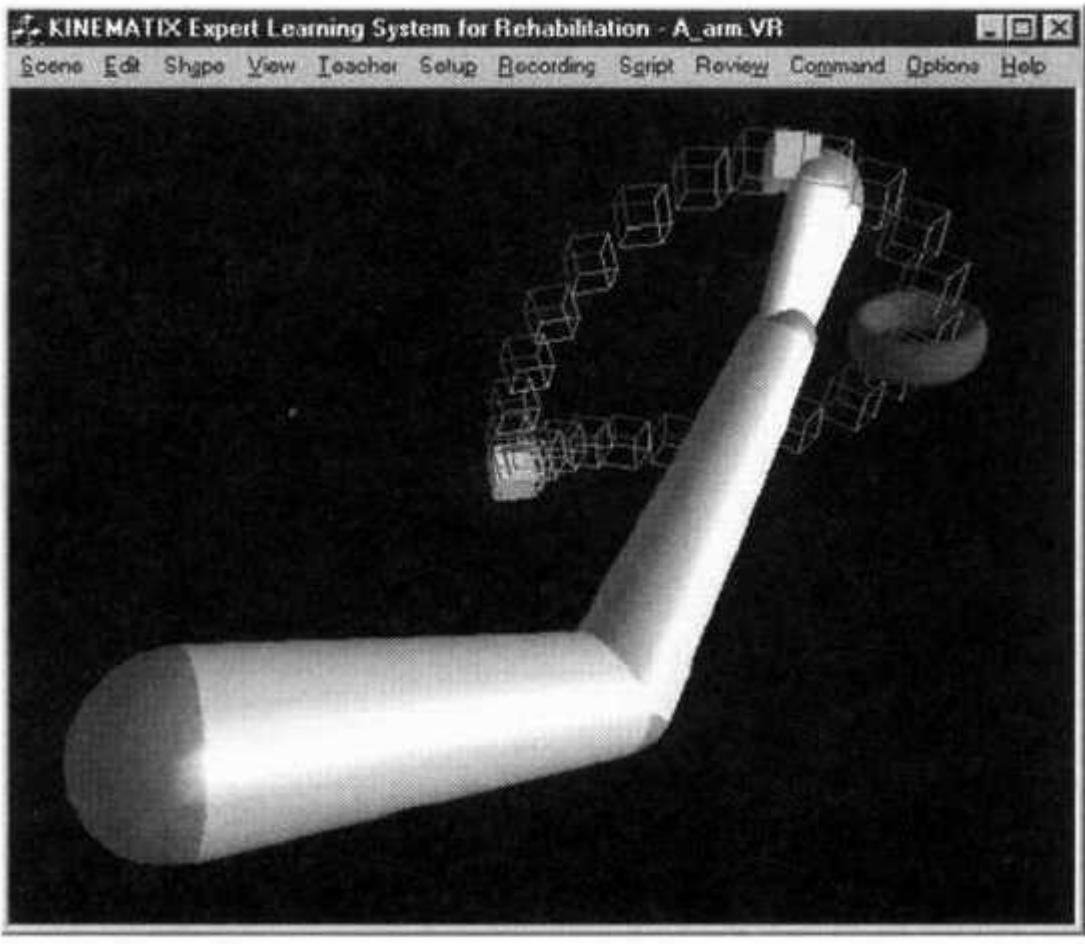
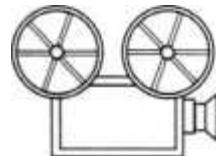


Fig. 8.14 Teaching an arm-reaching motion by imitation to chronic post-stroke patients. From Holden and Todorov [2002]. Reprinted by permission.

The hand telerehabilitation system developed at Rutgers University consists of a PC and two sensorized gloves (a CyberGlove and the Rutgers Master II). The CyberGlove is used to measure the finger range of motion, finger velocity, and independence of motion (fractionation). The force feedback produced by the Rutgers Master II is used in a grasp-strengthening exercise. The patient wearing one of the two gloves controls a virtual hand, as shown in Figure 8.15 [Boian et al., 2002].



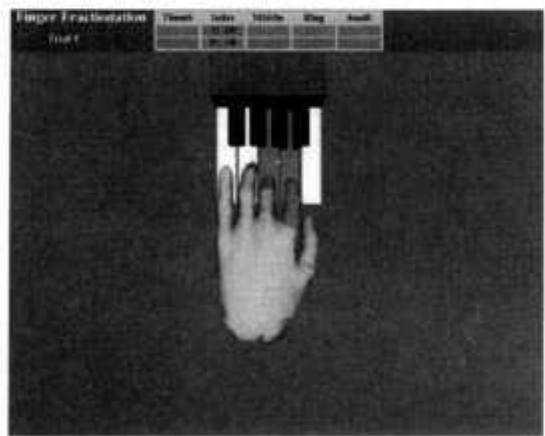
The first exercise (range of motion) asks the patient to "clean dirty pixels off the screen" such that a nice hidden image is revealed. The second exercise (speed of motion) consists of fast grasping in order to "scare away" a virtual butterfly. The third exercise is a piano keyboard designed to train in moving a single finger at a time. Finally, the strength exercise is a virtual rendering of the Rutgers Master II in which the patient has to "fill the pistons with color." This is necessary to provide the patient with visual feedback that replaces a lacking proprioception. The feedback force is kept constant such that the more the patient moves, the more mechanical work the patient's hand outputs. The patient's performance is stored transparently in an Oracle database locally and then uploaded to a server station. Therapists can then access this password-protected database over the Web and extract graphs showing a patient's performance history.



(a)



(b)

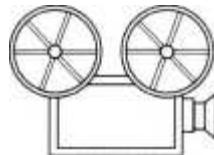


(c)



(d)

Fig. 8.15 Use of video game-like exercises for hand rehabilitation of chronic post-stroke patients: (a) range of motion; (b) speed of motion; (c) fractionation; (d) grasping strength. From Boian et al. [2002]. © Rutgers University. Reprinted by permission.



VC 8.5

The system is undergoing a series of pilot clinical trials to determine its efficacy as a new therapeutic approach. The first trials, done in summer 2000, used a VR-augmented rehabilitation method in which patients had to exercise daily for 3 hours during 2 weeks (on both real and virtual objects). The second round of trials, which took place 1 year later, was VR-based, with no conventional therapy being administered. During these trials patients performed 1.5 hours of VR exercises daily for 3 weeks. All patients improved significantly their hand motion and grasp strength. Interestingly, these improvements at the impairment level transferred to function. Patients kept their arm stationary during therapy and there was no arm-reaching motion. Nevertheless, comparative tests on a pick-and-place task done before and after the VR-based rehabilitation showed statistically significant improvement. The system is being tested in a multiplexed telerehabilitation configuration in which two patients exercise simultaneously while being monitored remotely. Further tests are also being done with the Rutgers Ankle to determine its efficacy in post-stroke rehabilitation.

8.1.4.3 Psychological and Cognitive Rehabilitation. This is used to improve the functioning of millions of patients. Psychological rehabilitation is prescribed to treat obsessive-compulsive disorders, eating disorders, posttraumatic stress disorders, and phobias [North et al., 2002].

Patients with phobias may be afraid of heights (acrophobia), public places (agoraphobia), confined places (claustrophobia), certain animals, or flying. The classic treatment for such patients is exposure therapy in which

the patient is given controlled amounts of phobia-inducing stimulus in order to desensitize him or her. Such treatment may be both expensive and risky and poses serious problems regarding the patient's confidentiality. For these type of patients VR-based rehabilitation has a number of clear advantages. Since it is done on a PC, it can take place in the doctor's office, not in public. The virtual phobic experience (such as a desert with spiders or snakes) is safe, and the costs involved are much smaller than those associated with real exposure.

Treatment of the fear of flying is typical of the advantages VR therapy brings. This phobia affects 25 million adult Americans, disrupting their life and work (if they have to travel), and costing the airline industry billions of dollars because such patients stop flying altogether. In classic exposure therapy, patients need to take actual flights on commercial airplanes, together with therapists, in order to gradually gain confidence. Clearly there is no privacy and the treatment is expensive, both in terms of tickets and therapist time. Researchers at Georgia Institute of Technology, in collaboration with Emory University School of Medicine, developed a VR simulator to treat fear of flying [Hodges et al., 1996]. The simulator uses a PC for rendering, an HMD to present the scene to the patient, and a special chair to produce vibrations. Such vibrations are associated with taxiing, landing, and flying in stormy weather. The simulation consists of an interior of a Boeing 737 aircraft modeled accurately and of window views to an airport and surrounding landscape (Figs. 8.16a and 8.16b). The simulations progress through a sequence of stages, illustrated by the state graph shown in Figure 8.16c. Taxiing is followed by takeoff, level flight, buzzing flight (aborted landing), and landing. The progression of events (either set by the therapist or self-selected by the patient) includes turbulent flight as well as landing in a storm, both inducing high levels of stress.

The first case study was done on this system in 1996, and involved a 42-year-old woman who avoided flying altogether. She underwent six therapy sessions, twice every week. During each session (lasting approximately 35-45 minutes) the subject was allowed to progress along the various flight scenarios at her own pace. Immediately following the last session the patient was able to take a cross-country flight together with her family.

Subsequently, the researchers conducted a controlled study using 45 subjects. These patients were divided in three equal groups, one undergoing standard exposure (SE) therapy, one undergoing VR exposure therapy (VRE), and one undergoing no therapy (waiting list, WL). Both SE and VRE patients underwent four sessions of anxiety management techniques (breathing, thinking, etc.). Subsequently, the SE patients went to an airport and were exposed to ticketing, the waiting area, and the sight of real airplanes. They then went on a parked airplane and imagined flying. The patients in the VRE experimental group underwent four sessions on the simulator, progressing from parked (virtual airplane) to taxiing, takeoff, flight, and landing (in good and bad weather) [Emory Health Sciences, 2000]. VRE was found to be as effective as SE, based on subjective evaluation questionnaires and reactions during a real graduation flight. A follow-up study was conducted after 1 year to ascertain retention of the gains patients made during therapy [Rothbaum et al., 2002]. Again it was found that there was no significant difference between SE and VRE when it came to retention. Thus the long-term gains made using the VR simulator were maintained after 12 months with no therapy. Almost all treated subjects (92% of VRE and 91% of SE subjects) had flown during the year following treatment.

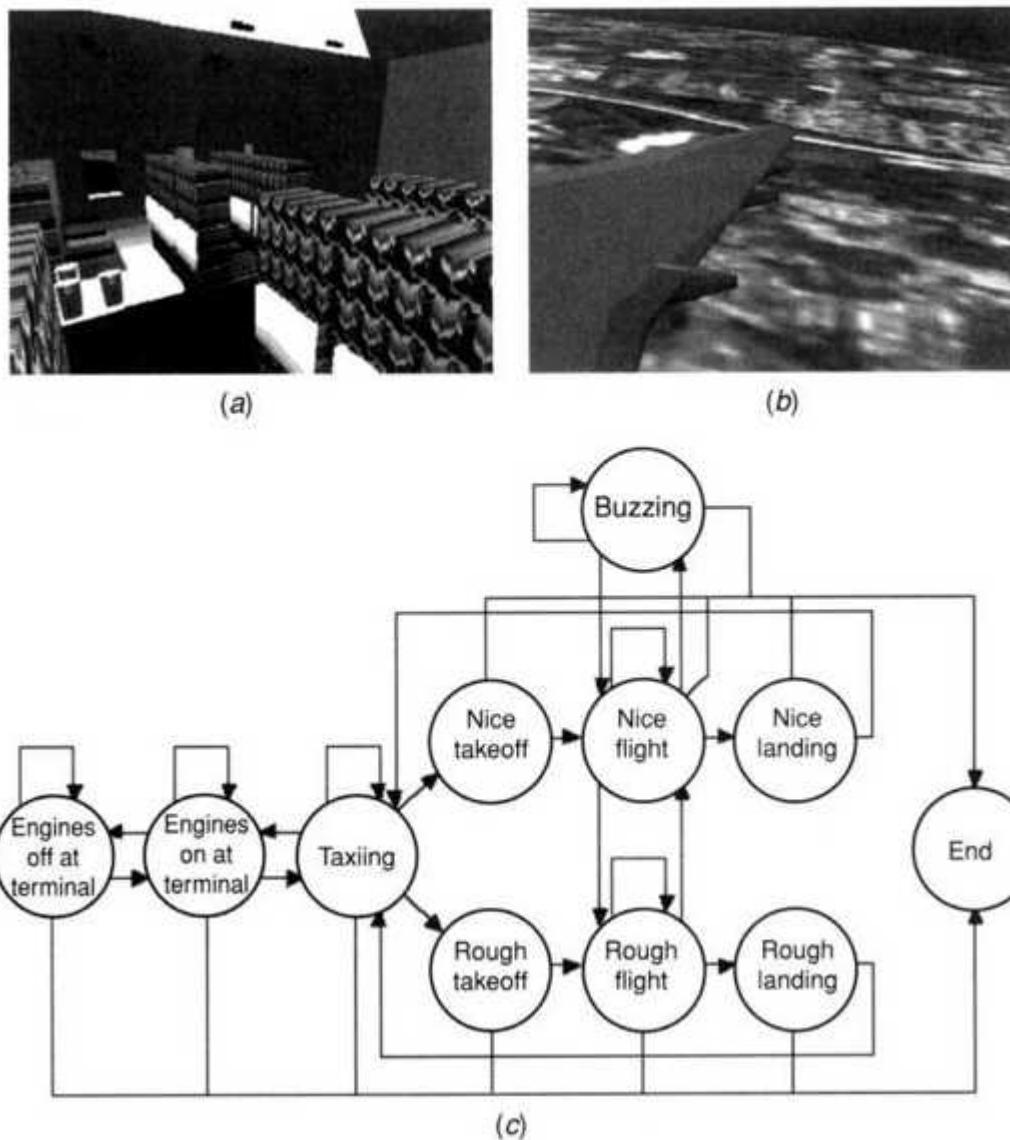


Fig. 8.16 Fear-of-flying VR-based rehabilitation: (a) airplane interior; (b) window view; (c) state machine for the flight simulator. From Hodges et al. [1996]. © 1996 IEEE. Reprinted by permission.

Cognitive deficits are associated with age (Alzheimer's disease and dementia), accidents producing traumatic brain injury, and stroke and also occur in children who exhibit attention deficit/hyperactivity disorders (ADHD) [Rizzo et al., 2002]. About 5% of American children of school age have ADHD. There is a lack of standardized and reliable methods for assessing (diagnosing) ADHD, and most assessment approaches involve indirect measure through questionnaires filled out by parents or teachers.

These questionnaires relate to the child's behavior over a 24-hour period (including the time spent in school).

Rizzo and his colleagues at the University of Southern California [Rizzo et al., 2000] developed a VR-based assessment approach to ADHD. This diagnostic method replicates a virtual classroom in which the subject (child) is immersed by way of an HMD. Trackers are used for the HMD as well as the nondominant hand and the opposite knee. These trackers are used to register the restlessness typically present in hyperactive children. Subjects sit at a regular school desk in order to increase the simulation realism. As illustrated in Figure 8.17a [Rizzo et al., 2001], the environment has a number of visual distractions (in this case a car that passes by the classroom window). Other distractions are 3D sounds, such as the sound of opening doors, moving chairs, whispers, etc. A virtual teacher (Fig. 8.17b) talks to the subject at the same time as letter characters or drawings appear on the virtual blackboard. The subject is then asked to push a mouse button whenever the teacher's verbal description does not match the drawing on the blackboard. Thus testing can replicate more realistically the school environment in which the child needs to function and also is able to measure the child's reaction and behavior directly.

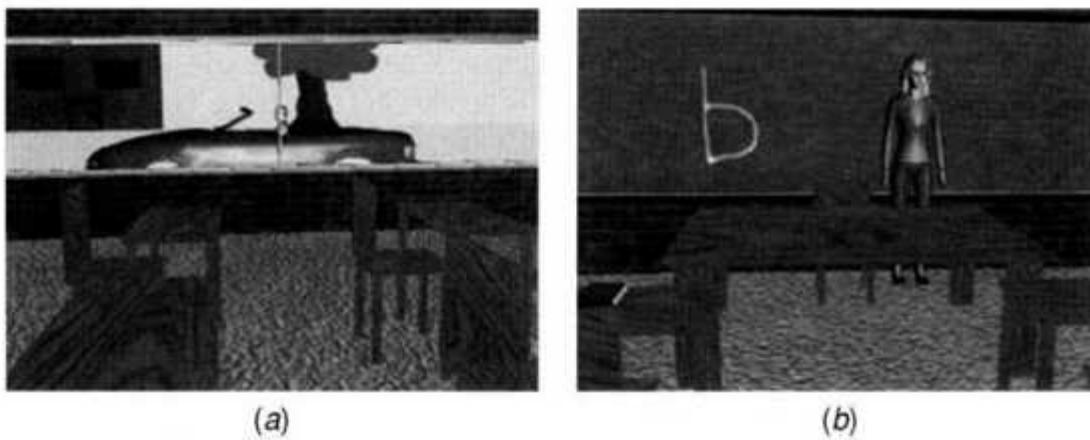


Fig. 8.17 Virtual classroom used in cognitive (ADHD) assessment: (a) distraction element; (b) virtual teacher. From Rizzo et al. [2001]. © 2001 Massachusetts Institute of Technology. Reprinted by permission.

The system underwent trials in order to gauge its sensitivity and ability to detect ADHD children. An experimental group of 10 diagnosed ADHD subjects and a control group of 10 normal subjects participated in the study [A. Rizzo, personal communication, 24 July 2002]. The study found that the motor movement (head, arm, leg) of the subjects in the experimental group was greater than that of the subjects in the control group. This motor movement was greater in the distracting conditions for the ADHD children than for those in the control group. A neural network algorithm trained on the first five experimental subjects was able to recognize accurately the hyperactivity exhibited by the remaining experimental subjects. Furthermore, ADHD children made more errors than the control group, whether distractions were present or not. Thus the initial data indicate the VR-based system could be used as a diagnostic tool in future ADHD assessment (and possibly rehabilitation).

8.2 EDUCATION, ARTS, AND ENTERTAINMENT

The foregoing cognitive assessment example showed how VR can serve as a tool to help diagnose ADHD children by recreating a classroom environment. At the same time VR is starting to be used as a way to convey knowledge, as a teaching tool for children and adults alike. In doing so VR takes advantage of its multisensorial interactivity, which is engaging and entertaining. Conversely, entertainment (whether using a virtual environment or not) can have an educational component, teaching arts, history, geography, zoology, and so on.

8.2.1 VR in Education

Educational theories classify learning activities as constructivist, constructionist, or situated [Moshel and Hughes, 2002]. Constructivist learning involves exploration of prebuilt worlds and discovery. During constructionist learning students must actively build models of the world as opposed to exploring already built ones. Situated learning is done through role playing, where students assume a certain character in order to better understand its way of life. Situated learning, more than any other method,

involves group interaction and development of social skills (an important educational component).

The three types of learning are well served by VR due to its ability to mediate world exploration and construction, its mapping of a user to any character he or she chooses, and the provision of shared virtual worlds. Through its interactive environment, repetition, and one-to-one experimentation, VR can help improve knowledge retention and student motivation. By simulating chemistry or physics experiments, knowledge can be transferred without increased safety risks associated with real laboratories. Through the Internet, VR can facilitate distance learning in places where schools or specialized teachers are scarce, and so on. In what follows we describe a few applications of VR in education, from the college level to the elementary school level.

8.2.1.1 Exploration-Based Learning. VR is well suited to convey difficult abstract concepts due to the visualization abilities. For example, at the college level students often have difficulty with differential algebra and geometry. Researchers at the Royal Institute of Technology in Sweden have developed CyberMath, a shared virtual environment that allows complex mathematical concepts to be presented in an exciting way [Taxen and Naeve, 2001]. The learning experience is organized as an exploratorium that the students, represented by avatars, visit under the guidance of a remote teacher. The exploratorium has a central virtual lecture hall and a number of interconnected exhibit areas. The distributed VR is built on top of DIVE (described in Chapter 4), using simple PCs for rendering. A Mathematica-to-DIVE converter is used to allow the modeling and animation of geometrical shapes. The exhibit rooms teach matrix transformations, generalized cylinders, cylindrical optics, and focal surfaces. Each of these rooms is rendered using radiosity for increased realism, as seen in Figure 8.18 [Taxen and Naeve, 2001]. Objects can be rotated or translated with the computer mouse. Virtual pushbuttons can be used to toggle rendering to wireframe mode or start animations. Some objects have associated URLs such that a Web browser can be used to get additional information.

The researchers reported on an initial human factors study with 13 college students from two Swedish universities. The students were all located in one room, while a professor was at a remote site. A course in differential geometry was given in the exploratorium, after which a subjective evaluation questionnaire was filled out by the subjects. They reported that CyberMath was successful in teaching complex geometrical shapes and that they felt captivated (immersed) by the exhibit, even though they had no HMDs or active glasses. Problems occurred when the avatar of one student occluded the view of another, an aspect solved by allowing users to make avatars transparent. The researchers are planning to measure long-term knowledge retention and add elementary mathematics and geometric algebra exhibits.

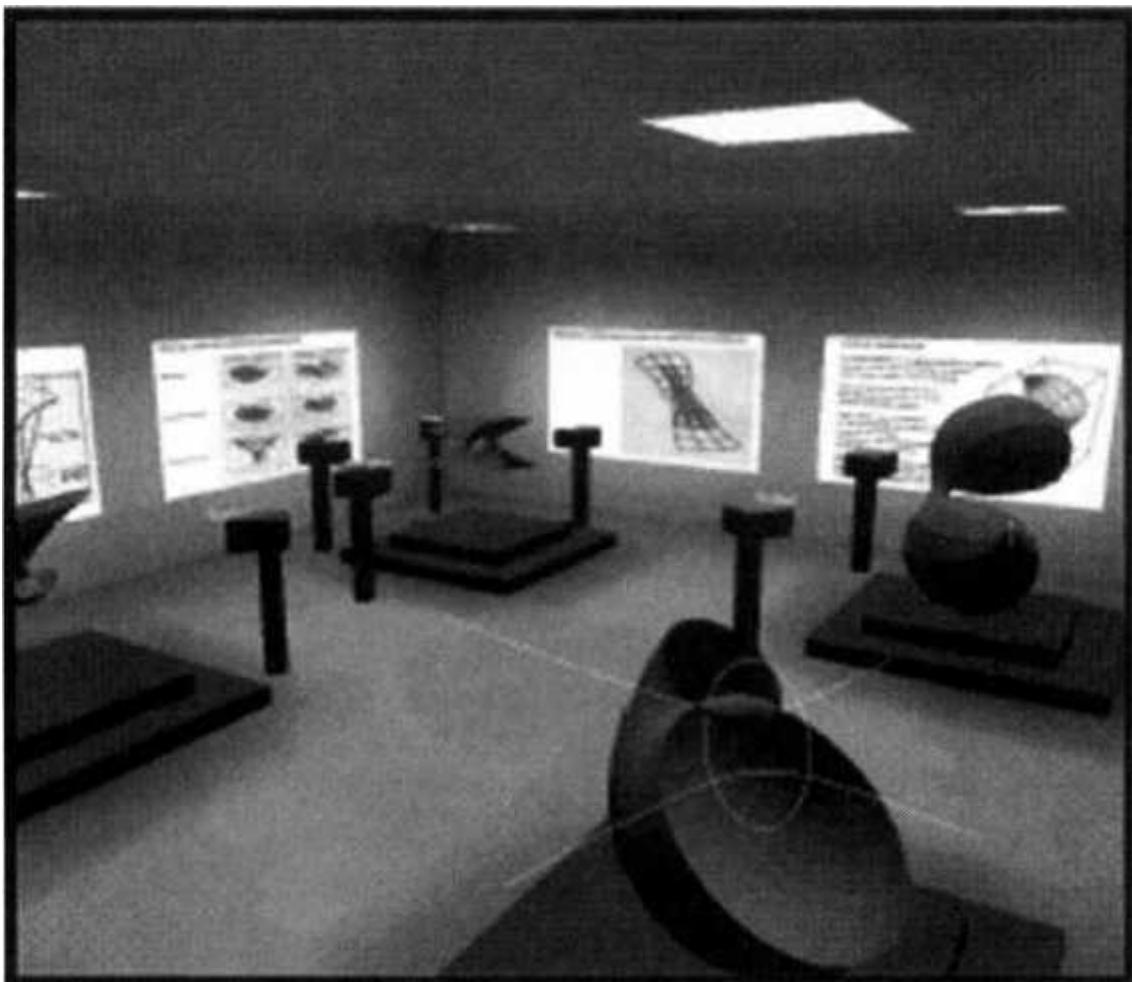


Fig. 8.18 CyberMath exhibit of focal surfaces. From Taxen and Naeve [2001]. Reprinted by permission.

Virtual reality is also useful in helping high school students understand complex concepts such as Newtonian or quantum physics. One of the earliest projects involving VR-based education was in fact the Virtual Physics Laboratory developed at the University of Houston [Loftin et al., 1993; Dede et al., 1999]. Many students have misconceptions about basic physical quantities such as position and speed, or force and displacement. The researchers hypothesized that giving students control of usually inaccessible environmental variables (such as gravity or time dilation) would help them acquire correct perceptions about basic physics. They developed an immersive VR system consisting of an EyePhone HMD, a VPL DataGlove, 3D sound, and two SGI 4D/320 VGX computers rendering the virtual scene. The graphics environment was cartoonish by today's standards (10,000 polygons scene complexity) in order to allow real-time rendering on the slow SGI workstations then available. The software environment used NASA's Solid System Modeler and Stereo Display Manager.

The laboratory experiments consisted of Newtonian motions (in the "Newton World") or electrostatic forces and fields (in the "Maxwell World"). The students could navigate through the virtual world using the sensing glove and gesture recognition. The glove was also used to interact with objects by grabbing them with a virtual hand. Figure 8.19a shows a user placing a charge in an electrical field. In the virtual laboratory students could shrink to the size of an electron or become one (obviously impossible in a real physics laboratory). They could thus ride the electron and better understand its motion in electric fields. The laboratory had a virtual control panel with switches, sliders, measurement devices, and pushbuttons (reset and freeze). The control panel could also be grabbed and moved in any part of the virtual laboratory room. Its buttons were activated by touching with the virtual hand, as shown in Figure 8.19b. In Newton World an experiment consisted in measuring the pendulum's oscillation period. In a real physics laboratory this is done by changing its length. This was also possible in the virtual laboratory. Additionally, it was possible to change the value of the gravitational acceleration, which cannot be done in a real laboratory. Atmospheric drag could also be reduced to zero (vacuum) or set to a predefined value.

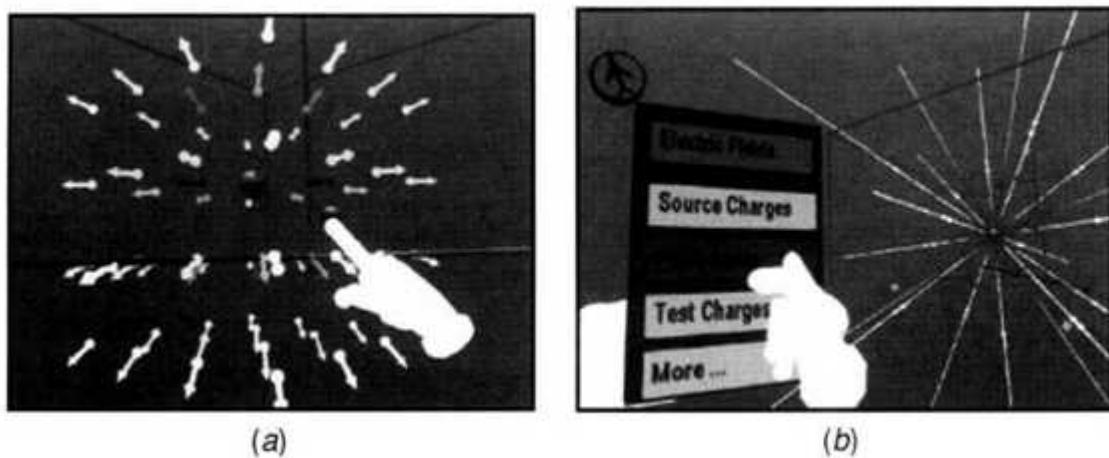


Fig. 8.19 Virtual physics laboratory: (a) student grasping an electrical charge in the Maxwell World; (b) menu selection using a virtual hand. From Dede et al. [2000]. Reprinted by permission.

Dede and his colleagues [2000] conducted a human factors study comparing learning on the Virtual Physics Laboratory with learning on traditional 2D multimedia educational software (EMF subjects). Students who used the Maxwell World simulation were better able to define concepts than EMF students. Furthermore, they were better able to demonstrate concepts in 3D than were EMF students. They felt more motivated to study by the VR simulation than the 2D multimedia software, despite the fact that they had greater simulation sickness symptoms and greater difficulty using the simulator. Since the Maxwell World subjects understood 3D concepts better, they had a better knowledge retention 5 months after being on the simulator than subjects who used the 2D instructional software.

8.2.1.2 Learning by Building Virtual Worlds. This approach was tested on elementary school children [Roussos et al., 1999]. The project, called Narrativebased, Immersive, Constructionist/Collaborative Environments (NICE), implements a persistent virtual garden. Children wear active glasses and are immersed in the environment displayed by a CAVE or an ImersaDesk. The groups of children elect a leader, who has his or her head tracked and interacts with the simulation through a tracked wand and through speech. The groups work collaboratively by telling the leader how

and where to seed plants, affect their growth by proper watering and positioning of the sun, and cull weeds. Each group sees the avatar representing the leader of the other group since the CAVE and ImersaDesk are networked (see Fig. 8.20). Besides learning about the life cycle of a garden, children can shrink and walk beneath the surface of the soil to look at plant roots. The plants are implemented by intelligent agents, which determine growth based on the amount of water, sun, and proximity to other plants. If plants are too crowded when seeded, they will not grow well. If too much sun is provided (children grab and move the sun), the plants put on sunglasses. If too much water is provided, they open an umbrella. These simple symbols of plant growth dynamics, together with audio communication with the remote group, help the learning process. Teachers and parents can also participate in the experience, and their avatars can guide the children and keep them focused on the task.

The scientists conducted a study with 52 second-grade children in order to determine whether there was an improvement in the subjects' knowledge of plants following VR-based learning. The subjects underwent an evaluation session lasting from 1 to 3 hours. Each session consisted of presimulation planning, the VR immersion, and postsimulation questions and discussions. The subjects were divided into teams of seven or eight children. Then the teams were divided into a CAVE group and an ImersaDesk group, each having a leader. Each leader was first instructed into the use of the wand and allowed to practice for 10 minutes. Subsequently, the two groups collaborated in planting and attending to the garden under the guidance of a teacher's avatar. After the simulation, the children were asked a number of questions about their VR experience, what they liked (or disliked) about the simulation, and what they thought they learned. The researchers observed that the leaders were more engaged into the simulation and the planting task, while the others were more distracted. Before the simulation only 12% of students (those ranking high in their class) understood gardening concepts. After the simulation this percentage went up to 35% (17 students). Of these the majority were children who were leaders in their simulation groups. Thus it seems the degree of interactivity had an effect on the cognitive gain. This is not surprising since this VR-based learning was a constructionist one. Some children

complained about the equipment they had to wear, which was not suited for their head sizes (active glasses are only manufactured in adult size). This shortcoming points to a major problem in implementing VR-based learning for elementary and middle school students. There is little VR equipment that fits these children. Another problem is the cost of such equipment, which is not negligible for meager school budgets. There is also the problem of supporting personnel knowledgeable about such equipment. Finally, many more studies are needed to assess the value of virtual reality as a teaching tool in both schools and universities. If these studies prove encouraging, then school and university curricula need to be revised accordingly. Eventually these new courses will have to pass the difficult test of board accreditation, without which VR cannot become part of mainstream education.



(a)



(b)

Fig. 8.20 Use of a virtual garden in a collaborative constructionist learning setting for elementary school children: (a) simulation scene with a child's avatar; (b) a group leader interacting with the simulation using a wand. From Roussos et al. [1999]. © 1999 Massachusetts Institute of Technology. Reprinted by permission.

8.2.2 VR and the Arts

The arts represents another area where VR is starting to play an important role. Virtual reality is a new medium of expression for conveying artists' messages [Robinett and Naimark, 1992]. Additionally, VR immersion and interactivity can transform static art (such as paintings or sculptures) into dynamic art which viewers can explore. This is particularly useful for art historians trying to decipher mysteries related to works of art. Finally, VR increases access not only to art creation, but also to its consumption. For example, physically disadvantaged people or people in remote locations can, using the Internet, visit virtual museums in the comfort of their homes.

8.2.2.1 The Virtual Florentine Pieta. This work was created by researchers at IBM in cooperation with Dr. Jack Wasserman, an art historian from Temple University [Bernardini et al., 2002]. Michelangelo's sculpture, dating to the mid 16th century, was originally intended for the artist's tomb. The 2.5-m-tall ensemble, shown in Figure 8.21a, depicts Christ resting across the lap of the Virgin Mary, being supported on the left by Mary Magdalene and at the back by a character whose face is believed to be that of Michelangelo. After sculpting the four characters out of a single block of marble, the artist abandoned his work, and broke away three arms and a portion of a leg. Later, one of Michelangelo's students reattached the three arms and finished part of Mary Magdalene's face. Currently it is on display at the Museo dell'Opera dell Domo in Florence, not at Michelangelo's tomb as originally intended.

The 3D model of the Florentine Pieta was created in a lengthy process that took 2 weeks to complete. The researchers first used a stereo camera-based scanner and digital camera/light assembly to acquire 4800 striped pictures and 4800 coregistered color images. Subsequently the overlapping 20 cm x 20 cm tiles of the sculpture's exterior were assembled using an array of laser dot landmarks. The 3D scanner dot clouds were transformed into triangle meshes using commercial software (Virtuoso Developer) and the digital images were used to add color. It was thus possible to obtain a 3D model of Michelangelo's masterpiece rendered at millimeter accuracy, with the stunning detail shown in Figure 8.21b. Since the virtual model was

intended for art historians, it was important to add interactivity without compromising detail. Thus the IBM team developed a 3D viewer that had a low-resolution version of the statue in order to assure high frame refresh rates on a laptop computer. When a detailed view of a certain area is desired it can be loaded by clicking on the area of interest. The viewer allows the statue to be seen from angles not possible in a museum, such as the top view shown in Figure 8.21c. Art historians regularly use flashlights to move repeatedly over a sculpture's surfaces in order to reveal intricate details that may have been initially missed. This functionality is recreated by the 3D viewer, which gives the user control over virtual light sources. The researchers were also able to do virtual restoration of the statue by attaching the missing portion of a leg. Finally, it was possible to put the 3D model in a garden with a virtual mausoleum, shown in Figure 8.21d. This way the sculpture can be admired in its originally intended surroundings.

8.2.2.2 Virtual Heritage. Researchers have created virtual replicas of famous archeological, architectural, and natural sites that constitute the world's heritage. This heritage has been protected since 1972 by the United Nations Educational, Scientific and Cultural Organization (UNESCO). In 2000 the Virtual Heritage Network was formed as a UNESCO affiliate in order to help the preservation of heritage sites in VR. It maintains a very informative Web site showing current projects, conferences, publications, and related links [Virtual Heritage Network, 2002]. Virtual models have been created for Stonehenge in England, the Frauenkirche in Germany, and the Terracotta Warriors in China, among others.



(a)



(b)



(c)



(d)

Fig. 8.21 Michelangelo's Florentine Pieta: (a) photograph of the real statue; (b) 3D model; (c) top view of the synthetic statue; (d) virtual sculpture in a mausoleum. From Bernardini et al. [2002]. © 2002 IEEE. Reprinted by permission.

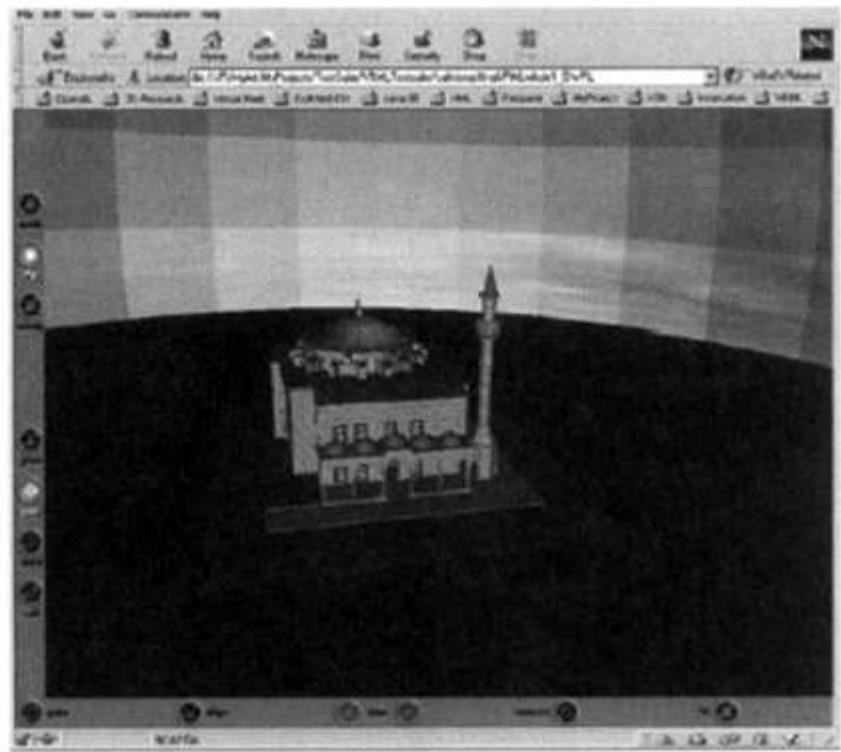
Creating a heritage model is both a technical and a political endeavor. Clearly cultural sensitivities exist, a problem exacerbated by the placement of such models for worldwide viewing on the Net [Delaney, 2000]. Thus developers have to be concerned not only with the technical aspects (which are covered in this book), but also with the related cultural and historical sensitivities surrounding a particular heritage.

Researchers at the University of Geneva undertook the virtualization of the Church of SS. Sergius and Bachhus in Istanbul, Turkey [Papagiannakis et al., 2001]. This old Byzantine church has the same architecture as the much better known Saint Sophia Cathedral and was similarly converted into a mosque. The creation of a virtual model started with architectural plans, historical documents, and visits to the actual site. Photographs were taken to document architectural details not present in the blueprints and to create material textures. Light measurements were made to serve as input to the radiosity lighting modeling of the edifice. Subsequently two 3D models of the church were created using 3D Studio Max. One model (18,000 polygons) represented the exterior, another the interior (59,000 polygons). The two models were then linked by a portal in order to improve rendering speed (only one model was rendered at a time). Each model was subsequently textured and exported to VRML97 in order to be viewed on the Web. The resulting scene, shown in Figure 8.22a, is rendered at over 30 frames/sec. It is geometrically accurate to about 2 cm, but initially, lacked realism. To improve realism the researchers created radiosity-based light maps. The radiosity calculation used material properties for the walls as well as the previously acquired illumination data. The light maps were created offline and then overlaid on the textured model during real-time rendering (as discussed in Chapter 5). Viewed with a custom extension of VRML'97 (to allow multitexturing), the model is rendered at 10-26 frames/sec. This reduction in frame refresh rate is compensated by increased realism, as seen in Figure 8.22b.

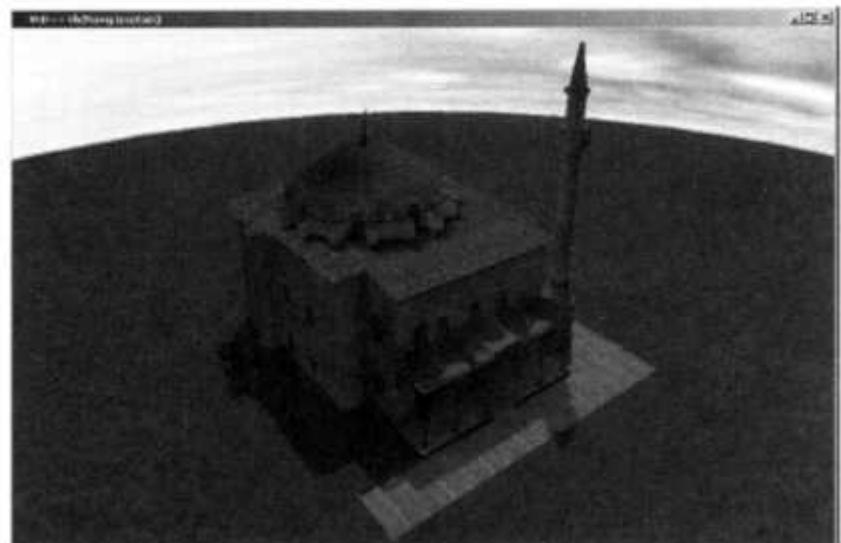
Another example of construction of a synthetic heritage site is Virtual Notre Dame [DeLeon and Berry, 2000]. This is the work of an international team of programmers and researchers from Digitalo Design Co. (Sunrise, FL), Gifu University (Japan), and the Centre de Recherches sur les

Monuments Historiques (Research Center for Historical Monuments, Paris). Their aim was to create a worldwide-accessible shared virtual environment allowing virtual tours of the famous Notre Dame Cathedral. Planned in the 12th century, the monumental architecture is so large and complex that it took 200 years to complete. Its beauty and rich history make Notre Dame one of the most respected religious and cultural symbols of Europe. To reconstruct such an edifice in VR is clearly not a trivial endeavor. The team's approach was initially similar to that described for the reconstruction of SS. Sergius and Bachhus in Istanbul. Thus they started by accumulating architectural blueprints and historical archives and taking digital photographs of the construction. The digital photographs and hand sketches were used to create a database of construction elements to help in the modeling process. These structures were modeled by hand (not with a scanner) and textures were added to reduce polygon count. Unlike the group in Switzerland, however, the rendering of the Notre Dame Cathedral was done on a proprietary engine (Epic Unreal), normally used in creating video games. This allowed 24-bit texturing, C++ programming, easy extension, and high-speed rendering of a complex model. The Epic Unreal graphical user interface has a point-and-click editor that allowed programmers to build and preview the model in real time. The result is a virtual cathedral of astonishing detail, as shown in Figure 8.23a.

Another difference is the addition on a virtual tour guide, represented by the friar shown in Figure 8.23b. He was modeled with Maya (Alias Wavefront) and then exported in the custom 3D environment. The tour guide character is a segmented model with about 1200 polygons, textured and animated through predetermined motion sequences. These motion sequences are built into a simple AI behavior engine that triggers them upon input from proximity sensors (as discussed in Chapter 6). Thus when the user (visitor) is close to an area of special interest, the virtual tour guide points and offers explanations (through text).



(a)



(b)

Fig. 8.22 The Church of SS. Sergius and Bachhus in Istanbul, Turkey: (a) 3D model rendered in VRML'97; (b) improved realism through radiosity-based light maps. From Papagiannakis et al. [2001]. © 2001 IEEE. Reprinted by permission.



(a)



(b)

Fig. 8.23 Notre Dame Cathedral in Paris: (a) interior view of the 3D model; (b) virtual tour guide. From DeLeon and Berry [2000]. © 2000 IEEE. Reprinted by permission.

8.2.3 Entertainment Applications of VR

Entertainment was the driving force of early VR technology and still is a substantial market. Rich sensorial interaction and 3D immersion make VR an ideal entertainment environment. The lucrative video game and theme park markets are the areas of fastest growth, with the financial and creative involvement of such heavyweights as Sega, Disney, Paramount Pictures, LucasArts, General Electric, Hughes, and Microsoft. VR entertainment applications now range from PC-based games, to networked multiplayer games, to location-based VR rides.

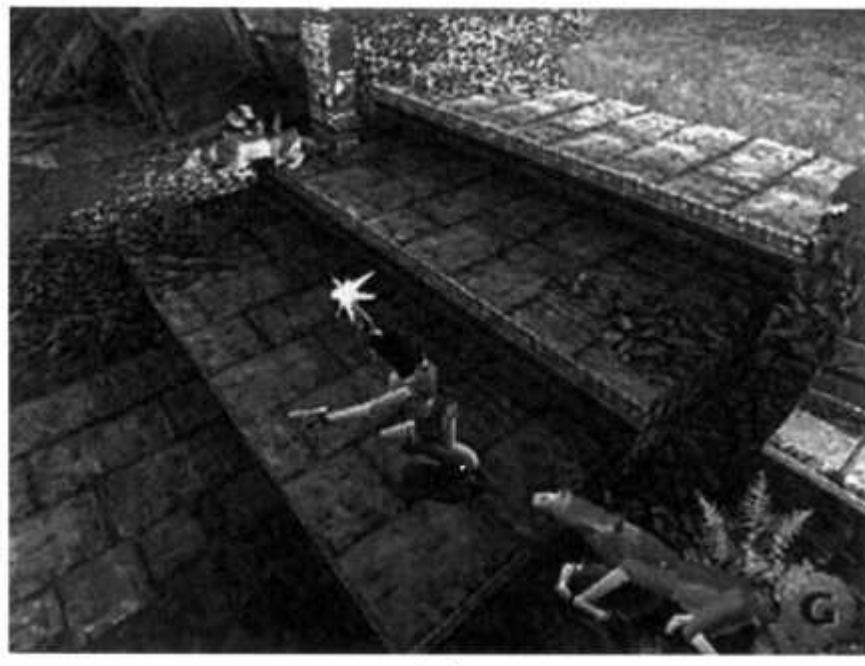
8.2.3.1 PC Video Games. These are lower end, mostly single-user, partly immersive VR simulations. Earlier attempts at HMD-based inexpensive simulations [Rich, 1993] failed due mainly to legal concerns related to simulation sickness. As a consequence, current PC games use monoscopic graphics and generally rely on standard keyboard or mouse-type interfaces. What these games lack in I/O sophistication is more than compensated by increasingly realistic graphics due to the proliferation of powerful rendering hardware available on the standard PC. Another important dimension of successful video games is sensorial overload, which keeps the player captive, and AI synthetic characters with realistic behavior [Badique et al., 2002].

Current PC-based games can be classified according to their story or plot. The vast majority are adventures, such as the Lara Croft Tomb Raider illustrated in Figure 8.24a. In adventure games the player shoots anything in sight while trying to avoid being shot by computer-generated opponents. These games are designed for the very large audience of male adolescents, which buy them by the millions. As a consequence, violence (and addiction) is high, while intellectual demand is low. At the other end of the spectrum are strategy games, such as StarCraft, shown in Figure 8.24b. Winning these games requires planning and management of finite resources, with the player acting more like a director or general in command of armies. The player is presented with a 3D closeup of a portion of the play space, while action is taking place on a much larger scale (represented by a map). In the example shown here the player commands "Terran" troops that have landed on a desert planet and construct defensive structures to repel an imminent alien attack. Actions are character-specific (a soldier shoots, a

transport ship loads/unloads, a medic heals, etc.) and are selectable from menus at the bottom of the screen. Proximity sensors trigger reflex behavior such as a sentry shooting when enemy forces penetrate the perimeter, etc. In general the player has to manage complex situations (including artificial intelligence, AI, aliens that evolve and get stronger in time) and the resulting cognitive load is high. As a consequence, the players of strategy games tend to be older than those having fun with adventure-type VR games.

Other types of PC games are sports (flying, driving), role-playing games, and online games. The proliferation of the Internet proved an incredible magnet for setting up large servers (such as Sony's Everquest, or www.kali.net), which allow remote players to log in and compete against each other. A particular type of online game that may proliferate in the future is gambling. These virtual casinos could prove extremely lucrative, but also extremely controversial, in terms of addiction and access by minors.

8.2.3.2 Location-Based Entertainment. This is the alternative, and historically the predecessor, to home-based PC VR gaming. In the early 1990s several VR arcades were formed in order to capitalize on the novel technology at a time when PC hardware was lagging. The situation is very different now, posing severe competition to traditional VR location-based entertainment. In order to survive and make a profit, modern VR arcades need to offer experiences not possible at home. As a consequence they tend to be bulky simulators, accommodating family-size teams immersed in a sensorially rich adventure. With large budgets and floor space, they provide motion platform-type haptics, wind, and even water effects that are simply impossible to do in the home.



(a)



(b)

Fig. 8.24 PC 3D interactive games: (a) action/adventure type (Lara Croft's Tomb Raider. Courtesy of Eidos Interactive Inc.); (b) strategy/command type (StarCraft®). Courtesy of Blizzard Entertainment®. Reprinted by permission.

One company that is famous throughout the world for its theme parks is Disney. In 1998 it launched Disney Quest in Orlando, Florida, a mall-size structure full of simulators. Two of its attractions merit special attention in view of their VR-rich content. One is the Pirates of the Caribbean: Battle for Buccaneer Gold. Teams of up to seven players sail a ship that is supported by a motion platform. The crew needs to cooperate to win, and every player is part of the experience, which lasts for about 4 minutes. Six of the players operate realistic cannons with ropes that trigger dynamically correct virtual projectiles in order to fight off pirate ships. The captain of the ship has a steering wheel and throttle, which are read by the computer. These inputs are used to change the viewpoint to the simulation interactively, branching on several possible scenarios (island, pillaged village, phantom ship, etc.). The virtual scene is rendered in stereo by an SGI Reality Engine and projected on a large (200°) screen [Fisher, 1999].

Another VR-based game at Disney Quest is the Virtual Jungle Cruise. The theme is a ride down the rapids of a prehistoric river, complete with caves, volcanoes, and dinosaurs. Up to four players board a real raft that is supported by inflatable air bags. These serve as a motion platform to replicate the haptic effects present during white water rafting. Each player has an oar retrofitted with joint bearings at the end. These make contact with a sensing textile lining the sides of the raft, such that the computer running the simulation knows who is rowing and how intensely. This input is used to change the direction of the virtual raft as well as its speed. The scene is rendered by an SGI Infinite Reality at 30 frames/sec [Griffith, 1999]. The players face a large screen showing the choppy river waters, which are synchronized with the up/down and side-to-side motion of the raft. To increase simulation realism even further, water sprays are intermittently projected on the players during the ride.

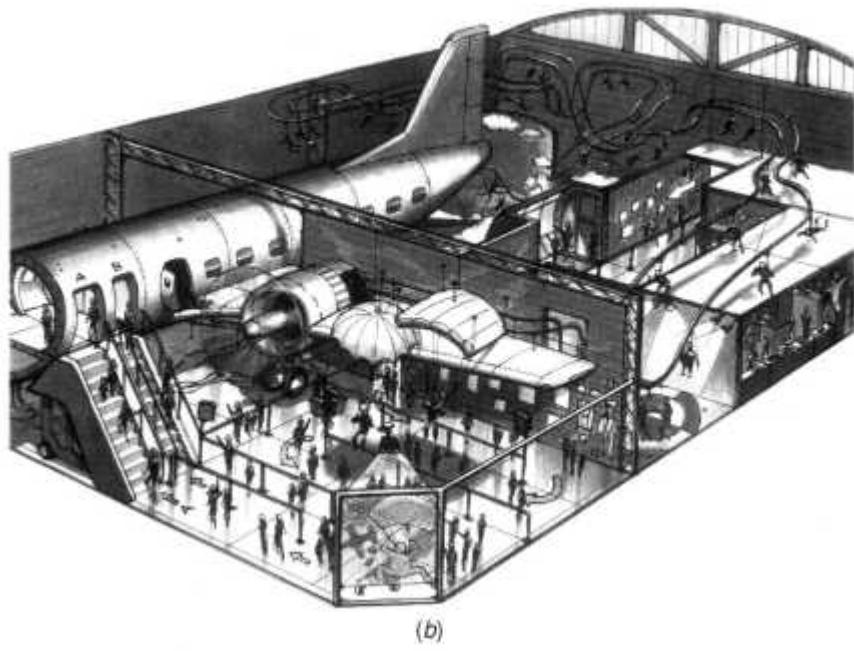
A newer location-based VR entertainment system is the JumpZone! standalone simulator, shown in Figure 8.25a [Brill, 2001]. It consists of an airplane interior mockup with a harness support and a trap door. The player dons a regular parachute harness and an HMD prior to the simulation. At the start of the ride the trap door opens and the player falls down a small distance. However, the rich visual feedback as well as wind provided by a

computer-controlled fan fool the brain into perceiving a much longer fall. Once the player pulls on the parachute harness, a virtual parachute opens and the player gets a jolt from the ceiling support. The HMD is tracked such that the aerial view of the landing field and surroundings move naturally. The pulling of the harness navigation ropes is another input to the simulation used to change direction. The user can thus descend for about 3 minutes and land in a forest, in a city, or on an aircraft carrier. The latter example underscores the duality of gaming hardware, which is easily adaptable to military use (by simply changing the software).

Prior to its recent closing Illusion Systems Inc. was developing a much larger JumpZone! Dark Ride intended for theme parks. Its concept drawing is shown in Figure 8.25b. The ride starts inside an actual airplane hull retrofitted with a ceiling chain drive. Here players don their parachute harness and helmet-attached HMD and progress to the tail of the airplane. This progression is synchronized with audio zones that replicate takeoff and level flight sounds followed by the pilot's voice telling players to get ready. Once the signal is given, two players at a time drop about 20 ft in free fall, after which they travel supported by a dual chain drive at speeds of about 2 ft/sec. Wind and heat effects are added in order to increase the simulation realism. The dual track has a total of 130 seats, and the actual parachute simulation lasts about 3 minutes (as in the standalone model). As opposed to the standalone simulator, which has a throughput of about 20 players/hour (and costs \$45,000), the larger Dark Ride model can accommodate up to 2400 players/hour (at a cost exceeding \$8 million). Due to the parent company disappearance, the future of the Dark Ride is unclear at the time of this writing.



(a)



(b)

Fig. 8.25 The JumpZone! parachute simulator: (a) single-seat version; (b) theme park version. From Brill [2001].

8.3 MILITARY VR APPLICATIONS

The military has long understood the importance of simulation and training under the doctrine "we train as we fight and we fight as we train." The current trend toward increased technological complexity and shorter military hardware lifespan requires simulators that are flexible, upgradeable, and less expensive. After all, if a simulator is designed only for a given tank model or aircraft model, it will get obsolete when those weapons do. Another trend in modern military training is networking. This allows remote simulation without having to transport trainees to the simulator site. Networking is also needed in team simulations, which are more realistic than singleuser ones. Virtual reality is networkable, flexible, and easily upgradeable, therefore it ideally matches the needs of military simulation outlined here.

Modern armies are called upon to perform a variety of missions, from military campaigns to peacekeeping operations and disaster relief, and to do this in many parts of the world. Training thus needs to be more flexible than ever before, allowing mission simulation ahead of execution and advanced debriefing after completion. VR matches this need for mission profile flexibility while allowing training at various levels of military hierarchy (from dismounted soldier to platoon, battalion, and so on). All military branches use some form of VR simulators, which rely increasingly on AI to control both friendly and enemy "automated" forces [Knerr et al., 2002]. The discussion here is structured along the various military branches (the Army, the Navy, and the Air Force). Owing to space limitations, we mention only a few of the multitude of existing VR defense applications.

8.3.1 Army Use of VR

The U.S. Army was an early adopter of VR-based training in the form of simulators for large-scale motorized units. This was appropriate for the envisioned large tank battles of the Cold War era (which fortunately never happened in reality). The current trend for small-scale conflicts (such as the so-called war on terrorism) emphasizes, however, the need to better train individual soldiers. Thus our discussion starts with singlesoldier simulators, followed by platoon-level leadership training. We then describe

the SIMNET company-size tank simulator and its follow-up, the Close Combat Tactical Trainer (which adds aviation units attached to the Army).

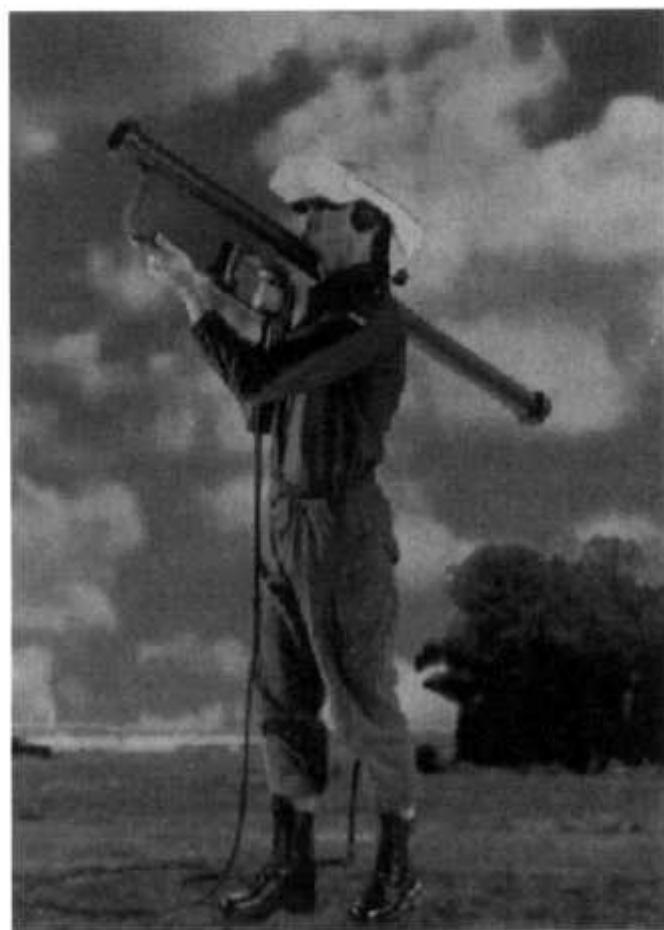
8.3.1.1 Single-Soldier Simulators. These are made possible by the dramatic reduction in computing hardware cost and substantial increase in its performance (as discussed in Chapter 4). Such simulators are useful not only for improving marksmanship, but also for better training in operating expensive weaponry. An example is the Virtual Stinger Trainer, shown in Figure 8.26a, which was developed in Holland by TNO Physics and Electronics Laboratory [H. Jense, TNO Physics and Electronics Laboratory, The Hague, The Netherlands, personal communication, August 1993; Division Ltd., 1993].

The Stinger is a compact shoulder-fired missile designed as a defense against low-flying aircraft and is in use in many armies around the world. The standard Stinger Trainer used by the military consists of a 20-m-diameter projection dome. The background landscape is projected by a single projector with a fish-eye lens mounted on top of the dome. Two moving video projectors mounted on mechanical arms project two independent images of attacking aircraft. Instructors determine the attack scenario and track the progress of the trainees using a workstation. The virtual simulation developed by TNO uses an HMD, interactive sound, and 3D tracker integrated with a workstation running the simulation. A full-size plastic mock-up of the Stinger was constructed in order to give the trainee a better tactile feel of the simulation. The 3D tracker is integrated inside the mock-up support handle so that it can track the position of the missile launcher. Pushbuttons are used to replicate the weapon's switches (safety, uncaging, and firing triggers). The software consists of a terrain model projected on the HMD, a model of the Stinger, and models of the attacking aircraft (including trajectory). The virtual Stinger model simulates only the essential weapon parts (launch tube, separable grip lock, battery/coolant unit, sight assembly, and missile). Attacking aircraft models are ported from the existing (domed) simulator system. Interactive sound is used to give the trainee an audio cue for target acquisition, in a similar fashion to the real weapon. The trainee hit/miss ratio together with any errors in the proper

firing sequence are automatically stored by the system and reported to the instructor.

The work done at TNO was the basis for a Stinger Trainer which entered service in the German army in 1999 [Reichert, 2000]. This transportable simulator is housed in a container such that the trainee's motion is tracked optically (to avoid magnetic interference). A high-resolution CRT-based HMD replaced the earlier HMD used in the TNO system in order to improve target spotting. Finally, the German version of the Stinger Trainer has a commander working together with the trainee (gunner) and each is represented in the virtual scene by avatars. The commander and gunner hold on to orientation rings, which are part of the container, since the HMD occludes their (real) surroundings.

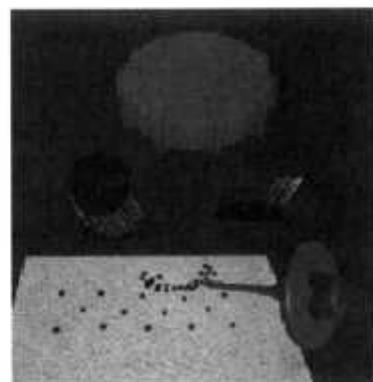
Another use of single-soldier VR simulators is in training for difficult missions, such as mine detection and defusing. The consequence of failure in such missions is often fatal, while failure in detecting a virtual mine is not. The French Defense Department has developed such a training system using a PHANTOM interface to provide haptic feedback [Stone, 2000; Pasquier, 2000]. As shown in Figure 8.26b, the trainee holds a standard-issue military probe attached to the PHANTOM and wears a tracked HMD. The trainee sees a virtual terrain and probe as well as probe penetration pattern (Figure 8.26c). When a virtual mine is located the probe suddenly increases the feedback force. The next step for the trainee is to determine the type of mine by continuing to probe around the initial contact location. A pattern recognition process monitors the trainee's actions in order to match the contact pattern against stored models of mines. Once a match is found, the corresponding mine 3D model is displayed in the virtual scene.



(a)



(b)



(c)

Fig. 8.26 Single-soldier simulators. (a) The Virtual Stinger Trainer. From Division Ltd. [1993]. (b) De-mining training system. (c) Virtual mines and probing pattern. From Stone [2000] and Pasquier [2000]. © 2000 Springer-Verlag GmbH. Reprinted by permission.

8.3.1.2 Platoon Leadership Training. This training is required in order to improve decision making for young lieutenants commanding these small units. Increasingly they are faced with the need to take difficult decisions under pressure, sometimes in circumstances not mentioned in training manuals and in unfamiliar surroundings. A case in point is peacekeeping operations, where the media can turn any error into an international incident. Having understood that traditional training is not sufficient, the U.S. Army created in 1999 a joint venture with the University of Southern California called the Institute for Creative Technologies. It was tasked to create realistic, highly emotional training Mission Rehearsal Exercises (MREs) developed by teams of artists and computer scientists.

One such MRE, illustrated in Figure 8.27 [Lindheim and Swartout, 2001], simulates a peacekeeping incident in Bosnia. The scenario is that of an accident between a military jeep responding to an emergency and a civilian car. The accident results in an injured young child, and the trainee needs to decide on the course of action (evacuate the child by helicopter, fractionate the unit by dispatching some squads to a nearby town uprising while leaving others to care for the child, and so on). The only real character is the trainee, who sits in front of a large wall display and interacts with the simulation through voice commands. The other 50 or so characters are modeled with PeopleShop (described in Chapter 6). Each character's behavior is controlled either by a script (for simple actions) or by AI, for characters that have complex behavior. An example of such a character is the platoon sergeant (shown facing the trainee), who coaches him on the best course of action. AI-controlled characters have text-to-speech voice generation, while those executing scripts have prerecorded (fixed) dialog. A special character is the victim's mother, who exhibits emotions. She gets progressively more upset if the trainee sends his men away since she fears her son will be abandoned. Her voice is also that of a prerecorded actress since text-to-speech conversion is unable to convey the emotional charge of the situation. Scene realism is supplemented by sound produced through a 10-channel audio feedback system.

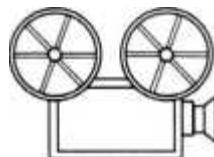


Fig. 8.27 Platoon leadership training for a peacekeeping operation. From Lindheim and Swartout [2001]. © 2001 IEEE. Reprinted by permission.

8.3.1.3 Company and Battalion-Level Simulators. These are used to train armored units and their artillery and close air support. In the early 1980s the Defense Advanced Research Projects Agency (DARPA) began the development of the first realistic virtual battlefield in order to train tank crews in joint exercises. The primary motivation for this endeavor was a reduction in training costs, but considerations also included increased safety and reduced environmental impact (explosions and tank tracks do great damage to the training area). The project, called Simulation Network (SIMNET), linked over 200 tank simulators in the United States and Germany [McDonough, 1993]. Each SIMNET simulator replicates the interior of an M 1 battle tank, complete with navigation, weapons, sensors, and displays. The vehicle weapons, sensors, and engine are modeled dynamically by the resident computer, which keeps a database copy of the whole virtual battlefield (the initial battlefield was a 50 x 75 km area in Central Europe, later extended to the Kuwait Theater of Operations). All databases represented the terrain relief accurately, including foliage, roads, buildings, bridges, etc. Communication between the members of a tank crew was done through intercoms, while communication with other tank simulators was done by voice and electronic messages over a WAN. The

distributed and real-time nature of the SIMNET required a minimization of network traffic. Additionally, the communication protocol had to function even if the network stopped transmitting for short periods of time or had large transmission delays. The solution adopted was that of dead reckoning, in which each tank simulator ran a model of all the tank crews on the network based on their latest transmitted message. Therefore messages were sent only when a significant state change occurred, and network traffic was minimized.

After mastering the problems of communication delays, designers realized that a tank-only battlefield was not a realistic one. In a real confrontation, tank commanders can count on support troops such as rearm vehicles, field artillery, or helicopters, and this needed to be replicated in VR. Designers also realized that it was not practical (due to high cost and complexity) to keep building various other types of simulators for these support troops. Instead instructors would sit at Management, Command and Control (MCC) workstations, where they would command virtual support forces. Upon receiving a radio call from tank commanders, the instructor would type on the MCC workstation, and support forces would appear in the virtual battlefield. By the late 1980s sufficient experience was obtained in the use of SIMNET to determine that the battle scenario had to be changed to show enemy superiority in numbers. The trainees wanted to practice against enemy forces three times their size, which required a very large number of simulators. Since this was too great a drain on resources, the designers decided to create Semi-Automated Forces (SAFORs). These were virtual units of tanks and helicopters that acted as intelligent agents. An instructor could issue orders through a SAFOR console and the enemy forces would move in formation and engage the trainees according to preprogrammed battle drills. The resulting virtual battlefield is illustrated in Figure 8.28.



VC 8.6

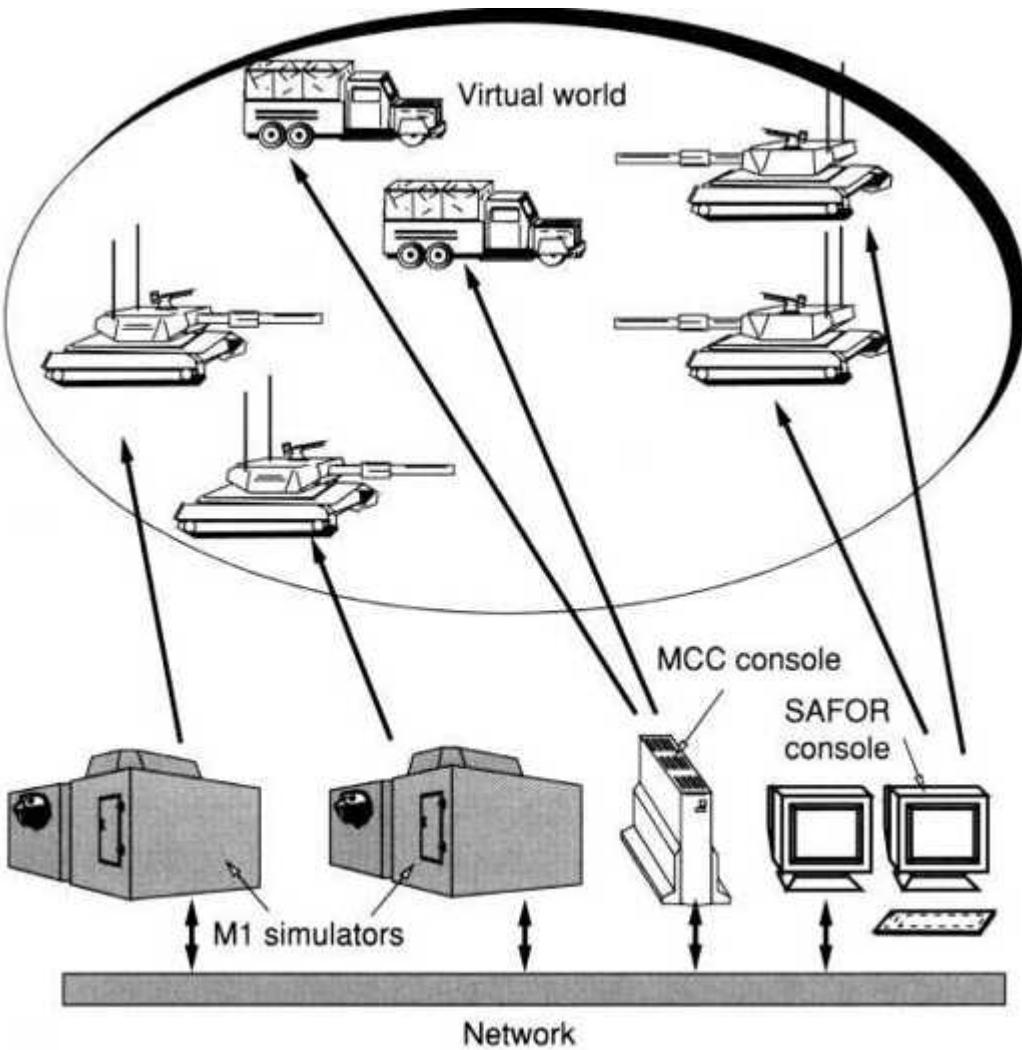


Fig. 8.28 The Simulation Network virtual battlefield. Adapted from McDonough [1993]. Reprinted by permission.

The addition of SAFOR and MCC workstations made the virtual engagement more realistic and dangerous. Battle damage and breakdowns were modeled in the system, as was gas and ammunition consumption. The latest entry into the SIMNET battlefield were Stealth Vehicles, which received, but did not broadcast, any messages on the network. In addition to eavesdropping on the network, they could take their occupants anywhere on the battlefield and let them observe enemy and friendly troops. The first combat use of these Stealth Vehicles was during the Gulf War of 1991. The military planners could thus navigate in the Kuwait Theater database and observe the likely disposition of the Iraqi units represented as SAFOR

troops. Immediately after the ground war ended, DARPA research teams collected detailed real data on a tank battle at "73 Easting." Subsequently, the real battle events were programmed into SAFOR forces and network traffic recorded. This recreation of the battle could then be played back to Stealth Vehicles that could position themselves anywhere and at any time in the battle. Thus they could roll the simulation forward and backward to the desired event in what McDonough called "the world's first four dimensional history book."

The ability to train troops in a VR mock-up before the actual engagement, and then to do a VR debriefing was a very powerful incentive for the military to invest in this new technology. SIMNET, while useful, had a number of shortcomings. First, it only allowed training in daylight missions and without weather effects (rain, fog). Furthermore it used nonstandard DIS communication and had a small terrain database for training. Finally, its graphics was low-resolution owing to the slow hardware then available. As a consequence, the SIMNET system is being replaced by the Close Combat Tactical Trainer (CCTT), which has a much larger virtual training area (in excess of 70,000 km²). The Army CCTT is fielded in two versions, fixed and mobile. The fixed installations are larger (allowing battalion-size training) and are intended for training active military units. The mobile CCTT simulators are trailer-based and intended for training Army Reserve and National Guard units. Furthermore, CCTT simulators are designed to train tank, infantry vehicles, dismounted infantry, as well as aviation support units.

An aviation CCTT (AVCCTT) mobile unit configuration is illustrated in Figure 8.29a. It contains a battle master control (BMC) room, an after action review theater (AART), and six reconfigurable manned modules. The BMC houses workstations for situation awareness, observer controller, and the SAF console. The AART has a large screen to present the mission successes and errors as well as recording equipment to tape mission debriefings. The six manned modules are the actual simulators where trainees fly various helicopter models. They have displays showing through-the-window views of the aircraft surroundings as well as replicas of cockpit dials. The modular design of the manned modules allows easy

reconfiguration for cargo, attack, reconnaissance, and assault mission helicopters. When networked over optical fiber and using standard DIS protocols, the various AVCCTTs allow complex team training. Furthermore, the AVCCTTs can network with tank CCTTs to more realistically train in close air support missions, as illustrated in Figure 8.29b.

8.3.2 VR Applications in the Navy

The U.S. Navy started its own studies in the use of virtual reality in the early 1990s. These studies, conducted at the Naval Command, Control and Ocean Surveillance Center in California, were prompted by the realization that the complex displays of a ship were producing operator overload. Earlier human factors studies had shown that operator performance improved with reduced workload, and the effect was amplified over prolonged use. There was a need to create new displays that were more natural to understand and that took advantage of human perceptual and cognitive skills [Gembicki and Rousseau, 1993]. Perceptual and cognitive skills are particularly important during submarine surface navigation and the operation of close-range naval artillery.

8.3.2.1 Virtual Environment for Submarine Ship Handling Training (VESUB). This simulator for "officer on deck" (OOD) training is currently in use at the Naval Submarine School in Groton, Connecticut. Navigating a surfaced submarine close to shore is difficult owing to the low profile of the vessel and the abundance of other vessels and obstacles ("contacts" in maritime parlance). The development of the VR simulator was prompted by the realization that older shore-based OOD trainers were inadequate. As a consequence, junior officers needed to learn on the job during the limited time of surface steaming, a situation that was risky.

The VESUB simulator, which became operational in early 2001 [Hanson, 2001], is illustrated in Figure 8.30. The trainee officer stands in a mock-up of a submarine deck while wearing a high-resolution stereo HMD. The trainee's head motion is tracked such that he or she can see a 360° view of the submarine and its surroundings. The virtual scene is rendered by an SGI

workstation and input to the simulation is through voice commands given over a mock communication system. Voice traffic (from simulated stations inside the virtual submarine) as well as other sounds (wind, ocean, harbor pilot, etc.), are fed back on the HMD headphones. The HMD can also be used as virtual binoculars allowing zooming in on distant contacts. The training scenario is controlled by an instructor sitting at an adjacent workstation. The instructor monitors the trainee's view on one of the workstations while changing weather conditions, seaport databases (four harbor models), or introducing emergencies (man overboard, or shallow water under the boat).

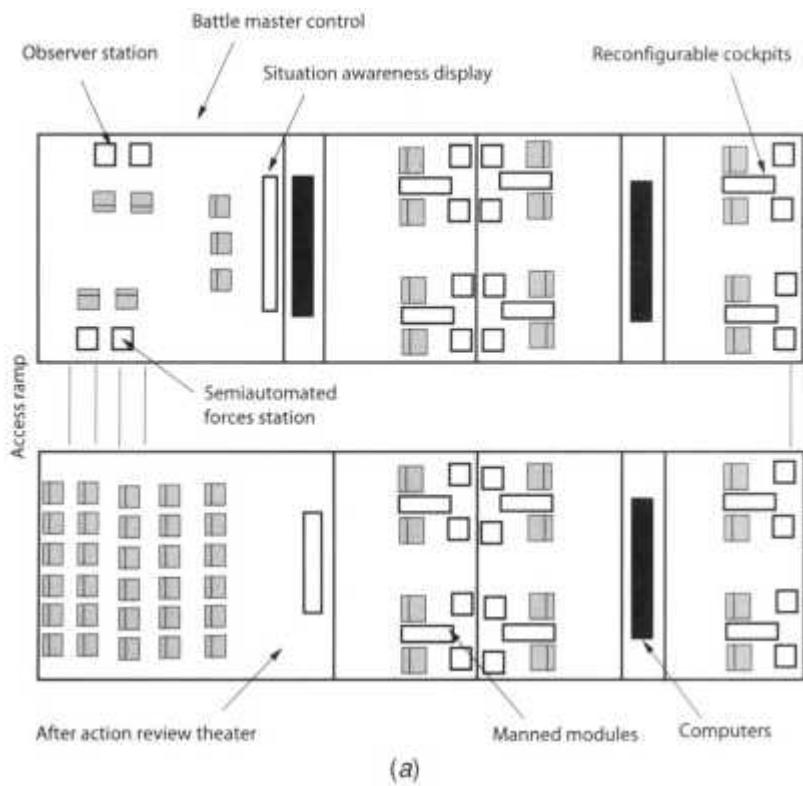
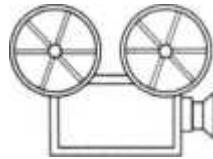


Fig. 8.29 The Close Combat Tactical Trainer simulator: (a) mobile AVCCTT setup; (b) virtual scene. From [National Simulation Center, 2002]. Reprinted by permission from the U.S Army.



VC 8.7

Prior to field deployment, the VESUB simulator underwent a training effectiveness evaluation study [Hays et al., 1998]. The experimental design was within-subject for mission scenario and between-subjects for prior experience (from 0 to 14 years). Each of the 41 subjects participated in one 3-hour simulation session consisting of familiarization, training, and examination scenarios. Results showed significant task learning (skill improvement) on several variables, such as issuing correct turning commands, contact management skills, checking range markers, and using correct commands during emergencies. The most learning occurred, unsurprisingly, with subjects who had little or no prior submarine handling experience. The most experienced subjects had little or negative learning (their performance was lower following the training on the simulator). Researchers speculated that these subjects had strong prior mental images of the task that may not have corresponded with the virtual model or simply did not sufficiently engage in the simulation (they did not take the training seriously).

8.3.2.2 VR-Based Close-Range Naval Artillery Training. This is used in the British Royal Navy to replace shore-based live fire training. The VR-based simulator, which entered service in 2001, is modular in design, allowing training for 20-mm and 30-mm rapid-fire cannons and other close-range artillery. Its fielding was necessary due to the high cost of ordinance as well as the negative impact on ocean habitats, which led to the suspension of shore-based live fire training in the United Kingdom. Further costs reduction resulted from the use of commercial hardware as opposed to proprietary defense computing hardware.

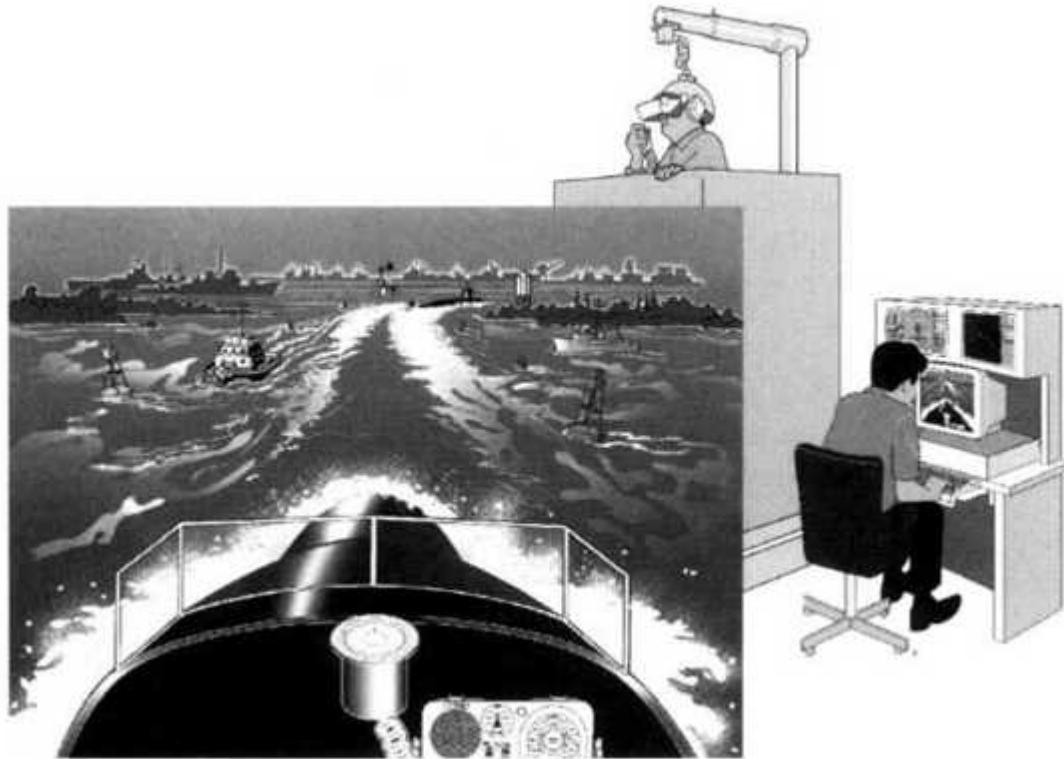


Fig. 8.30 View of the VESUB simulator. From Hays et al. [1998]. Reprinted by permission.

Of all close-range artillery models, the 20-mm gun is the most common deckmounted weapon in the British Royal Navy. It is a single-barrel cannon with a range of 2 km, firing up to 900 rounds per minute. The VR-based trainer incorporates an actual 20-mm cannon, as shown in Figure 8.31a [Stone and Rees, 2002]. Its sighting display was removed and replaced by a Keiser ProView XL50 binocular HMD. The HMD was modified by the addition of a button that allows zooming in-out, similar to the function of virtual binoculars. The trainee's head motion is tracked by an InterSense IS-900 tracker since the metal in the cannon and its surroundings made it impossible to use magnetic trackers. The same tracker measures the cannon azimuth and elevation, while the trigger is sensorized such that the host computer detects when the cannon is fired.

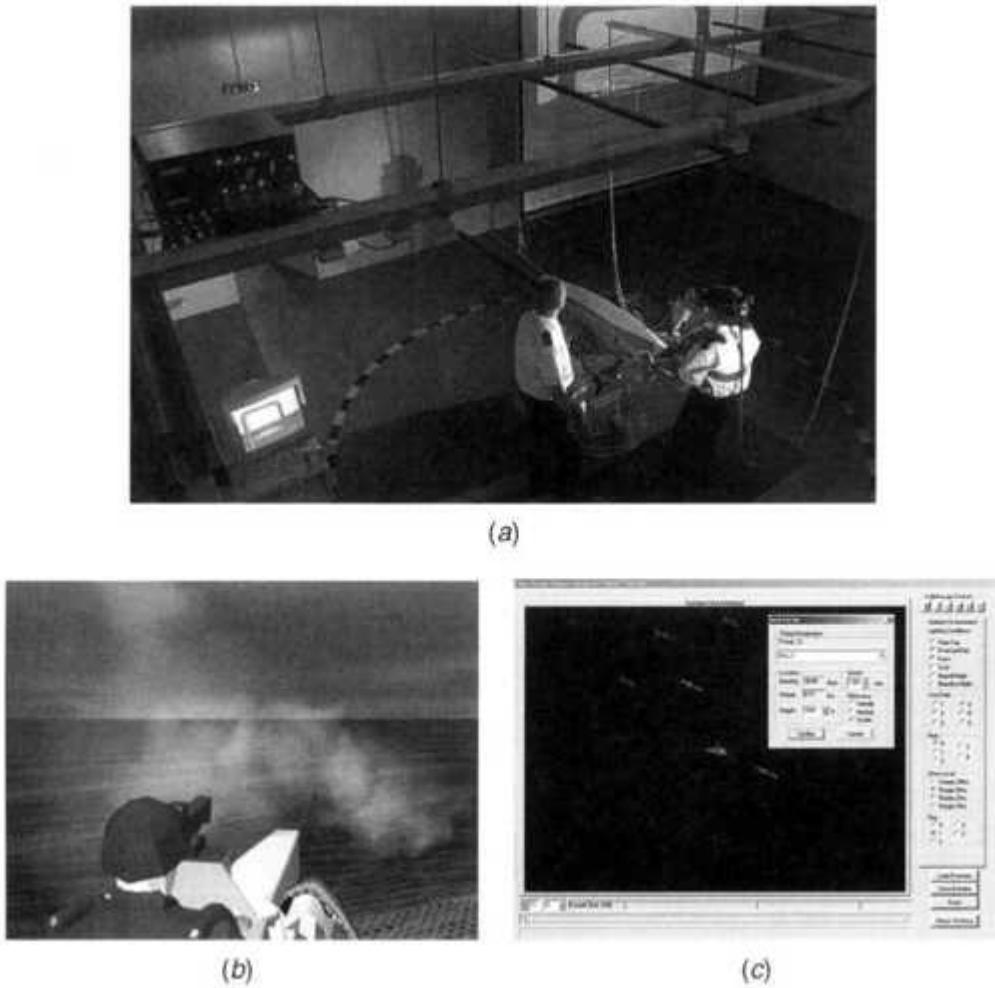


Fig. 8.31 Close-range naval artillery training: (a) simulator system; (b) gunner avatar seen by the gunner director; (c) scenario control interface view. From Stone and Rees [2002]. Reprinted by permission.

The host computer is a Pentium 1-GHz dual-processor PC with NVIDIA Co. GeForce 2 graphics accelerator. It is interfaced with a second PC that is dedicated to the gunner director platform. On a real ship the gunner director stands on a platform above the 20-mm cannon emplacement and is tasked to spot targets and their range and bearing and communicate these to the gunner. The gunner director then observes the gunner during the target engagement. In the simulator the gunner director wears a similar HMD and can see the ship surroundings as well as the gunner avatar (see Fig. 8.31b). A third PC integrated in the system is dedicated to the instructor station.

The instructor looks at a scenario control interface (SCI) and can communicate with higher echelons on a modem. The SCI screen displays the ship at the center surrounded by friendly, neutral, and enemy vessels, aircraft, and other entities. The instructor can modify the training scenario, save it for replay, change environmental conditions (daylight, dusk, moonlight, moonless night), sea state (calm to stormy), rain, fog levels and cloud base. At the end of training the instructor replays the session talking through the trainee's progress, viewing the action from different views (from ship or target viewpoints). Since its recent deployment, the Close Range Weapons Simulator has trained hundreds of students and saved the British Royal Navy tens of millions of dollars in live rounds and live aircraft hours saved.

8.3.3 Air Force Use of VR

Unlike doctors, pilots (civilian and military) have a long tradition of simulator-based training owing to the grave consequences human error has on passengers. The traditional flight simulator has been a large domelike structure placed on a motion platform and housing an identical replica of an airplane cockpit. While such training is very realistic (including engine sounds, airflow turbulence, and other effects), it is also very expensive. One hour on such simulators can cost \$5000. As a consequence, motion platform-supported dome simulators are housed in dedicated (fixed) facilities to which airlines send their pilot trainees.

Air forces around the world have to be on constant readiness and often redeploy aircraft and crews as missions dictate. It is thus becoming increasingly impractical to send crews to distant simulator facilities. What is needed is a smaller, transportable simulator that can travel with the squadron to accompany it in the combat zone. Furthermore, such unit simulator needs to be reconfigurable in order to keep up with frequent hardware upgrades.

8.3.3.1 The Unit Trainer and the Virtual Cockpit. Those are examples of the newer generation of simulators. They serve as multitask trainers, allowing

both standalone and networked simulations (necessary to train pilot teams and tactics).

Figure 8.32a shows a unit simulator for the A-10 ground attack airplane [Air Force Research Laboratory, 2002a]. The pilot sits in an exact replica of the cockpit, which gives direct haptic feedback and increased simulation realism. The scene is projected on three side-by-side portable displays, which are easily assembled on site. They create a 214° (horizontal) x 108° (vertical) field of view with a 1024 x 768 resolution. The computers are compact enough to be housed in the aircraft body, such that the simulator can be rolled in and assembled in a regular office room. The system also includes an instructor station such that pilots can train with their squadron and do so without use of real munitions or of bombing ranges. They can also get familiarity with the local flying area on an overseas deployment by using appropriate databases [Kehres, 2002].

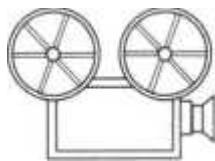


(a)



(b)

Fig. 8.32 Single-aircraft trainers. (a) A-10 simulator with real cockpit. From Air Force Research Laboratory [2002a]. Reprinted by permission. (b) view from the Mako aircraft virtual cockpit. From European Aeronautic Defense and Space Company [2001]. Reprinted by permission.



VC 8.8

Another type of unit trainer is the Mako aircraft simulator demonstrated at the 44th Paris Air Show in summer 2001. This two-seat advanced trainer and light attack airplane actually contains two simulators. The pilot in the front seat operates in an exact replica of the Mako aircraft cockpit and faces a wall-mounted display (similar to the A-10 simulator previously described). However, the pilot in the rear seat is immersed in a "virtual cockpit," as shown in Figure 8.32b [European Aeronautic Defense and Space Company, 2001]. Virtual cockpits use virtual switches projected on HMDs, 3D trackers, 3D sound generators, and sensing gloves. The first virtual cockpit, called the Virtual Environment Configurable Training Aids (VECTA), was demonstrated at the Paris Air Show in 1991. It was developed by British Aerospace [Kalawsky, 1993] as a tool for human factors design dealing with sensorial overload. The VECTA cockpit's tactile feedback was provided by actuators placed in the pilot's sensing gloves. The Mako virtual cockpit uses instead a combination of physical controls and flat barriers. The most important controls (thrust lever, flight stick, and pedals) are physically shaped. The feeling of touching the flat instrument display is provided by the correctly placed flat barriers. Thus, passive rather than active haptics is used in order to simplify the system and reduce its cost. Another reason not to replicate the cockpit dials and knobs exactly is the need for flexibility and adaptability, as the Mako aircraft is still under development.

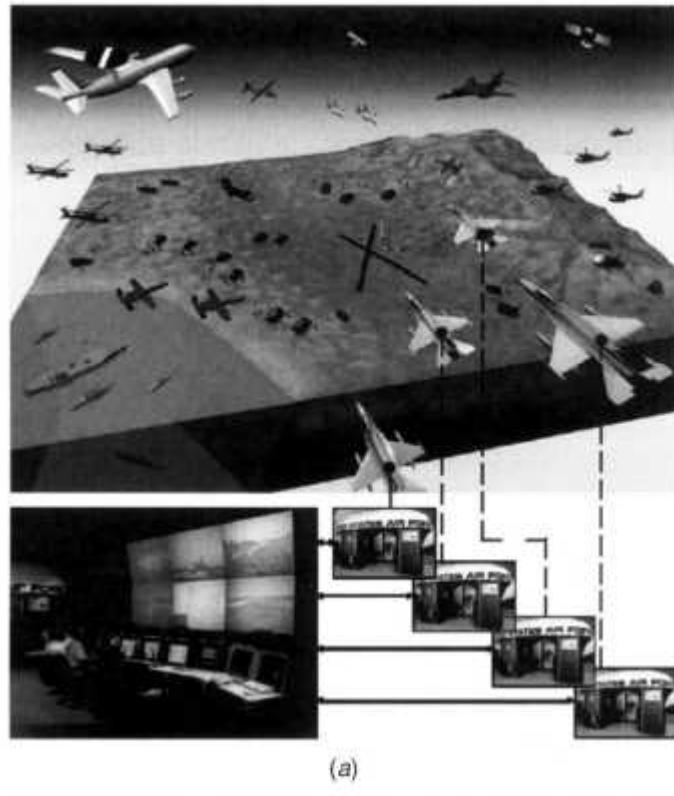
8.3.3.2 Distributed Mission Training (DMT). This networked simulation environment is being created by the U.S. Air Force Research Laboratory (Mesa, Arizona) in order to allow team pilot training [Air Force Research Laboratory, 2002b]. As illustrated in Figure 8.33a, mission team level training is based on networked nodes of four fighter simulators, Air Warning System (AWACS) simulators, and synthetic (computer-generated) enemy forces ("bandits"). An instructor station has six tiled displays that

show the mission in real time. Several such mission training simulators can then be networked over a WAN in order to allow remote Air Force units to train in complex missions.

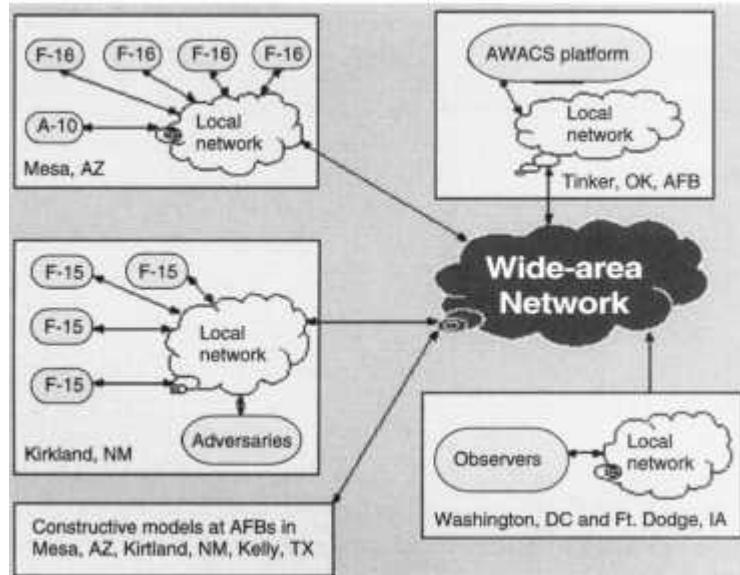
Of particular importance are missions in which pilots are in numerical inferiority, the so-called "four versus many Dissimilar Air Combat Tactics" (4 v X DACT) [Jensen and Crane, 1999]. Such training is rarely done on firing ranges owing to cost, safety concerns, and limited air space availability. However, DMT is ideal for this type of training since there is no limit on air space, altitude, or time spent in the mission. The DMT thus allows pilots to train their radar operating and communication skills in a highly realistic VR environment where opposing forces can either be other pilots or automated forces. The whole simulation is taped, allowing postmission debriefing either at the local node level or global battle space level. Typically, pilots fly two 1-hour missions each day, with a debriefing after each mission. The morning debriefing analyzes the mission mistakes, which pilots can improve upon by repeating the same simulation in the afternoon.

The DMT system was tested in summer 1998 in a week-long large-scale Roadrunner'98 exercise sponsored by the U.S. Air Force Modeling and Simulation Office. As illustrated in Figure 8.33b [Crane, 1999], the mission teams consisted of clusters of four F-16s, four F-15s, and a single A-10 simulator (acting as forward air controller). The F-16s and A-10 communicated over a LAN in Mesa, Arizona, while the F-15s were networked over a LAN at Kirkland, New Mexico. The two groups of simulators then communicated over a WAN that grouped AWACS simulators at Tinker, Oklahoma, and an observer station at the Pentagon, Washington, D.C. These friendly assets were used to fight numerically superior MIG-29s, SU-27 fighters, Mi-24 Hind helicopters, tanks, and air defenses (radar, surface-to-air missiles, and antiaircraft artillery). All enemy forces were computer-generated. Pilot trainees with varying degree of experience (in terms of flight hours) flew a total of 24 missions (air-to-air, air-to-ground, and rescue). The pilot's performance was assessed by instructors at monitoring stations as well as by data analysis from simulation-specific variables (number of intercepts, kill/loss ratio, correct

communication procedures, etc.). Their overall mission performance improved significantly between the first and the last day of the Roadrunner'98 exercise, jumping from 1.5 to 3.5 (4 being the maximum). Improvements were noticed in the communication skills, situation awareness (the pilots' mental 3D model of the battle), adaptability, and crew coordination. Interestingly, junior trainees, with no flight lead experience, were asked to take charge of missions (against overwhelming forces) and execute them correctly. By the end of the training week they were able to do so at least as well as seasoned veterans. These results showed that the DMT environment can be used successfully in team battle skill training as well as in leadership training.



(a)



(b)

Fig. 8.33 Distributed Mission Training (DMT) simulators. (a) Networked units. From Air Force Research Laboratory [2002b]. Reprinted by permission. (b) RoadRunner'98 DMT largescale exercise configuration. From Crane [1999]. © 1999 ACM Inc. Reprinted by permission.

The DMT infrastructure is being expanded under the umbrella of the Joint Simulation Systems (JSIMS), an open-architecture, Department of Defense Joint Taskforce simulation environment. Unlike existing proprietary architectures, which have difficulty communicating with each other, JSIMS is PC-based and uses C++ as its programming language. The goal is to link Army (CCTT), Navy, and Air Force (DMT) simulators in a global battlefield scenario. This is necessary to improve the training of joint force commanders in both operational and strategic-theater joint tasks.

8.4 CONCLUSION

This chapter surveyed a large number of traditional applications of virtual reality. These span the fields of medicine, entertainment, the arts and education, and the military. Although the whole field is rapidly changing, these more mature application domains have sufficient human factors studies to show benefits. There are still problems related to hardware, although the rapidly proliferating PC graphics is helping to open new doors. These emerging VR applications are discussed in the final chapter of this book.

8.5 REVIEW QUESTIONS

1. What are the most important VR application domains and geographic areas, according to the CyberEdge Information Systems [2002] survey?
2. What is the Visible Human database and how was it created? Describe some of its applications.
3. What is the Digital Anatomist project, and why is it increasing access for students?
4. Describe the system architecture and use of the BioSimMER system.
5. Describe the prostate palpation trainer. Comment on its human factors testing results.

6. What is the PreOp bronchoscopy training simulator? What did the Colt et al. [2001] study show about its training efficacy?
7. How does virtual colonoscopy differ from a real colonoscopy?
8. Why are nurses' IV procedure skills important? Describe the CathSim training system, including the type of patients it can simulate.
9. What is anastomosis, and how can it be taught in VR? What did the O'Toole et al. [1998] study show?
10. How is laparoscopy training done with MIST-VR? Is the system able to measure skill? Give examples.
11. What is telerehabilitation, and how was it realized for orthopedic patients?
12. What advantages does VR offer for rehabilitation, particularly for post-stroke patients? Give examples.
13. How is VR used for patients with fear of flying? What is the system architecture and study outcomes?
14. Why is ADHD assessment in a virtual classroom more realistic than current questionnaire-based methods?
15. How is VR being used in education (at college, high school, and elementary school levels)? Give examples and comment.
16. What is "virtual heritage" and what steps are needed to build a heritage edifice? Give examples.
17. Give types of PC video games.
18. How do location-based VR experiences compete with the home-based systems? Give examples.
19. Give examples of single-soldier VR-based training systems.

20. How is leadership training at the platoon level done more realistically in VR? Describe the Bosnia accident scenario.
21. Describe the SIMNET architecture.
22. What is the AVCTT, and how is it used for training?
23. What is the VESUB? What did the Hays et al. [1998] study show?
24. What is the architecture of the close-range artillery trainer used in the British Royal Navy?
25. What is a virtual cockpit? Give examples.
26. Why is DMT important for military pilots? How is it being combined with Army CCTT simulators?
27. Describe the Roadrunner '98 large-scale military exercise and its results.

REFERENCES

- Air Force Research Laboratory, 2002a, "Multitask Trainer and Unit Level Trainer: A 10, F16, C-130," Williams Air Force Base, online at www.williams.af.mil/html/mttult.htm.
- Air Force Research Laboratory, 2002b, "Distributed Mission Training Technology and Methods," Williams Air Force Base, online at www.williams.af.mil/html/dmtdev.htm.
- Amato, I., 2001, "Helping Doctors Feel Better," *Technology Review*, April 2001, online at www.technologyreview.com/articles/amato0401.asp.
- American Association of Orthopedic Surgeons, 2002, "Broken Ankle," online at orthoinfo.aaos.org.

American Stroke Association, 2002, "Impact of Stroke," online at www.strokeassociation.org.

Anon, 1996, "VR in Medicine," VR News, Vol. 5(3), pp. 34-37.

Anon, 1998, "Smooth Operators: Assessing Surgeons' Sewing Skills." Wellcome News, Issue 16 (Q3), online at www.wellcome.ac.uk/en/Ilbiosfgunkinfshrep98taf.html.

Badique, E., M. Cavazza, G. Klinker, G. Mair, T. Sweeny, D. Thalmann, and N. Thalmann, 2002, "Entertainment Applications of Virtual Environments," in K. Stanney (Ed.), *The Handbook of Virtual Environments Technology*, Erlbaum, Mahwah, NJ, pp. 1143-1166.

Bernardini, F., H. Rushmeier, I. Martin, J. Mittleman, and G. Taubin, 2002, "Building and Digital Model of Michelangelo's Florentine Pieta," IEEE Computer Graphics and Applications, 2002(January/February 2002), pp. 59-67.

Boian, R., A. Sharma, C. Han, G. Burdea, A. Merians, S. Adamovich, M. Recce, M. Tremaine, and H. Poizner, 2002, "Virtual Reality-Based Post-Stroke Hand Rehabilitation," in *Proceedings of Medicine Meets Virtual Reality 2002*, IOS Press, Amsterdam, pp. 64-70.

Boian, R., H. Kourtev, K. Erickson, J. Deutsch, J. Lewis and G. Burdea, 2003, "Dual StewartPlatform Gait Rehabilitation System for Individuals Post-Stroke. "Proceedings of Second International Workshop on Virtual Rehabilitation," Rutgers University, Piscataway NJ, pp. 92.

Bouzit, M., G. Burdea, V. Popescu, and R. Boian, 2002, "The Rutgers Master Force Feedback Glove," IEEE/ASME Transactions on Mechatronics, Vol. 7(2), pp. 256-263.

Brill, L., 2001, "Illusion: Jumping Feet First into Immersive Entertainment," Real Time Graphics, Vol. 10(4), pp. 6-8.

Brinkley, J., B. Wong, K. Hinshaw, and C. Rosse, 1999, "Design of an Anatomy Information System," IEEE Computer Graphics and Applications, 1999 (May/June), pp. 38-48.

Brooks, Jr., F., 1999, "What's Real About Virtual Reality," IEEE Computer Graphics and Applications, I999(November/December), pp. 16-27. Also online at www.cs.unc.edu/brooks/WhatsReal.pdf.

Burdea, G., G. Patounakis, V. Popescu, and R. Weiss, 1999, "Virtual Reality-Based Training for the Diagnosis of Prostate Cancer," IEEE Transactions on Biomedical Engineering, Vol. 46(10), pp. 1253-1260.

Burdea, G., V. Popescu, V. Hentz, and K. Colbert, 2000, "Virtual Reality-based Orthopedic Tele-rehabilitation," IEEE Transaction on Rehabilitation Engineering, Vol. 8(3), pp. 429432.

Burger, T., 2000, "MIST VR and Beyond-ESI Experiences with VR Training Systems," presented at American Association of Gynecology, San Francisco, November 2000. Also online at www.esi-online.de.

Colt, H., S. Crawford, and O. Galbraith, 2001, "Virtual Reality Bronchoscopy Simulation," Chest, Vol. 120(4), pp. 1333-1339. Also online at www.chstjournal.org.

Crane, P., 1999, "Implementing Distributed Mission Training," Communications of the ACM, Vol. 42(9), pp. 90-94.

CyberEdge Information Services, 2000, The Market for Visual Simulation-Virtual Reality Systems, 3rd ed., CyberEdge Information Services, New York.

CyberEdge Information Systems Inc., 2002, The Market for Visual Simulation/Virtual Reality Systems, 5th ed., CyberEdge Information Systems, New York.

Daventry Associates Inc., 1993, "Virtual Reality Inc.," company report, Daventry Associates Inc., New York.

Dede, C., M. Salzman, B. Loffin, and K. Ash, 2000, "The Design of Immersive Virtual Learning Environments: Fostering Deep Understanding of Complex Scientific Knowledge," in M. Jacobson and R. Kozma (Eds.), Innovations in Science and Mathematics Education: Advanced Designs for Technologies of Learning, Erlbaum, Hillsdale, NJ, pp.361-413.

Delaney, B., 2000, "The Future of History," Real Time Graphics, Vol. 8(8), pp. 12-14.

DeLeon, V., and R. Berry, 2000, "Bringing VR to the Desktop: Are You Game?" IEEE Multimedia, Vol. 7(2), pp. 68-72.

Deutsch, J., J. Latonio, G. Burdea, and R. Boian, 2001a, "Rehabilitation of Musculoskeletal Injury Using the Rutgers Ankle Haptic Interface: Three Case Reports," in Proceedings of Eurohaptics 2001 Conference, Birmingham, U.K., pp. 11-16.

Deutsch, J., J. Latonio, G. Burdea, and R. Boian, 2001b, "Post-Stroke Rehabilitation with the Rutgers Ankle System-A Case Study," Presence, Vol. 10(4), pp. 416-430, August.

Division Ltd., 1993, "Training and Simulation," company brochure, Division Ltd., Bristol, U.K.

Emory Health Sciences, 2000, "Virtual Reality Therapy Proven Effective to Combat Fear of Flying," press release, 18 December 2000, online at www.emory.edu.

European Aeronautic Defense and Space Company, 2001, "Flying with the Helmet Mounted Display," Mforum, 2001 (July), pp. 13.

Fisher, S., 1999, "Real-time interactive graphics: taking location-based entertainment to the next level," ACM SIGGRAPH, Vol. 33(1), pp. 18-20.

Gallagher, A., N. McClure, J. McGuigan, I. Crothers, and J. Browning, 1999, "Virtual Reality Training in Laparoscopic Surgery: A Preliminary

Assessment of Minimally Invasive Surgical Trainer Virtual Reality (MIST VR)," *Endoscopy*, Vol. 3](4), pp. 310-313.

Gembicki, M., and D. Rousseau, 1993, "Naval Applications of Virtual Reality," in *Proceedings of Virtual Systems'93 Conference*, New York, pp. 269-288.

Girone, M., G. Burdea, and M. Bouzit, 1999, "The 'Rutgers Ankle' Orthopedic Rehabilitation Interface," in *Proceedings of the ASME Haptics Symposium*, DSC-Vol. 67, ASME, pp. 305-312.

Gold Standard Multimedia, 2000, "The Virtual Human Gallery," online at vhgallery.gsm.com.

Griffith, D., 1999, "Focus on Virtual Jungle Cruise," *Micro Publishing News*, online at www.micropubnews.com/pages/issues/1999/799-jungle-inpn.shtml

Hanson, B., 2001, "Invention and Innovation: Training at Submarine School," *Naval Submarine School*, online at www.cnet.navy.mil/newlondn/invention.htm.

Hays, R., D. Vincenzi, A. Seamon, and S. Bradley, 1998, "Training Effectiveness Evaluation of the VESUB Technology Demonstration System," Technical Report 98-003, Naval Air Warfare Center Training Systems Division, Orlando FL.

Helsel, S., and S. D. Doherty, 1993, "Virtual Reality Market Place 1993: Insights from the Meckler Database," in *Proceedings of Virtual Reality '93*, San Jose, CA, pp. 209-211.

Hirshfeld, Jr., J., S. Ellis, and D. Faxon, 1998, "Recommendations for the Assessment and Maintenance of Proficiency in Coronary Interventional Procedures: Statement of the American College of Cardiology," *Journal of American College of Cardiologists*, Vol. 31, pp.722-743.

Hodges, L., B. Rothbaum, B. Watson, D. Kessler, and D. Opdyke, 1996, "A Virtual Airplane for Fear of Flying Therapy," in Proceedings IEEE VRAIS, pp. 86-93.

Holden, H., 2002, "Virtual Environment Training: A New Tool for Neurorehabilitation," *Neurology Report*, Vol. 26(2), pp. 62-71.

Holden, M., and E. Todorov, 2002, "Use of Virtual Environments in Motor Learning and Rehabilitation," in K. Stanney (Ed.), *The Handbook of Virtual Environments Technology*, Erlbaum, Mahwah, NJ, pp. 999-1026.

HT Medical Systems Inc., 1998a, "PreOp Endoscopy Simulator," company brochure, HT Medical Systems Inc., Rockville, MD.

HT Medical Systems Inc., 1998b, "Virtual Reality takes the 'practice' out of the 'Practice of Medicine,'" news release, 4 March 1998, online at www.immersion.com.

Immersion Co., 2002, "Laparoscopic Surgical Workstation," Immersion Co., San Jose, CA, online at www.immersion.com/medical/docs/LSW-date-Sheet.pdf.

Jack, D., R. Boian, A. Merians, M. Tremaine, G. Burdea, S. Adamovich, M. Recce, and H. Poizner, 2001, "Virtual Reality-Enhanced Stroke Rehabilitation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 9(3), pp. 308-318.

Jensen, R., and P. Crane, 1999, "Human-Centered Development for Distributed Mission Training Systems," in Proceedings of Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), online at www.williams.af.mil/pdf/PCraneRJensen.pdf.

Kalawsky, R., 1993, *The Science of Virtual Reality and Virtual Environments*, Addison-Wesley, Reading, MA.

Kehres, P., 2002, "Defense News-New Simulator Helps Train A-10 pilots." *Journal of Aerospace and Defense Industry News*, 22 February 2002,

online at www.aerotechnews.coin/starc/2002/022202/AIO-simulator.html.

Knerr, B., R. Breaux, S. Goldberg, and R. Thurman, 2002, "National Defense," in K. Stanney (Ed.), *The Handbook of Virtual Environments Technology*, Erlbaum, Mahwah, NJ, pp. 857-872.

Kopp, B., A. Kunkel, W. Muhlnickel, K. Villringer, E. Taub, and H. Flor, 1999, "Plasticity in the Motor System Related to Therapy-Induced Improvement of Movement After Stroke," *Neuroreport*, Vol. 10, pp. 807-810.

Lewis, J., R. Boian, G. Burdea, and J. Deutsch, 2003, "Real-Time Web-Based Telerehabilitation Monitoring," in *Medicine Meets Virtual Reality Conference*, pp. 190-192.

Liang, J., 2001, "Virtual Colonoscopy: An Alternative Approach to Examination of the Entire Colon," Viatronix Co., Stony Brook, NY. Also online at www.viatronix.com.

Lindheim, R., and W. Swartout, 2001, "Forging a New Simulation Technology at the ICT," *Computer*, 2001 (January), pp. 72-79.

Loftin, B., M. Engelberg, and R. Benedetti, 1993, "Applying Virtual Reality in Education: A Prototypical Virtual Physics Laboratory," in *Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality*, San Jose, CA, pp. 67-74.

Lorensen, B., 2001, "Marching Through the Visible Man," General Electric Global Research, New York. Online at www.crd.ge.com/esvcgsp/projects/vm.

Marchese, M., 2002, "The Reality of Virtual Colonoscopy," *Medical Imaging*, Vol. 170, online at www.viatronix.net/PDFs/article-RPR-MIDI-0302.pdf.

McDonough, J., 1993, "Doorways to the Virtual Battlefield," in *Proceedings of Virtual Reality '92*, Meckler, pp. 104-114.

Moshell, M., and C. Hughes, 2002, "Virtual Environments as a Tool for Academic Learning," in K. Stanney (Ed.), *The Handbook of Virtual Environments Technology*, Erlbaum, Mahwah, NJ, pp. 893-910.

National Library of Medicine, 2001, "The Visible Human Project," online at www.nlm.nih.gov/research/visible/visible_human.html.

National Rural Health Association, 1999, "Legislative and Regulatory Agenda," NRHA e-News, Vol. 2(3), online at www.nrharural.org.

National Simulation Center, 2002, "Close Combat Tactical Trainer," Fort Leavenworth, online at www.leav.army.mil/nsc/tsm/cctt-p.html.

North, M., S. North, and J. Coble, 2002, "Virtual Reality Therapy: An Effective Treatment for Psychological Disorders," in K. Stanney (Ed.), *The Handbook of Virtual Environments Technology*, Erlbaum, Mahwah, NJ, pp. 1065-1078.

O'Toole, R., R. Playter, W. Blank, N. Cornelius, W. Roberts, and M. Raibert, 1997, "A Novel Virtual Reality Surgical Trainer with Force Feedback: Surgeon vs. Medical Student Performance," in Proceedings of the Second PHANTOM User's Group Workshop, MIT Artificial Intelligence Lab Technical Report No. 1617 and RLE Technical Report No. 618. Also online at www.bdi.com/Skills-test.html.

O'Toole, R., R. Playter, T. Krummel, W. Blank, N. Cornelius, W. Roberts, W. Bell, and M. Raibert, 1998, "Assessing Skill and Learning in Surgeon and Medical Student Using a Force Feedback Surgical Simulator," in Proceedings of the First International Conference on Medical Image Computing and Computer-Assisted Intervention, Cambridge, MA, pp. 899-909.

Papagiannakis, G., G. L'Hoste, A. Foni, and N. Magnenat-Thalmann, 2001, "RealTime Photo Realistic Simulation of Complex Heritage Edifices, in "Proceedings of Virtual Systems and Multimedia 2001, Berkeley, CA, pp. 218-227. Also online at [www.miralab.unige.ch/newMIRA\[Research](http://www.miralab.unige.ch/newMIRA[Research).

Pasquier, T., 2000, "Influence of Sensory Modes and Personal Cues on Buried Targets (Mines) Detection: Comparing Real and Virtual Environments," Ph.D. Thesis, University of Angers, France.

Popescu, V., G. Burdea, M. Bourit, M. Girone, and V. Hentz, 2000, "Orthopedic Telerehabilitation with Virtual Force Feedback," IEEE Transactions on Information Technology, Vol. 4(1), pp. 45-51.

Reichert, M., 2000, "Advanced Air Defense Training Simulation System," in CD Proceedings of NATO Workshop on "What is Essential for Virtual Reality to Meet Military Human Performance Goals?" Hague, The Netherlands. p. 5-1.

Rich, J., 1993, "Virtual Reality: The New Dimension," Sega Guide, Vol. 4(2), pp. 17-22.

Rigamonti, D., H. Bryant, O. Bustos, L. Moore, and H. Hoffman, 2000, "Implementing Anatomic VisualizeR Learning Modules in Anatomy Education," in Proceedings of the Third Visible Human Project Conference [CD-ROM], Bethesda, MD. Online at www.nlm.nih.gov/research/visible/vhpconf2000/AUTHORS/HOFFMAN/HOFFMAN.HTM.

Rizzo, A., J. Buckwalter, T. Bowerly, C. van der Zaag, L. Humphrey, U. Neumann, C. Chua, C. Kyriakakis, A. van Rooyen, and D. Sisemore, 2000, "The Virtual Classroom: A Virtual Reality Environment for the Assessment and Rehabilitation of Attention Deficits," CyberPsychology and Behavior, Vol. 3(3), pp. 483-499.

Rizzo, A., J. Buckwalter, J. McGee, T. Bowerly, C. van der Zaag, U. Neumann, M. Thiebaux, L. Kim, J. Pair, and C. Chua, 2001, "Virtual Environments for Assessing and Rehabilitating Cognitive/Functional Performance. A Review of Projects at the USC Integrated Media Systems Center," Presence, Vol. 10(4), pp. 359-374.

Rizzo, A., G. Buckwalter, and C. van der Zaag, 2002, "Virtual Environment Applications in Clinical Neuropsychology," in K. Stanney (Ed.), The

Handbook of Virtual Environments Technology, Erlbaum, Mahwah, NJ, pp. 1027-1064.

Robinett, W. and M. Naimark, 1992, "Artists Explore Virtual Reality: The Bioapparatus Residency at the Banff Centre for the Arts," *Presence*, Vol. 1(2), pp. 248-250.

Rosen, J., A. Lasko-Harwill, and R. Satava, 1996, "Virtual Reality and Surgery," in R. Taylor, S. Lavallee, G. Burdea, and R. Moesges (Eds.), *Computer Integrated Surgery*, MIT Press, Cambridge, MA, pp. 231-243.

Rothbaum, B., L. Hodges, P. Anderson, L. Price, and S. Smith, 2002, "Twelve-Month FollowUp of Virtual Reality and Standard Exposure Therapies for the Fear of Flying," *Journal of Consulting and Clinical Psychology*, Vol. 70(2), pp. 428-432.

Roussos, M., A. Johnson, T. Moher, J. Leigh, C. Vasilakis, and C. Barnes, 1999, "Learning and Building Together in an Immersive Virtual World," *Presence*, Vol. 8(3), pp. 247-263.

Satava, R., 1993, "Virtual Reality Surgical Simulator-The First Steps," in *Proceedings of VR Systems '93 Conference*, New York, pp. 41-49.

Satava, R., and S. Jones, 2002, "Medical Applications of Virtual Environments," in K. Stanney (Ed.), *The Handbook of Virtual Environments Technology*, Erlbaum, Mahwah, NJ, pp. 937957.

Sato, M., S. Lakare, M. Wan, A. Kaufman, Z. Liang, and M. Wax, 2000, "An Automatic Colon Segmentation for 3D Virtual Colonoscopy," *IEICE Transactions on Information and Systems*, Vol. E84-D(1), pp. 201-208.

Schwartz, J., 1993, "A Computerized Cadaver to Aid Medical Students," *The New York Times*, 7 April 1993, p. D5.

Stansfield, S., D. Shawer, and A. Sobel, 1998, "MediSim: A Prototype VR System for Training Medical First Responders." in *Proceedings of IEEE*

Virtual Reality Annual International Symposium (VRAIS), Atlanta, GA, pp. 198-205.

Stansfield, S., D. Shawver, A. Sobel, M. Prasad, and L. Tapia, 2000, "Design and Implementation of a Virtual Reality System and Its Application to Training Medical First Responders," *Presence*, Vol. 9(6), pp. 524-556.

Stone, R.. 1999, "The Opportunities for Virtual Reality and Simulation in the Training and Assessment of Technical Surgical Skills," in *Surgical Competence: Challenges of Assessment in Training and Practice*, London, Royal College of Surgeons, pp. 109-125.

Stone, R., 2000, "Haptic Feedback: A Brief History from Telepresence to Virtual Reality," in S. Brewster, and R. Murray-Smith, (Eds.), *Proceedings of the First International Workshop on Haptic Human-Computer Interaction*, Springer-Verlag, Berlin, pp. 1-16.

Stone, R., 2001, "Virtual Reality in the Real World-A Personal Reflection on 12 Years of Human-Centered Endeavour." in *Proceedings of International Conference on Artificial Reality and Teleexistence (ICAT 2001)*, Tokyo, Japan, pp. 23-32.

Stone, R., and J. Rees, 2002, "Application of VR to the Development of Naval Weapons Simulator," in *Proceedings of 1/ITSEC*, pp. 129-139.

Taffinder, N., C. Sutton, R. Fishwick, I. McManus, and A. Darzi, 1998, "Validation of Virtual Reality to Teach and Assess Psychomotor Skills in Laparoscopic Surgery: Results from Randomized Controlled Studies Using the MIST VR Laparoscopic Simulator," in *Proceedings of Medicine Meets Virtual Reality*, IOS Press, Amsterdam, pp. 124-130.

Taxen, G., and A. Naeve, 2001, "CyberMath: A Shared Virtual Environment for Mathematics Exploration." Center for User Oriented IT Design, Royal Institute of Technology, Technical Report CID- 129, Stockholm, Sweden.

Taylor, R., S. Lavallee, G. Burdea, and R. Moesges (Eds.), 1996, Computer Integrated Surgery, MIT Press, Cambridge, MA.

University of Washington, 2001, "Digital Anatomist Interactive Atlas User Manual," online at www9.biostr.washington.edu/DA/docs/help/index.html.

Ursino, M., J. Tasto, B. Nguyen, R. Cunningham, and G. Merrill, 1999, "Cathsim: An Intravascular Catheterization Simulator on a PC," in Proceedings of Medicine Meets Virtual Reality, IOS Press, Amsterdam, pp. 360-366.

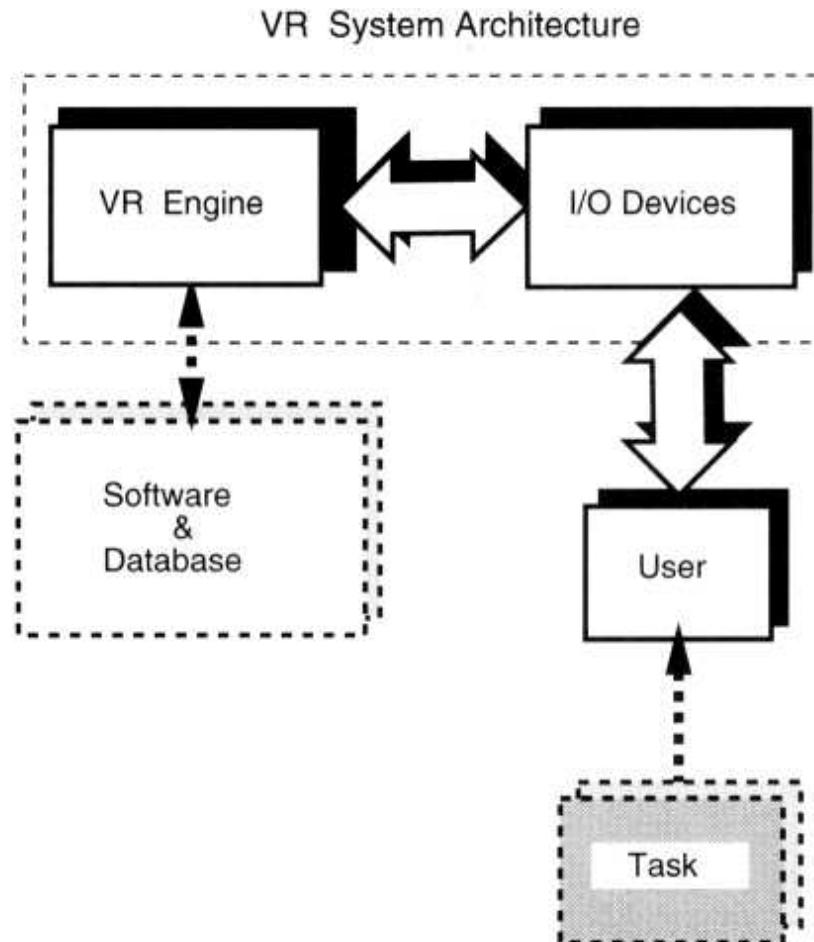
Virtual Heritage Network, 2002, online at www.virtualheritage.net.

Wax, M., 2002, "Virtual Colonoscopy Becomes Clinical Reality," Viatronix Co., Stony Brook, NY. Online at www.viatronix.com/pdf/article_dr-wax.pdf.

Yoon, J., R. Boian, J. Ryu, and G. Burdea, n.d., "Control of the Dual-Platform Rutgers Ankle'... _.. e.L...i ",,,io.. rAViccinn

CHAPTER 9

EMERGING APPLICATIONS OF VR



The previous chapter described VR applications in traditional areas such as medicine, education, arts, entertainment, and the military. Are there other domains where VR could prove beneficial now or in the near future? A survey conducted in the United Kingdom concerning the business potential of VR looked at the number of companies using, experimenting with, or considering VR [Information Society Initiative, 2000]. The survey found a significant VR presence in areas such as oil and gas exploration (46%),

telecommunications (29%), engineering contracting (26%), and the automotive industry (22%).

Emerging VR applications have less maturity and even less validation data at this time than the traditional applications described in Chapter 8. Nevertheless, it is important to review such applications here for the benefit of students and professionals alike. For presentation purposes, the discussion is organized along the topics of manufacturing, robotics, and data visualization. This is by no means comprehensive owing to space limitations and to the active research nature of the topic.

9.1 VR APPLICATIONS IN MANUFACTURING

Manufacturing is an important sector in most countries and represents the transition of products from the concept stage to production and sales. In the United States, the manufacturing sector accounts for about 20% of the gross national product, compared to primary industries (such as agriculture), which account for 5% [Shewchuk et al., 2002]. Consumers demand quality, variety, and low cost. Therefore companies are always looking for ways to increase production flexibility and cost savings. At the same time companies aim at reducing product development time (so-called time to market).

The life cycle of a manufacturing system involves the stages of design, manifestation (purchase of resources, equipment installation), personnel training, and operation [Williams, 1992]. Virtual reality has many attributes that are appealing to manufacturers. These include natural multimodal interaction (useful in concept design and personnel training), flexibility (beneficial to design revisions and small-batch production), and remote shared access (useful in ergonomic analysis, product approval, training, and marketing, where multidisciplinary teams are involved). The various applications of VR in manufacturing are illustrated in Figure 9.1 [Fraunhofer Institute for Computer Graphics, 1997].

9.1.1 Virtual Prototyping

An obligatory stage in the development of a new product is a prototype, its first physical realization. This stage validates the design, establishes manufacturing methods, and determines how easy to use (or ergonomic) the product is. Often prototype realization uncovers design mistakes and thus it may be iterative. The more complex the product, the more expensive is its physical mockup, especially when several versions need to be constructed.

An instance when prototypes are extremely expensive occurs in aircraft and automotive manufacturing. A new car model mockup, for example, is realized to study the visual effect of a designer's concept. It is made of clay at full scale, painted to look like a real car, and may cost 1 million dollars to realize. It is thus not surprising that aircraft and car manufacturers have been pioneers in the introduction of virtual prototyping as a way to replace the more expensive physical counterpart. Apart from cost savings, virtual prototypes have the advantages of flexibility (it is easier to change software than to build a new physical mockup), online documentation, and the ability to be viewed remotely (for design approval and marketing). Our discussion of virtual prototyping groups several related topics: free-form design and validation, CAD and automotive assembly verification, and ergonomic studies.

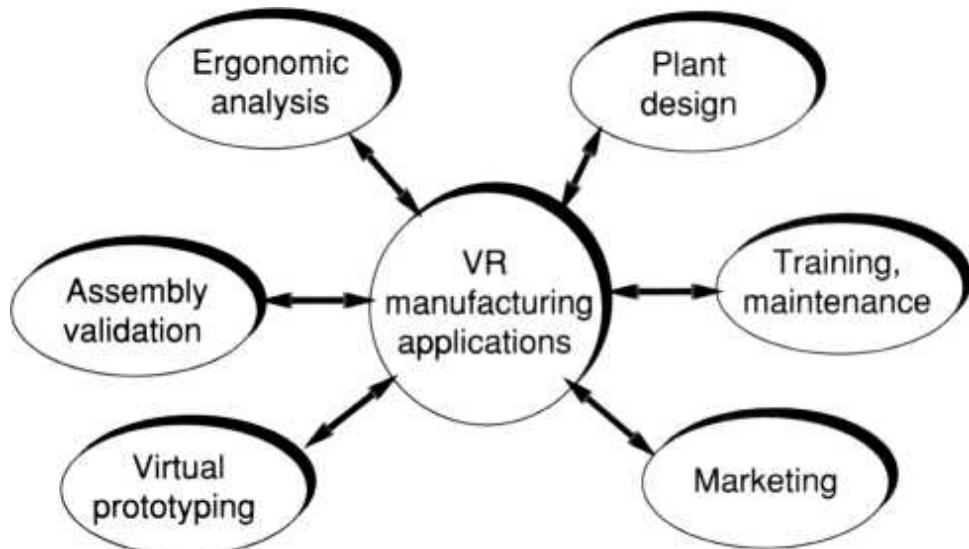
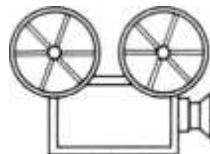


Fig. 9.1 Domains of VR application in manufacturing. Adapted from Fraunhofer Institute for Computer Graphics [1997]. Reprinted by permission.

9.1.1.1 Free-Form Design and Validation. This process occurs in the early stages of a new product creation. Designers typically use pencil and paper to sketch drawings, then place them on viewing boards so that colleagues can comment and provide feedback. The need to do tens of such sketches each day and the availability of computer-assisted industrial design systems has modernized the process. Instead of drawing on paper the designer can now create digital sketches using a stylus and a workbench-type display [Buxton et al., 2000]. This approach has the advantage that data are directly available online, but they are essentially 2D. An improvement is the use of display walls or CAVEs, where 3D models can be viewed collectively. This recreates the collaborative nature of a design studio and allows illumination adjustment in order to help judge the surface quality of a new car. Furthermore, the model can be rendered in a natural background (on the road, for example) helping gauge the styling effect of the new design.

In this example, the designer receives only visual feedback from the simulation. There are instances, such as sculpting to create a clay prototype, when haptic feedback plays an important role. SensAble Technologies has created an application called FreeForm in which designers sculpt "digital clay" using a PHANToM desktop interface [SensAble Technologies, 2002]. The software consists of a GUI that sits on top of the GHOST toolkit (discussed in Chapter 6). As shown in Figure 9.2a, the GUI is based on static and dynamic bar menus (dynabars), a representation of the sculpting tool, as well as a piece of digital clay. The hardness of the digital clay as well as the view to the scene can be changed using function keys.



VC 9.1

Various FreeForm tools can be selected from a toolbar menu, and are represented by 3D icons. These include a scraper for rough sculpting, a ball for drilling holes, a pencil, and so on. The pencil tool allows designers to draw contours on a sketch plane prior to pushing it through the digital clay block. Figure 9.2b shows the effect of a wire cut, where a rectangle and a

circle were drawn on the sketch plane (seen at the top of the clay block). Once the wire cut is executed, material outside of the circle and rectangular areas is removed, leaving a cylinder and a prism in place (a so-called outside wire cut). Besides wire cutting, FreeForm also allows stretching of the clay area ("tug" mode), addition of clay when too much was removed ("add clay" mode), surface smoothing ("smooth area" mode), and mirroring. During the "mirror all" mode, any carvings done on one side of the digital clay block are replicated on the other side of a selected symmetry plane. This assures correct symmetry as well as reduced sculpting time. Once the 3D model is finished, it can be viewed from any direction (much like a real sculpture), animated (using standard keyframe techniques), or exported to a milling machine to be carved out of aluminum or plastic.

FreeForm has been used successfully to create digital prototypes of products as varied as bicycle helmets, car seats, running shoes, and toys. As an example, Figure 9.2c shows the statue of Saint Fruition, done as a demonstration at the SIGGRAPH 2000 Conference [SensAble Technologies, 2000]. The same model was used to create small rubber toys given away to booth visitors as well as a video animation used in promotions. More importantly, the digital clay statue was used to create a 5-ft, 200-lb milled aluminum statue (shown in Fig. 9.2d). The FreeForm file containing the 3D model was first decimated (50%) to reduce complexity, then imported into the computer-aided manufacturing (CAM) package used for milling. The individual milling tool paths were generated directly from the FreeForm data, but the type of tool and milling machine had to be set manually. Eventually the aluminum statue was hand-finished to smooth out the cutter marks and areas of exposed skin.

This example illustrates the close collaboration that has to occur between designers and die manufacturers in order to create physical prototypes. Most of the time designing and manufacturing divisions of a company are not colocated, in which case designers have to travel to the manufacturing facility for approval of their free-form digital model. During approval the die manufacturer reviews the shape of the model for esthetic appearance and manufacturability (can it be done with the machines and processes

available?). This results in lost productivity, a problem that is compounded if mistakes are found and a new cycle of design and approval undertaken.

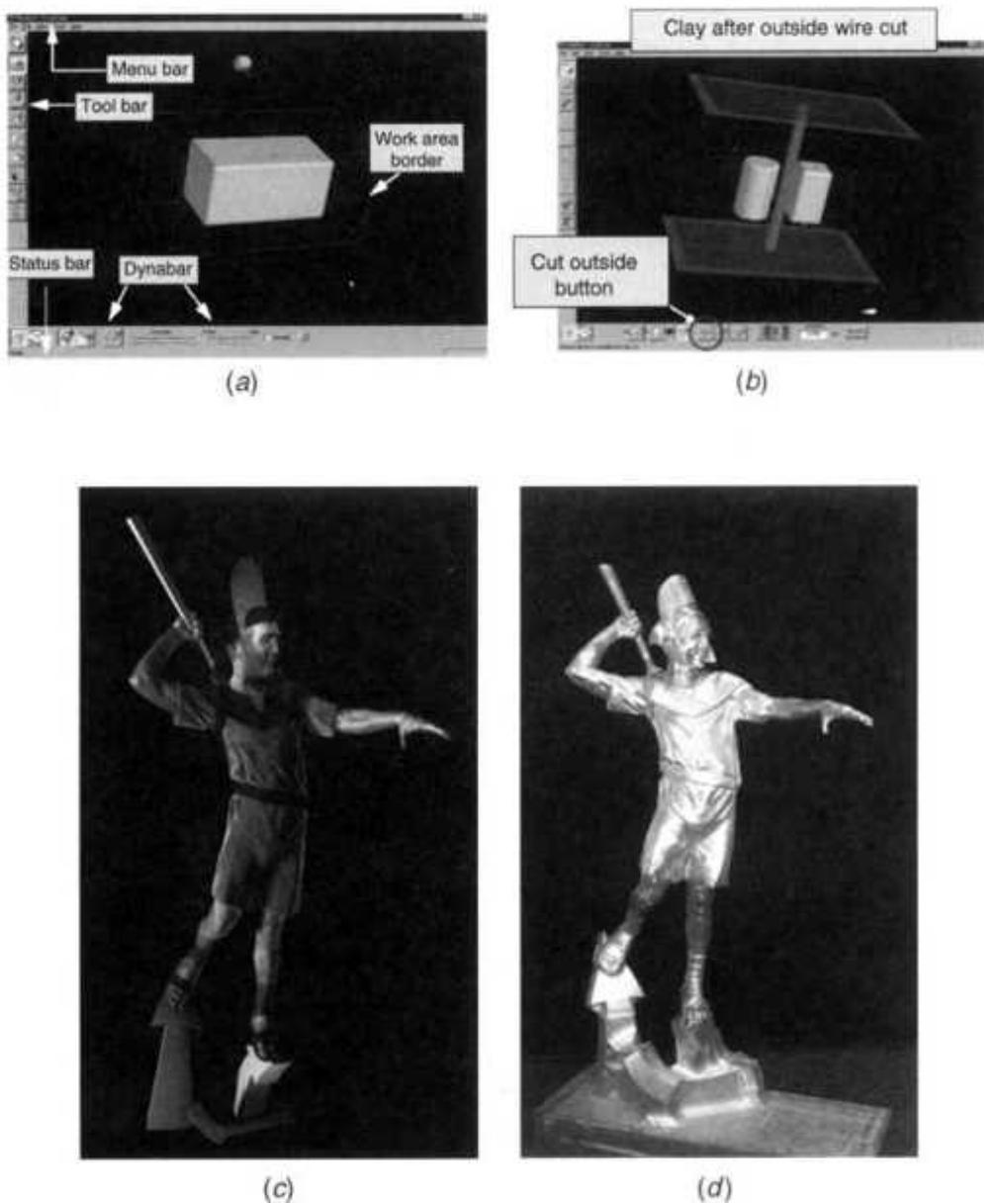


Fig. 9.2 FreeForm application for sculpting digital clay with a PHANToM interface. (a) Initial GUI showing menus, tools, and a block of digital clay; (b) digital clay after an outside wire cut. From SensAble Technologies [2002]. Reprinted by permission. (c) The digital clay statue of Saint Fruition; (d) its real counterpart milled out of aluminum. From SensAble Technologies [2000]. Reprinted by permission.

The availability of virtual prototype (CAD) files could in principle avoid travel if the data were sent to the manufacturing facility ahead of time and approval undertaken through teleconferencing. However, the die manufacturer's decision in this case is based solely on visual feedback, as there is no haptic interaction with the model. Kanai and his colleagues at Hokkaido University, Japan [Kanai et al., 1997], developed a prototype system for distant design review of free-form surfaces. The system consists of PCs at both locations linked through ISDN and Internet lines using TCP/IP communication. Both designer and die manufacturer interact with the freeform model using a haptic interface with three linear actuators (providing X, Y, and Z forces). This allows information on surface smoothness, compliance, ridges, etc., to be factored into the decision process. Position and force data from one site are transmitted to the other site over the ISDN line, while the Internet connection is used to transmit video images. The designer and die manufacturer take turns inspecting and modifying the model while the other looks on (only one user interacts at a time to avoid database conflicts). The system was tested successfully between Tokyo and Sapporo, which are 900 km apart.

9.1.1.2 Assembly Verification. This is another stage in prototype development whenever products are composed of multiple parts. This stage is needed to determine whether parts designed separately fit together correctly without unwanted interference (collisions). Another aim of assembly verification is to determine whether the assembly, once put together, can easily be taken apart. Disassembly is needed whenever the product has to be serviced, and ease of disassembly saves time (and money).

Jayaram and his colleagues at the University of Washington developed the Virtual Assembly Design Environment (VADE) to verify CAD-designed assemblies [Jayaram et al., 1999]. As shown in Figure 9.3a, the parts geometry and assembly information as well as their tolerances and physical attributes are imported into VR from the various CAD files. Once in VADE, the user, who wears an HMD, grabs the parts and assembles them in VR. Haptic feedback is provided by a sensing glove with touch feedback (the CyberTouch described in Chapter 3). During this process the part trajectory information as well as sequencing information are generated for later use by

actual robotic assembly stations. Furthermore, the simulation can be saved and played back for use in VR-based training, saving additional programming effort.

If errors are found (tolerances so tight that parts cannot mate, for example), then parametric design modifications can be done directly in VADE. As shown in Figure 9.3b, the designer uses 3D menus to change the critical dimensions of the part of interest (in this case the cylinder diameter). Another use of VADE is to check for collisions in the swept volumes generated by parts during the assembly sequence. Figure 9.3c shows an example of such a collision, indicating that the parts cannot fit in their current state. Suggested design changes are then sent by VADE back to the CAD environment and the process is repeated as needed.

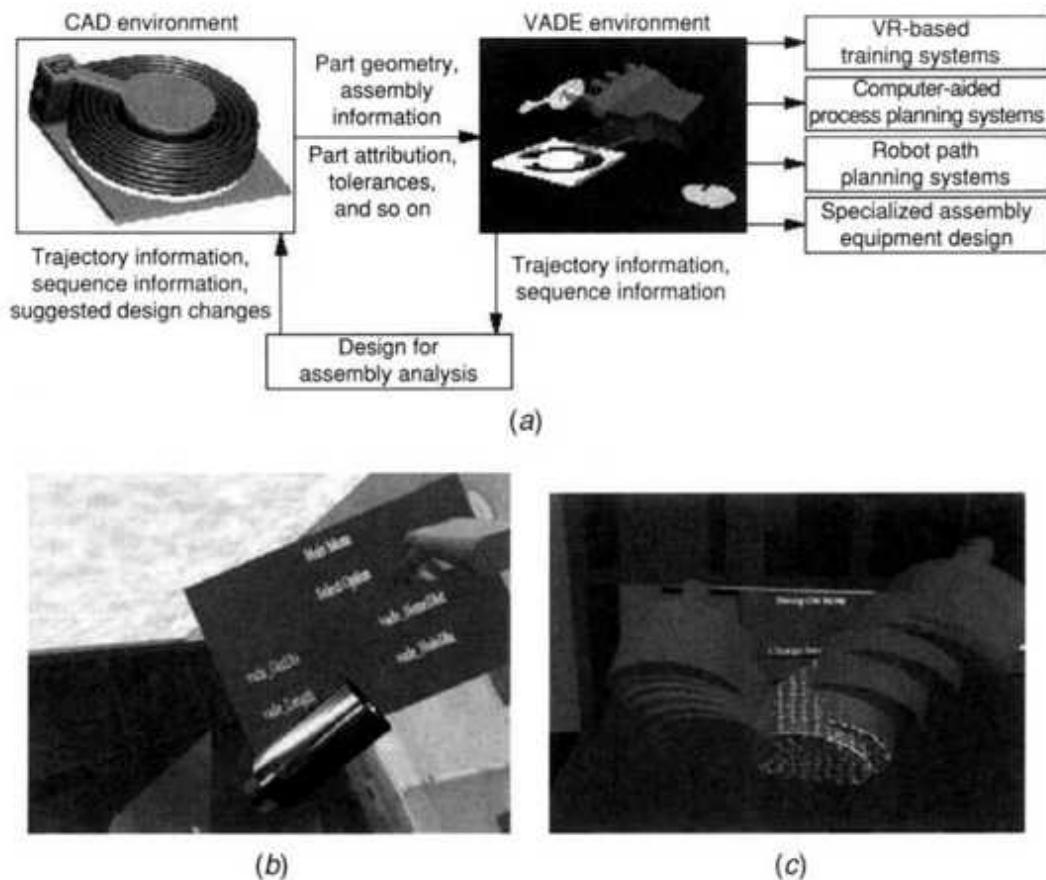


Fig. 9.3 Virtual assembly: (a) VADE usage scenario; (b) design modification in VR using 3D menus; (c) collision detection through swept volume

visualization. From Jayaram et al. [1999]. © 1999 IEEE. Reprinted by permission.

A design package with similar functionality to VADE is Dynamic Designer [Mechanical Dynamics Inc., 2002], which checks for interference while generating swept volumes. Unlike VADE, however, the simulation is partly immersive, such that no HMD or sensing gloves are used. This is due to the integration of Dynamic Designer within the Autodesk Inventor CAD authoring tool. Dynamic Designer, while less immersive, does provide additional functionality compared to VADE. It computes and plots velocity, acceleration, load, power consumption, kinetic energy, potential energy, and momentum. By calculating worst-case loading conditions Dynamic Designer is able to determine not only whether parts can be assembled/disassembled, but also whether the assembly, once realized, can withstand its envisioned usage conditions.

Assembly quality is important for car designers since customers are influenced by the esthetics of the car body. Parts making up the car exterior have varying tolerances and thus varying gap sizes. The tighter the tolerances, the better the assembly fit and the smoother the resulting car exterior. Unfortunately, tight tolerances result in increased production costs. Finding the best compromise is difficult based on CAD drawings alone and physical prototypes are costly and take a long time to build.

The flexibility of VR makes it ideal for creating "what if" simulations in which a range of tolerances is generated and the result analyzed for esthetic impact. This works as long as the simulation is photorealistic in order to provide designers with exact car surface light reflection characteristics. We know from Chapter 5 that global radiosity illumination is a way to create such photorealistic scenes. This approach was taken by researchers at the universities of Strathclyde and Leeds in the United Kingdom, who developed a virtual car body inspection room [Juster et al., 2001]. The room was developed based on digital photographs of a real inspection room and the modeling of various light sources at 20 geodesic viewpoints. As a case study, they used the Rover R75 prototype (developed by the BMW Group), which was rendered inside this virtual inspection room. The resulting virtual model (Figure 9.4a) closely replicates the metallic

shininess of its real counterpart (Fig. 9.4b). A simulation package developed for the project, called Visualization of the Impact of Tolerance Allocation (VITAL), allowed researchers to vary the assembly tolerances and repeatedly view the results in the virtual inspection room. A standard method for determining whether a (real) car body is smooth is to illuminate it with striped light. Figure 9.4c shows the virtual model being inspected with striped light. Discontinuities in the resulting lines indicate unacceptable tolerances, as is the case at the top portion of the Rover R75 fuel filler flap assembly. Tolerances for this assembly were found to be unacceptable and had to be tightened in order to reduce the gap.

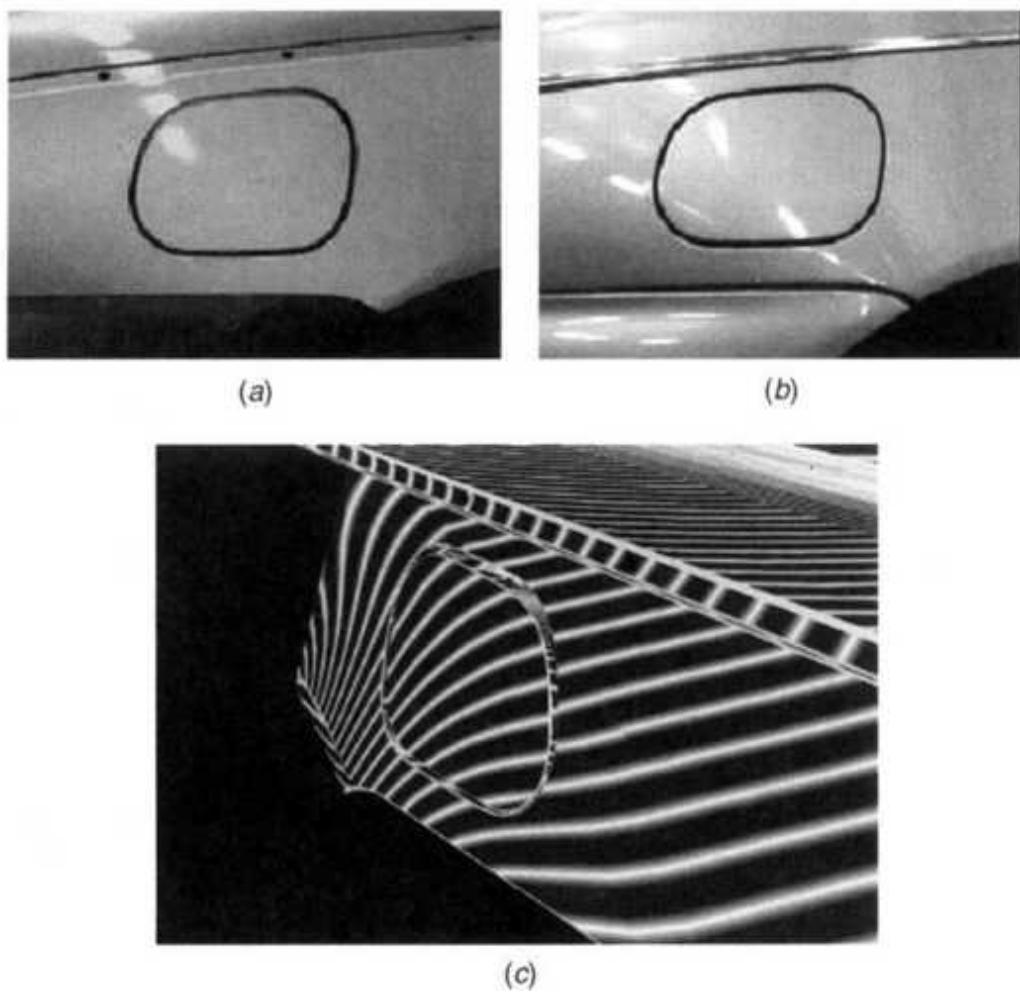


Fig. 9.4 Assembly verification of prototype car bodies: (a) virtual inspection room; (b) view from a real inspection room; (c) striped lighting highlighting unacceptable tolerances in the virtual model. From Juster et al. [2001]. © 2001 ASME. Reprinted by permission.

9.1.1.3 Ergonomic Analysis. This is usually done once a physical prototype exists in order to ascertain how easy to use (or service) the new product is. Virtual prototyping accelerates the process by allowing ergonomic studies at the earliest stages of design. This is possible due to several software tools that place humanoid models of specified sex, dimensions, body mass, even clothing, directly in the VR simulation. Complementing the graphics simulation is an imbedded computational engine that determines variables of interest such as reach envelope, joint torques, field of view, and so on.

Researchers at the Fraunhofer Institute for Industrial Engineering (Stuttgart, Germany) developed a real-time humanoid model called VirtualANTHROPOS and its companion ergonomic analysis tool, ERGONAUT [Deisinger et al., 2000]. As a case study, the researchers worked with John Deere Co. to study the ergonomics of their tractor cabins. Rather than relying on desk-top CAD-based ergonomic packages, they opted for a CAVE in order to perform full-scale simulations. The analysis was performed at three levels. The first, shown in Figure 9.5a, allowed the user to drive VirtualANTHROPOS in real time using a wireless tracking suit. This was done to determine the reaching work envelope as well as force iso- dynes (surfaces of equal loading) of the arms, depending on body dimensions and posture.

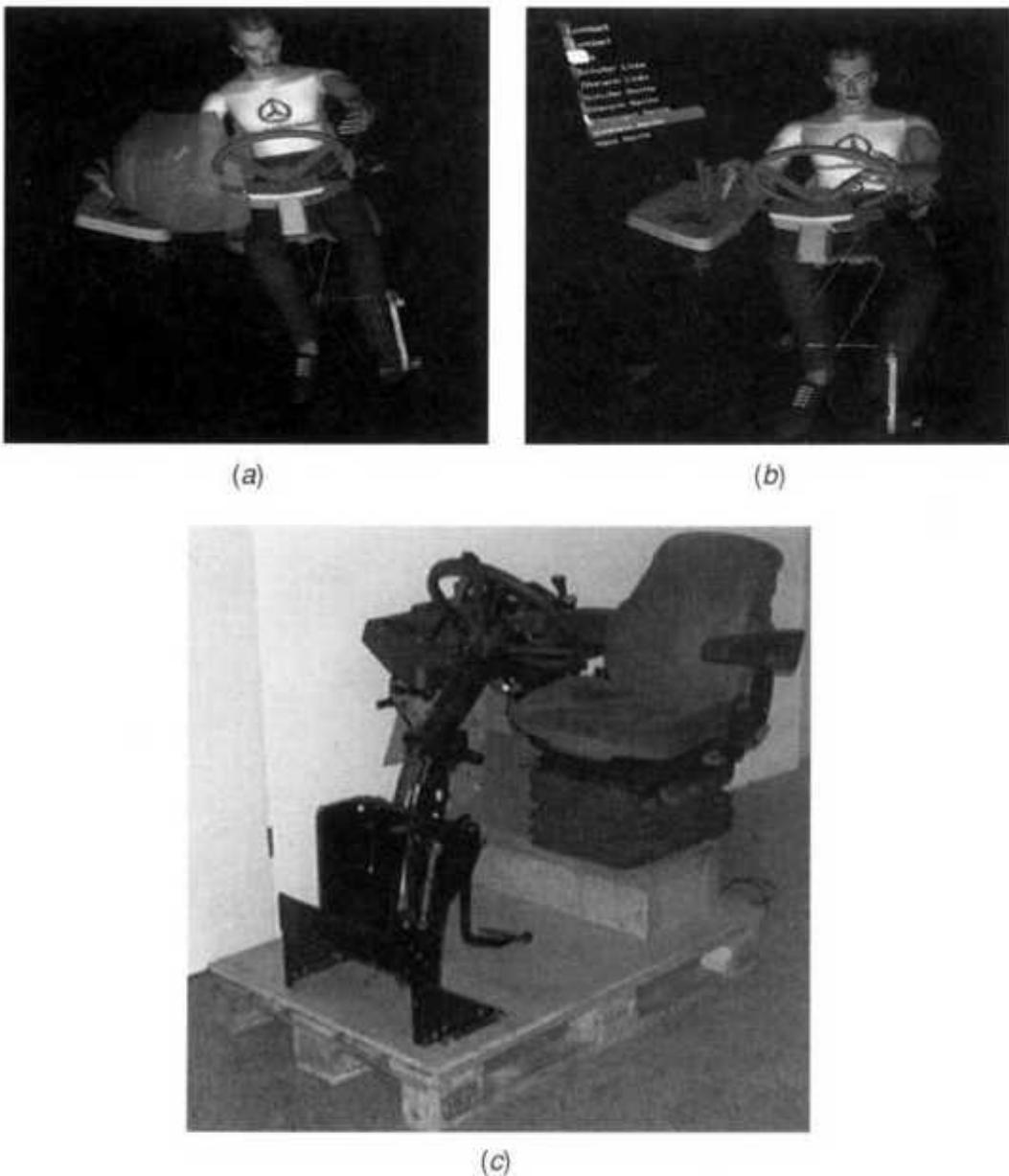


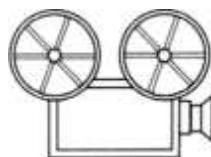
Fig. 9.5 Ergonomic analysis of a tractor cabin using VirtualANTHROPOS: (a) reach envelope; (b) joint discomfort level; (c) physical cabin mockup placed in a CAVE. From Deisinger et al. [2000]. Reprinted by permission of Fraunhofer IAO.

The second level of ergonomic analysis looked at body discomfort. During this stage the avatar was driven manually with a control box in order to move individual joints and determine their torques. As illustrated in Figure 9.5b, horizontal bar graphs showing discomfort levels in selected joints are overimposed on the virtual scene. The longer the bar, the higher is

the discomfort level (in this case corresponding to the right forearm, which is stressed).

The third level of analysis uses a physical mockup of the driver's cabin (seat, clutch pedal, steering wheel), as shown in Figure 9.5c. The control console is rendered graphically in order to change the position/orientation of its controls. Subsequently, subjects undertake testing in the CAVE while their movements are tracked. Using the ERGONAUT analysis tool it is possible to compute ergonomic variables (as previously described) and map them to the subjective feedback provided by the subjects. This allows intuitive generation of variants in the control design and fast design iterations. The combination of real objects with digital models leads to quantifiable and reproducible ergonomic assessments.

Another simulation package for ergonomic analysis is the Task Analysis Toolkit marketed by Electronic Data Systems (EDS) [2002]. It uses the humanoid model JACK@ originally developed at the University of Pennsylvania [Badler et al., 1993]. Unlike VirtualANTHROPOS, JACK has collision detection as well as a higher level of intelligent agent capabilities. Similarly, the Task Analysis Toolkit, illustrated in Figure 9.6, has more capabilities than ERGONAUT [Anon, 1999]. It computes the loading of the lower back and compares it with the National Institute of Safety and Health (NIOSH) norms. Another feature is the ability to compute the energy expenditure necessary to perform a task and consequently the state of worker fatigue.



VC 9.2, 9.3

Bombardier Ltd. was one of the first aircraft manufacturers to switch to fully digital models with their introduction of the Learjet 45 in 1996 [Short Brothers, 1998]. Although CAD/CAM manufacturing reduced the parts count by 20%, it could not detect access problems to the system area above the baggage bay. This is because their first-generation VR visualization of the

CAD data did not allow mechanism simulation (as described previously). The CAD models were monolithic, hinges and slides were not operable, and subassemblies could not be moved. As a consequence, an expensive redesign had to be undertaken to fix the problem. Subsequently, the company switched to a second-generation VR simulation that replicated mechanical assembly physical behavior. As a consequence, they could integrate the JACK ergonomic toolkit such that feasibility exercises could be done at the design stage. These studies were intended both for assembly operations and to train airline maintenance personnel, thus providing valuable input to the designers.

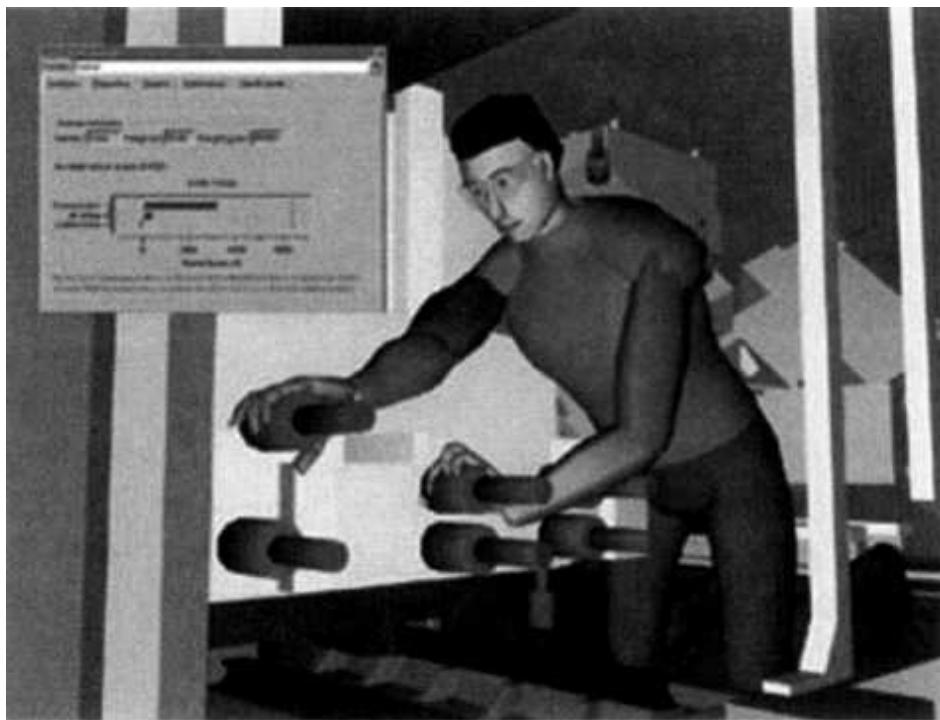


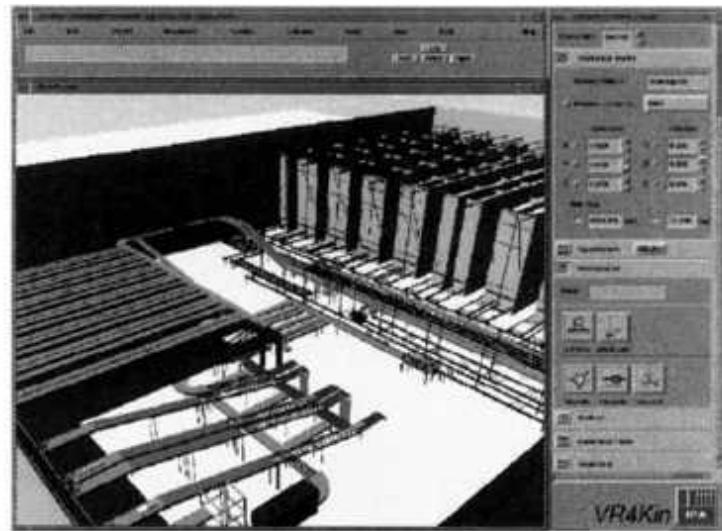
Fig. 9.6 A sample screen of the Task Analysis Toolkit. JACK® Human Simulation System. Image courtesy of Electronic Data Systems Co.

9.1.2 Other VR Applications in Manufacturing

Whereas virtual prototyping represents the oldest and largest area of VR use in manufacturing, it is by no means the only one. In fact VR is used throughout the manufacturing continuum, from plant design and construction to production, personnel training, and product marketing. These applications are discussed next.

9.1.2.1 Plant Design and Construction. This is a precursor to any large-scale production. Whereas small-batch (trial) runs can be done in design-oriented shops, large-quantity output of a given product (such as cars, airplanes, shoes, etc.) requires dedicated facilities. The planning and construction of such factories are areas where VR is starting to be used.

Factory planning has the important tasks of layout design (done by architects) and material flow simulation (done by industrial engineers), which grow in difficulty with the complexity of the product being made. Researchers at the Fraunhofer Institute in Germany developed a VR-based authoring environment combining object simulation with material flow simulation [Schraft et al., 1997]. The GUI, illustrated in Figure 9.7a, allows the designer to select various types of industrial equipment (in this case conveyors, pallets, and storage arrays) and imbed virtual low-level control and sensing equipment. The control equipment (programmable logic controllers) can be coded in the native assembly language and the code translated for use in the plant simulation. The reverse is also true, allowing programs verified in the VR simulation to be ported to the real work cells. This results in time savings as well as a reduced risk of flow bottlenecks. Additional advantages are increased transparency of the logistic processes in the plant, increased safety, and assessment of multiple variables and dynamic dependences. As a case study the researchers modeled a consumer product warehouse with 150 technical components, 250 sensors, and 350 information links. The high-complexity model (700,000 textured polygons) ran on an SGI Infinite Reality workstation at a frame refresh rate of 20 frames/sec.



(a)



(b)



(c)

Fig. 9.7 VR applications in plant design and construction. (a) GUI for simulation of large production facilities. From Schraft et al. [1997]. Reprinted by permission. (b, c) Visualization of building construction stages. From Davies [2002]. Reprinted by permission.

Once factory blueprints are drawn, a construction company needs to undertake the physical realization of the project. This involves not only interaction between different engineering specialties, but also frequent visits from nonprofessionals (such as the client corporation for which the factory is being built and the bank financing the project), who monitor progress. A survey of 11 construction companies in the United Kingdom and the United States [Whyte, 2001] found that VR was used both internally within the company and for interaction with nonprofessionals. The internal use of VR

involved coordination of detail design and construction scheduling. These were mostly symbolic visualizations where photorealism was not important, but the early detection of design flaws was. A classic example would be clash detection of heating, ventilation, and air conditioning pathways with electrical building pathways. Interaction with nonprofessionals used VR for demonstration of technical competence (at the bidding stage), visualization of construction scheduling (illustrated in Figs. 9.7b and 9.7c), and marketing. For such applications the virtual scene needs to be photorealistic and allow navigation in order to be easily understood by nonengineers. Another feature of such simulations is the ability to interrogate plant elements by clicking on them and retrieving (textual or synthetic voice) related information.

9.1.2.2 Training of Personnel and Actual Production. These are other activities that follow the product design stage. One of the oldest applications of VR in training was a pilot project undertaken by Adams Consulting for Motorola [Anon, 1994; Adams, 1996]. At the time Motorola was requiring every employee to take an annual course of their choice at their corporate Technology Education Center (TEC). Adams Consulting developed a VR simulation of the Pager Assembly Line, which was one of the training facilities at the TEC. This flexible robotic assembly line demanded regular worker training, which represented a major expense for Motorola. The main goal of the project was to ascertain the efficacy of the VR-based training (with and without an HMD) compared to standard training on the Pager Assembly Line. The simulation had textured graphics and realistic sound (sampled from the noise of real equipment), but no haptics. A total of 21 subjects, divided into three groups, participated in the study. Following familiarization with either the real or the virtual assembly line, each subject had to operate the line while monitored by an instructor. Experimental data showed that the immersive VR group had six times fewer errors on average than either the group trained on desk-top VR or on the real assembly line (1 vs. 6 errors). This result is even more impressive considering that the simulation was simplistic and ran on a 66-MHz PC with a frame refresh rate of at most 10 frames/sec.

Another type of task with significant training needs is maintenance of complex products, such as an airplane. This task has a tool/part manipulation component and an information retrieval (cognitive) component. The two components of a maintenance task are sequential. Thus a novice maintenance worker trying to identify the source of failure first reads related manuals, then proceeds to repairs. Neumann and Majoros [1998] reported that about half of the time is spent finding procedural information and that many of the procedural errors are due to negligence. The disastrous consequences of negligence in aircraft maintenance are obvious, thus there is a need to increase worker motivation and reduce time for information retrieval. Neumann and Majoros developed an automated system to help maintenance workers by placing task-related information directly in the scene using augmented reality. Apart from faster information retrieval, such a system helps novice workers memorize the task by enhancing associative memory. It is well known that humans memorize better by associating information with real-world locations. The system uses vision-based tracking to recognize the worker's view of the scene rendered on a see-through HMD. It then helps associative memory by consistently placing text information in relation to task objects, as illustrated in Figure 9.8 [Neumann and Majoros, 1998].

The figure presents a maintenance task (testing a solenoid valve) on a mockup of a C-17 military transport aircraft. Initially the system detects the pose (based on small circular markers placed on the outer aircraft surface) and labels the cover that needs to be removed (Fig. 9.8a). Subsequently, the training system detects the removal of the cover and labels related parts located underneath. It instructs the trainee to remove a central cover to set controls in the off position and "3. Remove cap" (Fig. 9.8b). The system then detects the removal of the cap (since the color in the scene changes again) and dynamically changes the content of the associated label to "4. Press to test" (Fig. 9.8c). If the test result (failure) requires additional information, then other areas of interest are highlighted ("Filter Bypass"-Fig. 9.8d).

One problem that may hamper the widespread introduction of VR on the factory floor is unwillingness of workers to cope with shortcomings in interface technology. Researchers at BMW and the Fraunhofer Institute for

Computer Graphics (Darmstadt, Germany) conducted a survey among 30 workers, engineers, and information technology specialists [Gomes de Sa and Zachmann, 1999]. The subjects had to perform a door lock assembly task in VR using a CyberTouch glove and voice input and received graphics feedback on a Virtual Research FS5 HMD. Both glove and HMD were tracked. The participants provided a subjective evaluation of the task interactivity and were uniform in liking the naturalness of gesture interaction and voice input. However, they complained of the lack of force feedback (they found touch feedback unnatural), which made the task difficult to complete. Additional problems were related to the limited tracking envelope (the test used Ascension trackers), the HMD weight, and the entangled tethers. The researchers concluded that wireless interfaces allowing large-volume tracking, force feedback, and better HMDs are needed before VR is really accepted as part of manufacturing training.

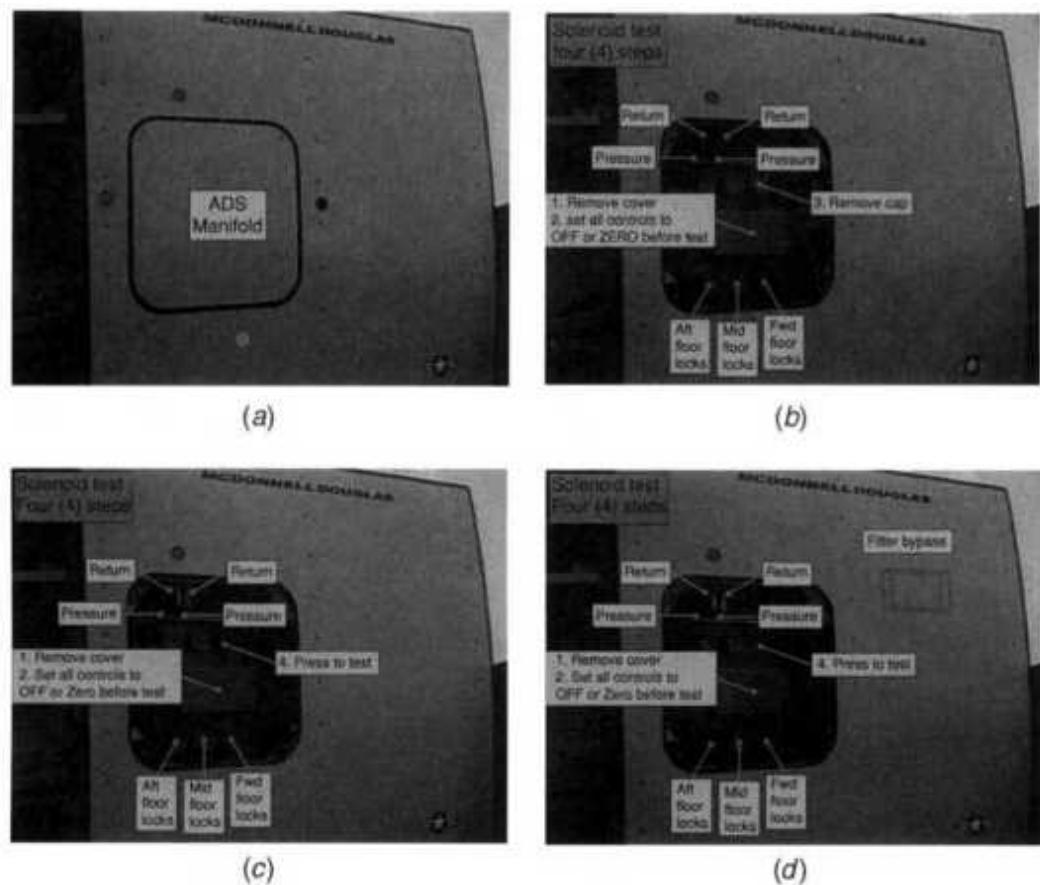


Fig. 9.8 Augmented reality use in C-17 aircraft maintenance training: (a) outer aircraft surface; (b) solenoid assembly cover removed; (c) dynamic text

labeling; (d) additional labels. From Neumann and Majoros [1998]. © 1998 IEEE. Reprinted by permission.

This lesson was learned by Boeing Co., which applies augmented reality in aircraft production to aid assembly workers route complex electrical wiring. The company fitted its workers with wearable PCs and tracked monocular HMDs at its plant in Everett, Washington [Nash, 1997]. The need for cognitive aids in this case stemmed from the fact that each aircraft has custom features, such that it is hard to remember exact specifications without proper documentation. A Boeing 747, for example, has about 1000 wire bundles (ranging in length from a few feet to more than 100 ft). Rather than using magnetic trackers (with their small tracking volume and tethers), Boeing's setup uses vision-based tracking and wireless transmission. Wiring data are thus retrieved from online databases and overlaid on video images of real wire assembly boards. This way each worker sees the assembly board and the correctly registered virtual wiring diagram, which he or she then replicates. Initial tests showed significant time savings (20-50% faster when using the AR system). Considering that airplanes cost \$100 million or more, this corresponds to significant cost savings.

9.1.2.3 Marketing. This is the last area of VR application in manufacturing to be discussed here. Many car dealerships have set up virtual showrooms on the Web allowing potential buyers to see customized versions of the car of their choice. Interactivity is limited, however, since the only feedback is 2D images (photographs). Peugeot/Citroen took the concept of the virtual showroom a step further by creating an immersive VR simulation depicting a 3D version of their Picasso model. As illustrated in Figure 9.9a, customers can visit one of over 30 European dealerships via tracked HMDs and interact with the new car model [AutoWeb.com.au, 1999]. They can open doors, rotate the car, and look at light reflection depending on specified body color. The interior of the virtual car can also be customized by changing seat cover color and trim and can then be visited (Fig. 9.9d).

Another use of VR in marketing is prediction of consumer purchasing behavior based on virtual shopping tours [Ipsos Reid, 2002]. The approach involves simulations of such environments as a drug store, a grocery store, or a car dealership in which subjects are asked to choose among competing

products. The researchers have total control over the environment and can thus change the store location, label, price, store interior (lighting), and so on. The buyer's choices are tabulated in order to ascertain the effects of the varying stimuli on purchasing decisions. This allows companies to test market products in VR well before they are placed in actual stores.

9.2 APPLICATIONS OF VR IN ROBOTICS

Robots are mechanical arms that perform physical actions on their environment. They are characterized by flexibility (versatility), as a given robot can execute a variety of tasks or execute a given task in a variety of ways. Finally, robots are intelligent or self-adapting, so they can react to changes in the environment and take corrective actions in order to perform their tasks successfully. In reality, we are far from achieving true artificial intelligence, which would allow a robot to execute tasks independently in almost any environment and with a minimum of human help.

Robotic manipulators with their programming flexibility are used on the factory floor (as previously discussed) as well as in medical, military, and other application areas, including VR. In fact there is a true synergy between robotics and VR [Burdea, 1999]. We saw in Chapter 3 how backdrivable manipulators (such as the PHANTOM arm) serve as haptic interfaces. Even general-purpose industrial robots such as the four-DOF Adept 1 have been used for robotic graphics [Gruenbaum et al., 1997]. In such an application the robot arm moves a control panel mockup to provide just-in-time haptics to a user wearing an HMD. Conversely, VR has been shown to be beneficial in several areas of robotics, such as in robot CAD design (as previously discussed), task programming, and teleoperation. These are discussed next.

9.2.1 Robot Programming

The reachable space (so-called work envelope) of a robot arm is limited by the range of its mechanical articulations (or joints). The robot may, in fact, damage itself if it attempts to move past its work envelope, and thus correct programming is important. The current practice is to set software limits on the goal points. These limits create a virtual work envelope, becoming

virtual limits. The robot manipulator can also be damaged if it hits obstacles within its work envelope. Normally, a robot relies on its sensors (vision, ultrasonic, lasers, proximity) to detect such obstacles and avoid them. Unfortunately, sensor data can be unreliable (or noisy, as discussed in Chapter 2). Therefore the software program generates artificial repelling forces based on external sensing data. The overall effect is that of enlarging the real obstacles by a buffer zone equal to the sensor uncertainty and creating virtual obstacles.

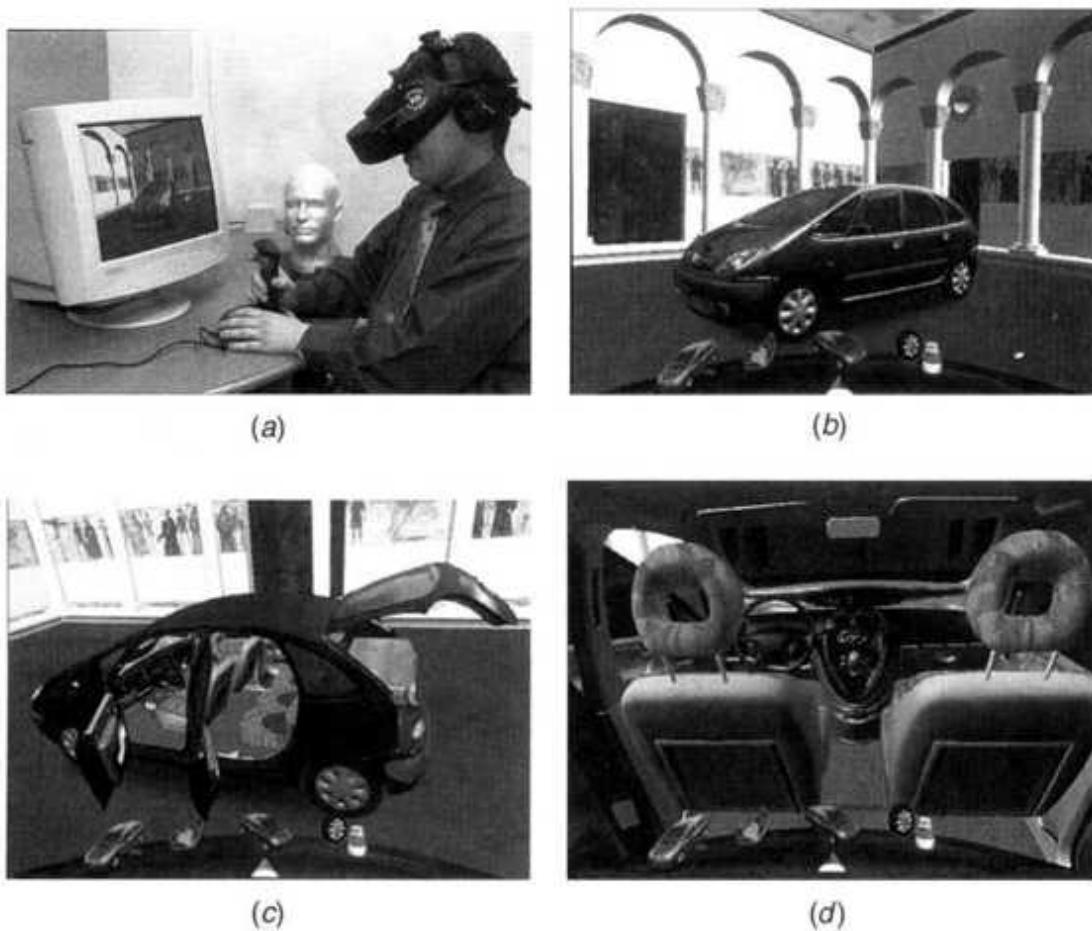


Fig. 9.9 Citroen virtual showroom: (a) potential buyer immersed in the simulation; (b) car exterior; (c) car exterior after the user opened all doors; (d) car interior after customization. Images courtesy of Citroen Co.

These examples show that robotic programming is a complex process. Industrial robot programming is performed manually (using a teach pendant), offline, or at "task level." Task-level programming instructs the robot what

needs to be done, but not how, and is an area of research. Offline programming creates software (usually in a robotic programming language) that subsequently needs to be tested running the real robot and requires good programming skills. Because of these issues, most industrial robots are programmed manually, using a simple teach pendant. The operator moves the manipulator slowly over its task trajectory by pressing buttons on the teach pendant and the required via points are stored in memory. This approach has the advantage of simplicity since it does not require programming skills.

Manual programming fails when the trajectory is complex. This is due to inertia effects that are absent when the robot is programmed, but are present when it moves at its normal high speed. Furthermore, teach pendants are ill suited for tasks where external sensors (of the type previously mentioned) condition alternative trajectories. Yanagihara and his colleagues at NTT (Japan) developed a multimodal teaching advisor (MTA) for use in seam welding of complicated car chassis [Yanagihara et al., 1996]. As shown in Figure 9.10, the system consists of a Mitsubishi PA10 robot (with seven DOF), a laser range finder (Fanet FLP-400), a human operator with the teach pendant, a seethrough I-glasses HMD, and a video tracking system with two CCD cameras. The MTA runs on a host PC, which receives voice commands from the operator and is interfaced with the other controllers. The MTA calculates the difference between the modeled welding path (based on task specifications) and that taught by the operator. Subsequently, it provides remedial advice to the operator through graphics and audio feedback. Tests showed that the use of the MTA produced better taught trajectories even for the novice operators, while the duration of the teaching time was reduced.

Offline programming is more suited for sensor-intensive tasks, but robotic languages are dependent on the particular manipulator used, and the debugging stage is quite time-consuming [Rehg, 1997]. Due to the various robot and task modeling errors, it is necessary to use the real robot in the program debugging phase. Therefore the robot has to be taken out of the production line for up to 30 hours. This represents an unacceptable loss of revenue for industries relying heavily on robotics, such as car manufacturing. Modern programming techniques using graphics simulators

have reduced the number of hours required to take the robot offline to about 6 hours. This is still large for what industry needs.

One project attempting to advance the state of the art in robot programming has been ongoing at the Fraunhofer Institute for Manufacturing Engineering and Automation in Stuttgart, Germany [Strommer et al., 1993]. As illustrated in Figure 9.11, programming is done on a virtual robot, with the programmer's interaction being mediated by VR I/O devices (sensing gloves, tracker, HMD). Thus the robot programmer is immersed in the VR simulation and can navigate and look at the assembly scene from any angle. In this way the programmer can see details that may not be visible in real life due to occlusions or scaling factors. In VR, trajectories can be specified naturally through hand gestures rather than by pushing buttons (manual programming) or writing text (offline programming). The code is automatically stored using a special-purpose toolkit called VR4 [Flaig et al., 1996]. This toolkit allows the generation of graphics at high frame refresh rates using variable level of detail (as discussed in Chapter 5). Collision detection between the robot and other virtual objects is optimized in a hierarchical way (bounding-box checking being done first). VR4 also has a database of preexisting models as well as a GUI for custom-made configurations [Fraunhofer Institute for Manufacturing Engineering and Automation, 1996]. This GUI allows the user to specify the dynamic behavior of components such as their paths, accelerations, velocities, interpolations, etc. Once the program is completed, it is downloaded to a real robot connected to the same VR engine and the programmed task is executed. Feedback from the sensors on the real robot is then used to fine tune the program.

9.2.2 Robot Teleoperation

The difficulties related to machine reasoning and unknown environments has led to the development of two types of robots, for the industrial and service sectors, respectively. Industrial robots perform tasks automatically, but in completely known (or structured) environments. Service robots, such as those used in space exploration, perform tasks in unstructured environments, but under the remote (or teleoperation) control of a human. This is necessary

in order to compensate for the inadequacies of the automatic control loop. There is another reason why robots may need to be controlled at a distance, namely when the remote environment is dangerous (explosive disposal sites or battlefields) or unfriendly to humans (outer space, undersea, nuclear sites, etc.).

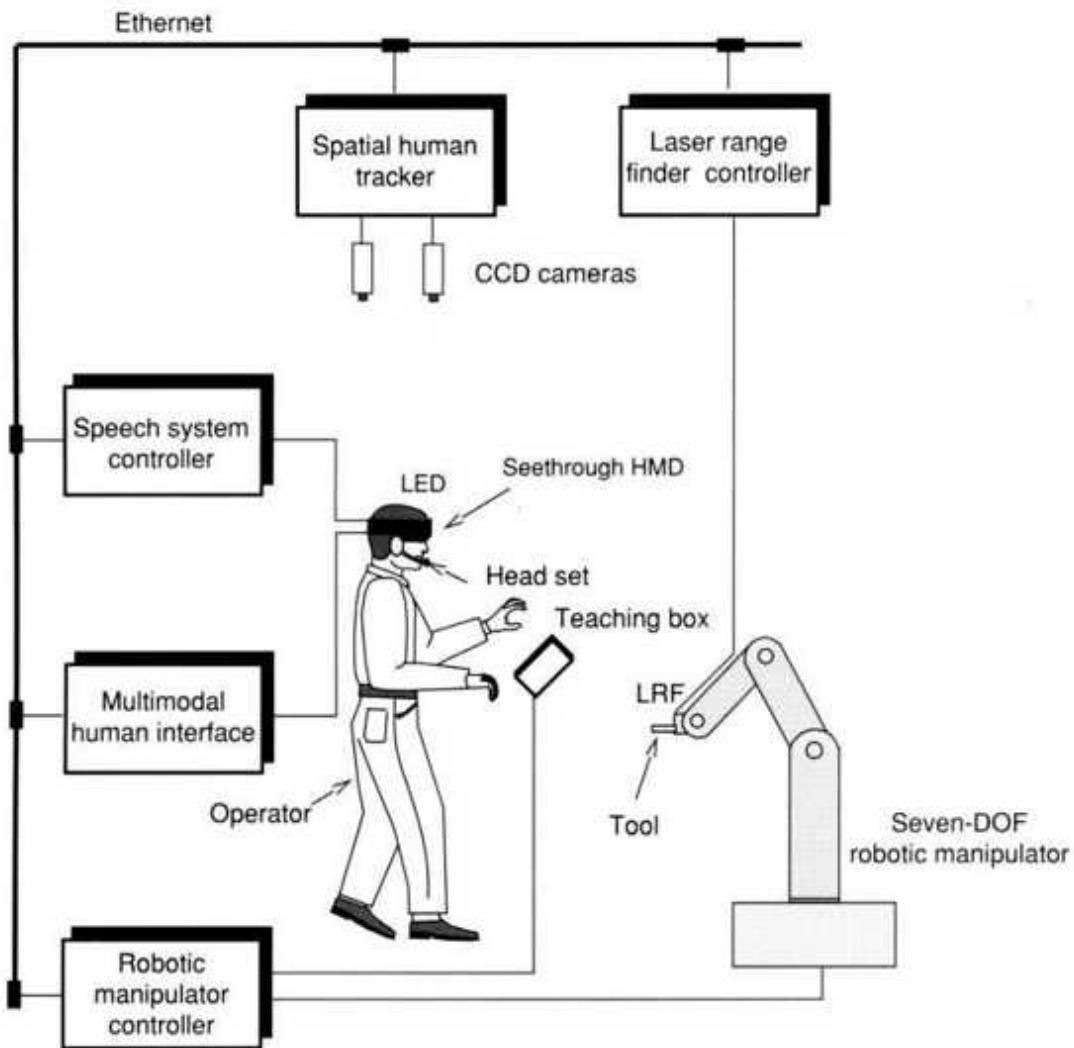


Fig. 9.10 Teach pendant-based robot programming using a multimodal teaching advisor. From Yanagihara et al. [1996]. © 1996 IEEE. Reprinted by permission.

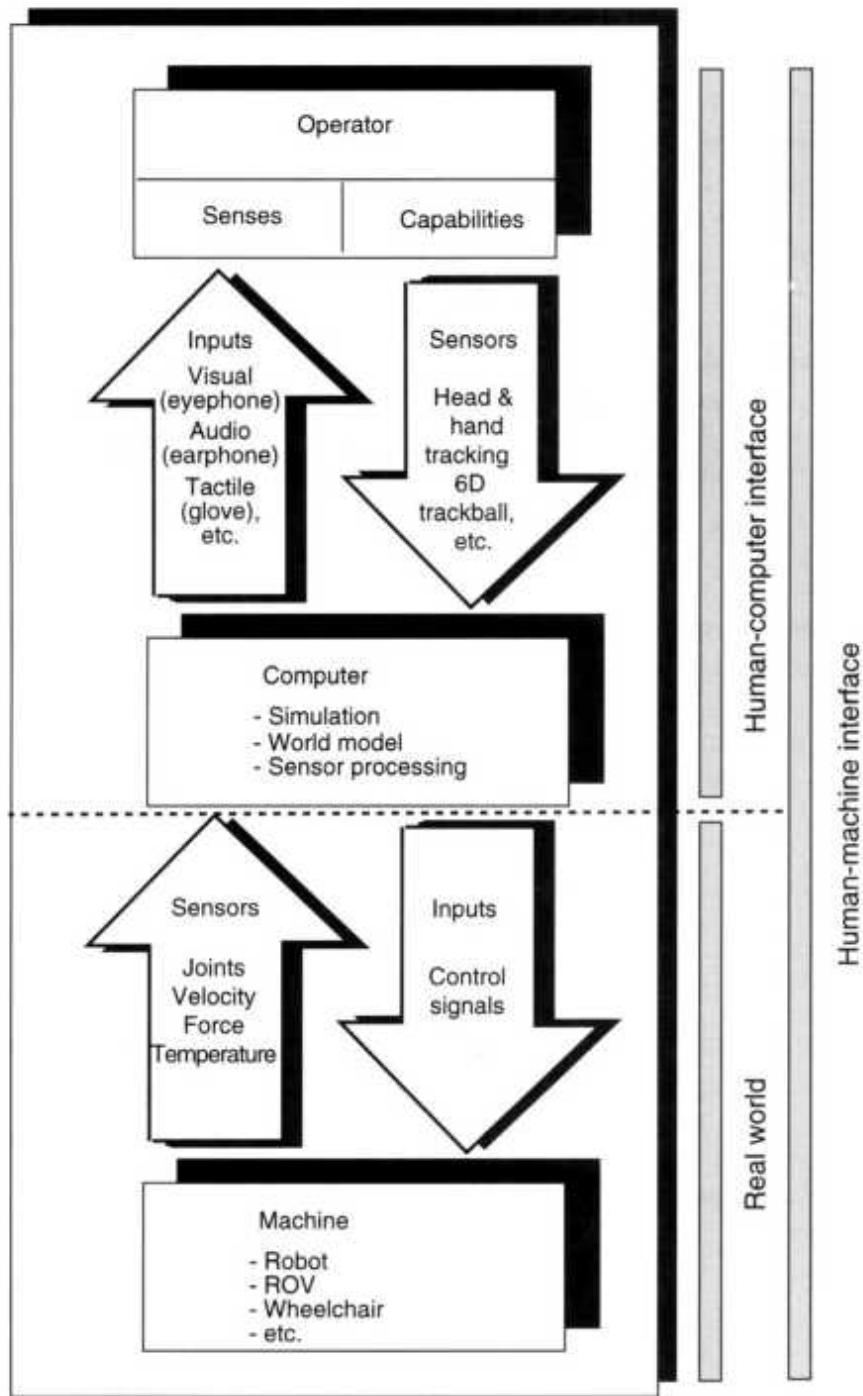


Fig. 9.11 VR-aided robot offline programming. Adapted from Strommer et al. [1993]. Reprinted by permission.

Whenever a robot is controlled at a distance, the human operator acts on a master arm inputting commands (position, velocity, or force). These commands are then sent to the slave robot, which executes the task and feeds

back information to the operator (such as video images of the remote site). If the commands are transmitted in real time, faithfully executed, and feedback from the remote task correctly presented to the user, then the teleoperation system is said to be transparent [Kheddar et al., 2002]. A transparent system coupled with rich feedback (video, sound, force) makes the user feel immersed in the (real) remote environment, much like users feel immersed in virtual environments. In fact virtual environments can be added to real ones (for operator training or to help actual teleoperation), as is discussed next.

9.2.2.1 Teleoperation with Corrupted Visual Feedback. Corrupted data due to poor illumination, smoke, or visual occlusions of the remote camera can make teleoperation very difficult to execute. If elements of the remote environment are known (such as pipe systems in a power plant), then they can be modeled ahead of time in VR. When the visual data become corrupted, the operator can switch to the VR model of the task and teleoperate a virtual replica of the remote robot. Data are then sent by the local workstation to the remote robot computer for execution.

An instance when visual feedback is corrupted is in the teleoperation of a firefighting robot. Figure 9.12a shows the video images of a pipe structure engulfed in thick smoke [Oyama et al., 1993]. The valve handle that has to be turned by the robotic arm is indeed very difficult to discern. Researchers at the Ministry of International Trade and Industry (MITI) (Japan) developed a system that uses a VR model of the firefighting robot to execute this difficult task (Figure 9.12b). Key to the success of their approach is the accurate calibration between the local (virtual) scene and the remote real one. The researchers manually selected corresponding points in the VR and remote scene and then used a least-squares algorithm to calibrate the system. Since only one camera is used, the errors are on the order of 3 cm in the axial view direction, such that only low-accuracy tasks can be executed. A better approach, subsequently developed by NASA researchers, is to use multiple views (from multiple cameras) and the addition of a nonlinear calibration algorithm. This results in a positional discrepancy of (only) 1 cm [Kim, 1996], better suited for low-orbit satellite servicing tasks.

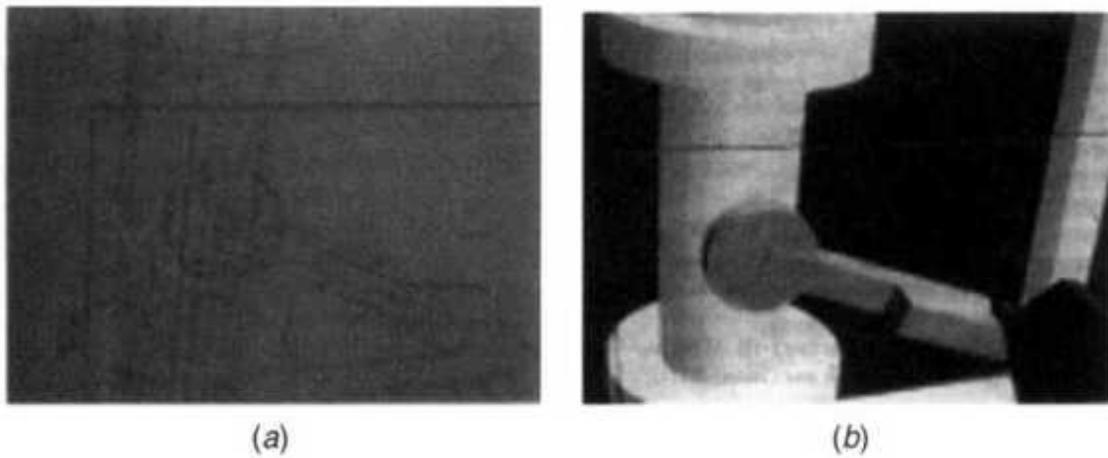


Fig. 9.12 VR-enhanced teleoperation of a firefighting robot: (a) poor visual feedback from the remote site; (b) VR model of the remote robot. From Oyama et al. [1993]. © 1993 Massachusetts Institute of Technology. Reprinted by permission.

9.2.2.2 Teleoperation with Time Delays. This occurs when the signal from the master to the slave robot has to be transmitted over long distances. A typical example is robot operations in outer space, where time delays are a few seconds or more. In such instances the operator is forced to move the master small distances, then wait for the video from the remote site to confirm that the small motion is finished, and so on. Small motions are needed to avoid collisions with the environment. This move-and-wait strategy unfortunately results in unacceptably long times to complete the task through teleoperation.

An improvement to teleoperation with time delays was the introduction of the "phantom robot," an accurate 3D model of the remote robot, which is controlled locally by the master [Bejczy et al., 1990]. Unlike the remote robot, its virtual counterpart responds instantaneously to the operator's commands. When overlaid on the video feedback and correctly registered to the remote robot, the phantom robot serves as a predictor display. It can be moved smoothly and safely to the desired location and the motion stored in the local computer. Once the operator is satisfied with the results, the corresponding trajectory is sent to the remote robot for execution. Figure 9.13a [Kim, 1996] shows a phantom robot (rendered in wire frame) overlaid on the image of a space repair task. Once the command

for motion is sent to the robot, it will go to the position of the phantom robot (as seen in Fig. 9.13b). While this still is a move-and-wait strategy, the tests done for NASA (Jet Propulsion Laboratory, Pasadena, CA) showed a reduction in task completion time by 50% when the phantom robot was used.

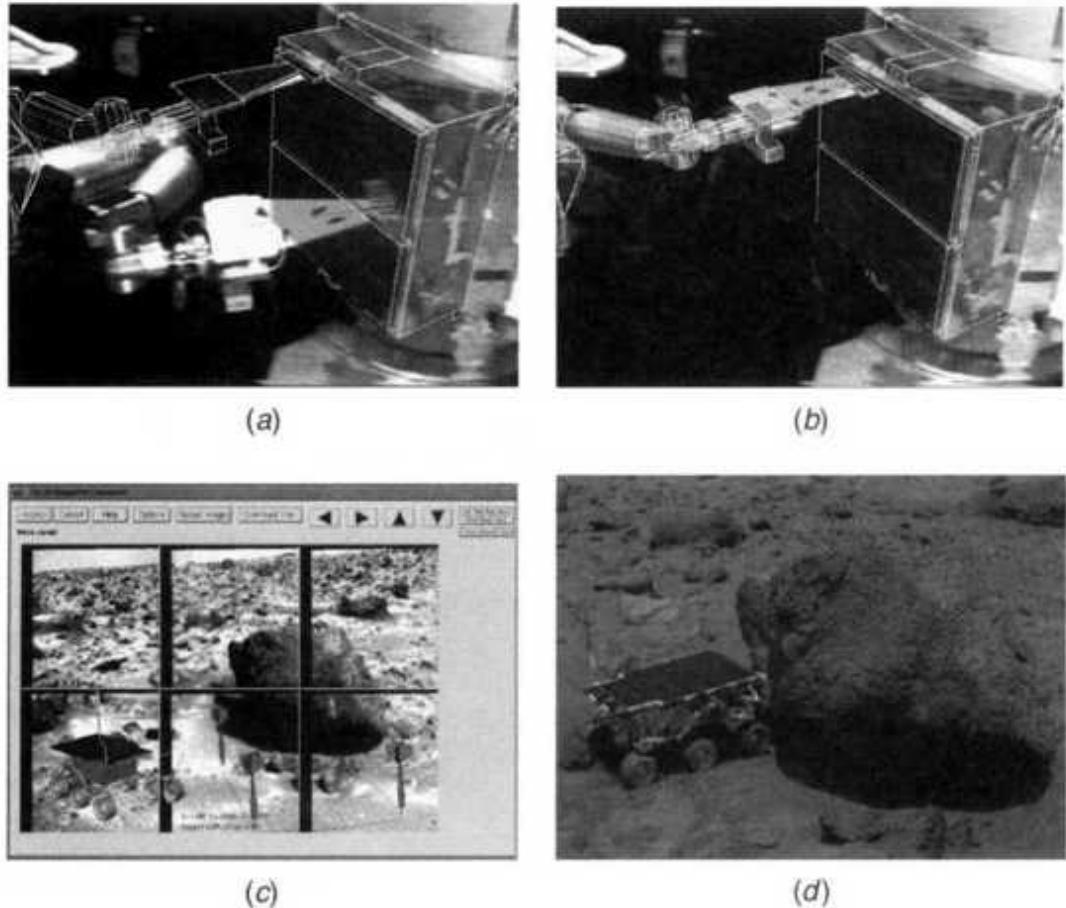
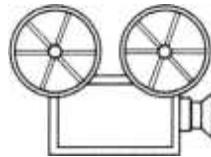


Fig. 9.13 Space teleoperation with time delays. (a, h) Phantom robot used in low-orbit repairs. From Kim [1996]. © 1996 Massachusetts Institute of Technology. Reprinted by permission. (c, d) Use of VR-based teleprogramming for operation of a rover on Mars. From Jet Propulsion Laboratory [2002]. Reprinted by permission.

A dramatic example of the benefits of VR for teleoperation with long time delays is the Mars Pathfinder mission, which occurred in 1997 [Jet Propulsion Laboratory, 2002]. Scientists at the Jet Propulsion Laboratory in California had to teleoperate a rover on the surface of Mars despite a one-way time delay of about 20 minutes. To do so they first recreated a model of

the planet surface using images from previous missions. They also created a virtual model of the rover, which was inserted in the Mars virtual scenery, as seen in Figure 9.13c. Instead of sending low-level motion commands, the scientists used a teleprogramming approach, sending only macro commands. These commands instructed the rover to perform tasks such as image taking, position calibration versus the Sun, travel to a way point, etc. Graphics programming metaphors were used to develop a safe path over the uneven Mars surface and preview resulting motion. Subsequently such paths were sent to the rover on Mars for execution (Fig. 9.13d).



VC 9.4

Time delay not only affects teleoperation task completion time, it also impacts its stability when force feedback is used. Time delays as small as 0.1 sec can make force feedback detrimental rather than beneficial for teleoperation. Kotoku [1992] at MITI (Japan) extended the concept of the phantom robot to haptics. As illustrated in Figure 9.14, the operator controls a master arm and the position commands arrive at the remote slave robot with a communication delay (in this case 0.5 sec). The master input is also used to instantaneously move a virtual model of the slave robot. Simulation contact forces during virtual slave interactions are instantaneously fed back to the operator through the master robot. Human factors studies were conducted on a simple (planar) task of tracing a rigid barrier while pushing with a constant force. A simplified haptic model neglected friction and considered the remote slave as a point object. Subjects carried out the task observing the virtual slave arm (and its modeled contact forces) on the graphics display and the real slave on a video monitor. The master arm trajectory was sent to the remote slave arm with a 0.5-sec communication delay. Contact forces were fed back to the subjects in half of the trials. Results showed that the addition of force feedback produced a more stable control. Furthermore, the movement of the slave robot was three times faster when force feedback was present.

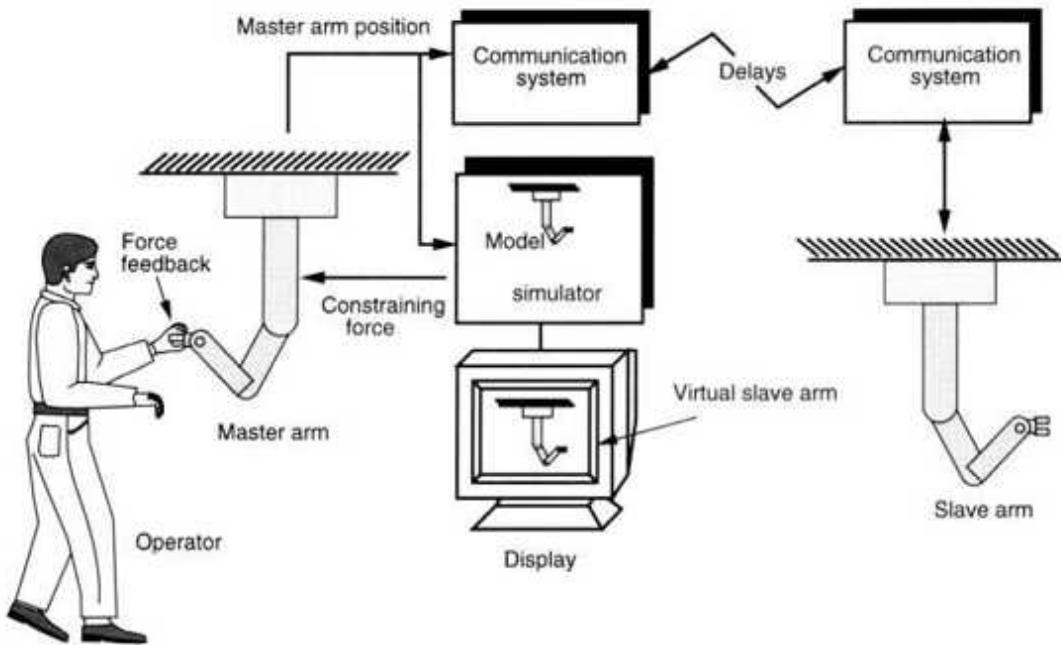


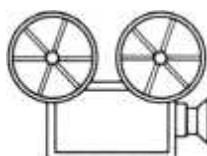
Fig. 9.14 Telerobotic system using a virtual slave robot to overcome force feedback instabilities due to communication delays. Adapted from Kotoku [1992]. © 1992 IEEE. Reprinted by permission.

A more realistic approach for solving the problem of teleoperation with time delay and force feedback was taken by Rosenberg at Stanford University [Rosenberg, 1993], who proposed to use virtual fixtures, defined as abstract sensorial data overlaid on top of the remote workspace and only interacting with the operator. Thus virtual fixtures can occupy the same physical space as objects in the workspace without geometrical or physical constraints. In a peg-in-hole insertion task done through teleoperation [Rosenberg, 1994], the movement time increased as much as 45% when no virtual fixtures were present and the time delay was 450 msec. The virtual fixtures provided guidance (or enhanced localization), which had a beneficial effect by reducing this difference to only 3%.

9.2.2.3 Supervisory Control. In this modern teleoperation technique the remote robot is controlled indirectly, using an intermediate graphics layer. In classic teleoperation the operator needs to be familiar with the slave robot characteristics and system transparency is reduced due to nonnatural interactions. Whereas gestures are a natural interaction modality, they are

not understood in classic teleoperation (which uses joysticks, trackballs, or full-size replicas of the remote robot). VR simulations have successfully used gesture input (as seen in Chapter 2) and can make teleoperation more intuitive for the operator.

Researchers at the Institute for Robotics Research (Dortmund, Germany) have developed a VR-based flexible supervisory teleoperation environment. As seen in Figure 9.15a, the operator interacts through gestures (sampled by a sensing glove) and receives stereographic feedback from the simulation. The intermediate control layer consists of a virtual representation of the remote scene, but without showing the actual robot. In this environment the operator is represented by an avatar, who follows the user's input (Fig. 9.15b). An actual robot in the remote environment receives commands from the simulation and acts on its environment (based on local sensing) (Fig. 9.15c). If, for example, the task is visual inspection of a piece of equipment at the remote site, the robot picks up a real camera. The simulation then overlays real video images on a virtual view finder (or TV) manipulated by the avatar (Fig. 9.15d) and seen by the user in the virtual scene. In this way the operator feels as if he or she is inspecting the equipment directly, being totally unaware of the actual robot doing the job. It is as if the robot extends (or projects) the operator's action from the virtual to the real domain, in what the researchers call "projective virtual reality" [Institute for Robotics Research, 2002].



VC 9.5, 9.6

In this example the actual robot is hidden from view (although a wireframe model can be introduced in the virtual scene when requested). The concept of using a hidden robot for supervisory control was pioneered by researchers at the Robotics Laboratory of Paris [Kheddar et al., 1997] with the aim of increasing teleoperation intuitiveness and flexibility. Since the robot details are hidden, the same VR simulation can be mapped to several, dissimilar slave robots without any change at the operator's site.

This idea was taken by the French researchers a step further by applying the principle of one-to-many teleoperation. This means that a single operator can simultaneously supervise several remote robots, which execute the same task. As illustrated in Figure 9.16, the research team successfully tested this multiplexed teleoperation approach between France and Japan [Kheddar et al., 1998]. The task consisted in the assembly of planar pieces of a puzzle using dissimilar industrial robots. The operator viewed a virtual scene on a CRT that depicted the puzzle pieces and a virtual hand. He then controlled the virtual hand using a prototype haptic glove (the LRP glove [Bouzit et al., 1993]). A range of commands (some at higher level) were then sent to the remote robots for execution. Visual feedback from the real remote sites was provided only for safety and recovery purposes.

9.3 INFORMATION VISUALIZATION

The last application area to be discussed is information visualization using virtual environments. Within this context information visualization represents the transformation of abstract data, normally shown as 2D graphs or spreadsheets, into 3D geometry. Such geometry can be interactively manipulated by the user immersed into the simulation, with obvious cognitive benefits.

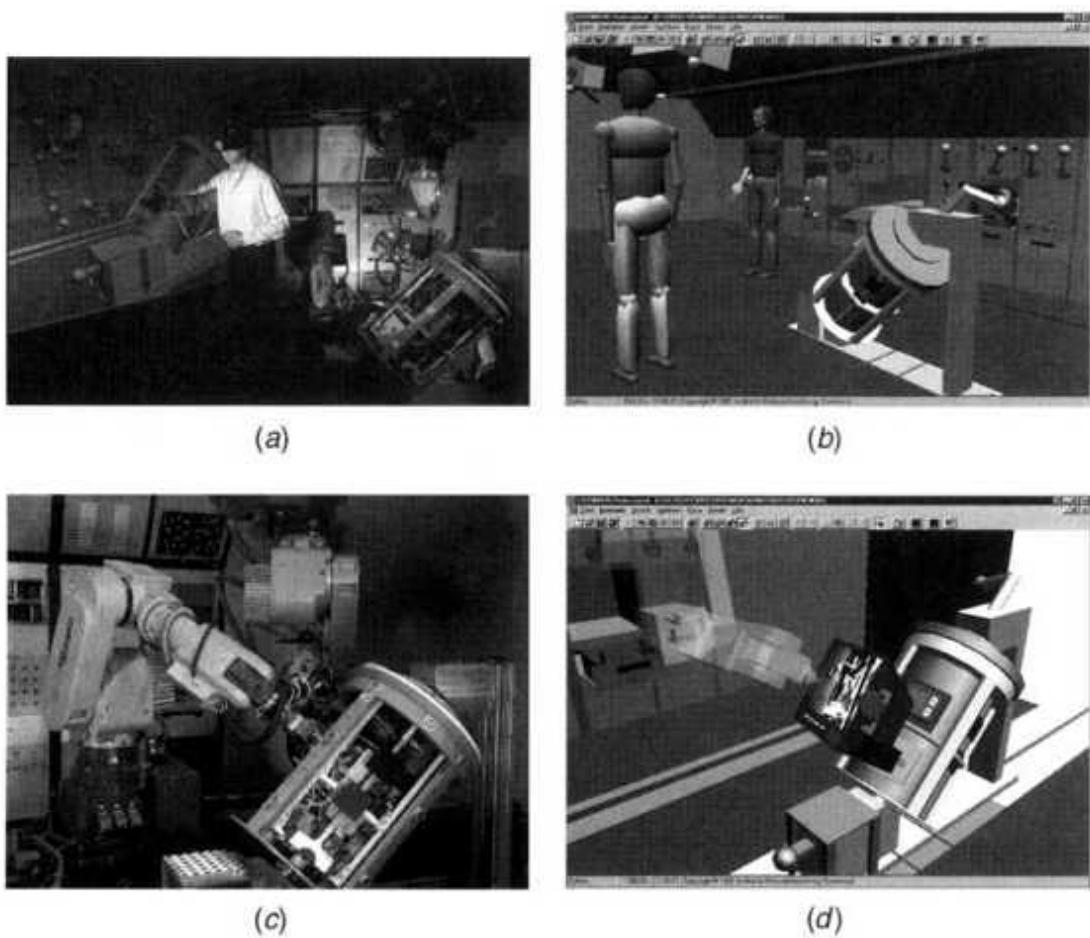


Fig. 9.15 VR-based supervisory teleoperation: (a) operator station; (b) virtual scene showing the remote site; (c) slave robot responding to the avatar's actions; (d) real video overlaid on the virtual scene during an inspection task. Image Courtesy of the Institute for Robotics Research.

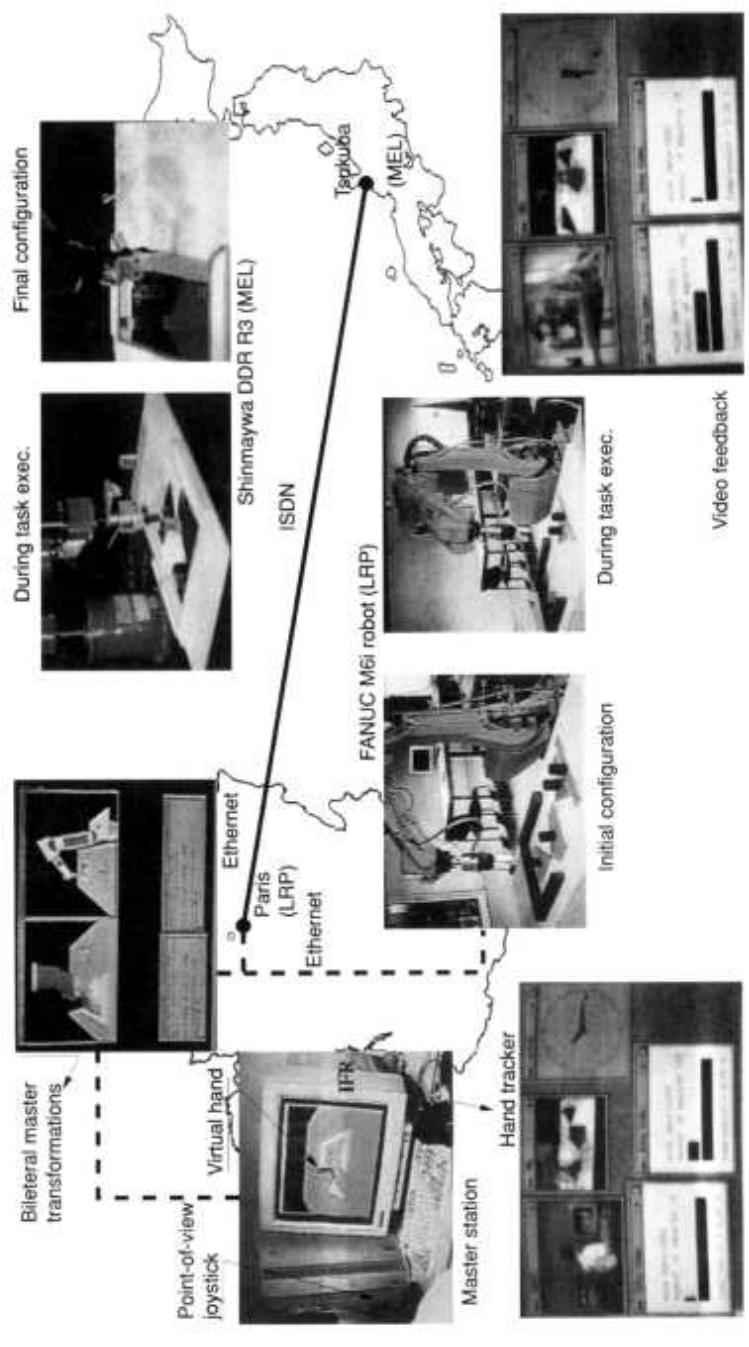


Fig. 9.16 Multiplexed teleoperation using the hidden robot concept. From Kheddar et al. [1998].
 © 1998 IEEE. Reprinted by permission.

Figure 9.17 is a block diagram of a generic VR information visualization system (adapted from [Bryson, 2002]). The user controls input devices (such as trackballs or sensing gloves), which allow him or her to specify the view to the visualized data as well as an area of particular interest. This input is sent to the data extraction processes, which convert data into 3D geometry

and sends them to be rendered and fed back (visually) to the user. The time constraints set by simulation interactivity (latencies of less than 100 msec) cannot be met by the data extraction process owing to the large computation load involved. In order to maintain interactivity, the two loops (I/O and data extraction) need to be asynchronous. This is reminiscent of the asynchronous communication between the graphics and haptics rendering pipelines discussed in Chapter 4. Thus the output from the input device/process is written in a buffer, which is read by the data extraction hardware when a new visualization loop is started (suggested by Bryson to be every 0.3 sec). In order to maintain interactivity, the input device data are also sent to the display hardware without delay.

A special case is represented by data that are time-varying. In this case the computation load increases even more since the data extraction needs to take into account the time step. The user time (associated with input and real-time graphics output) will in most cases be different from the time clock used to visualize time-dependent variables. It is in fact possible to slow down or to accelerate data visualization time (for example, the growth of a plant that would otherwise take weeks or months to complete).

In order to maintain reasonable interactivity and short data extraction time, the computation load needs to be distributed. Thus information visualization systems consist of either multiprocessor workstations (local load distribution) or networked computers (of which at least one is in charge of graphics rendering and user I/O). The advantage in using network-based load distribution is the ability to visualize data in a multiuser simulation. Such shared VEs satisfy the needs of multidisciplinary, geographically separated teams of experts that have to interpret complex data. One domain where multidisciplinary teams routinely use information visualization collaboratively is oil and gas exploration. This is discussed next, followed by VR-based visualization of volumetric (mostly medical) data.

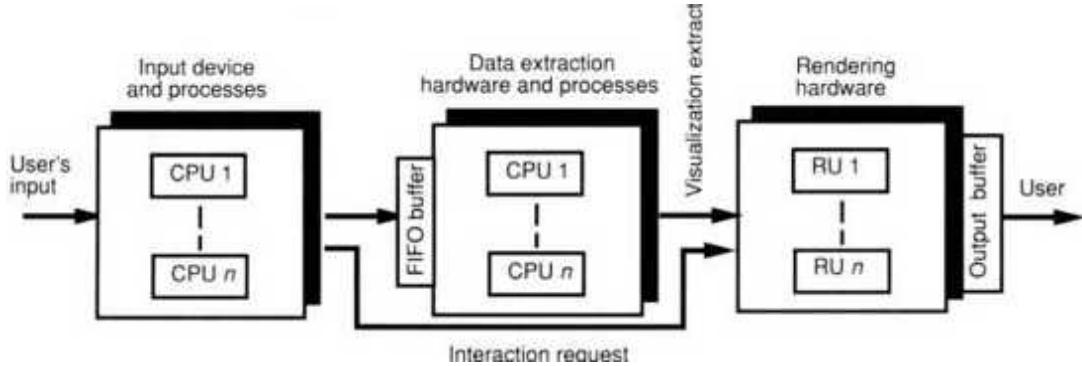


Fig. 9.17 Information visualization pipeline in VR. Adapted from Bryson [2002]. Reprinted by permission.

9.3.1 Oil Exploration and Well Management

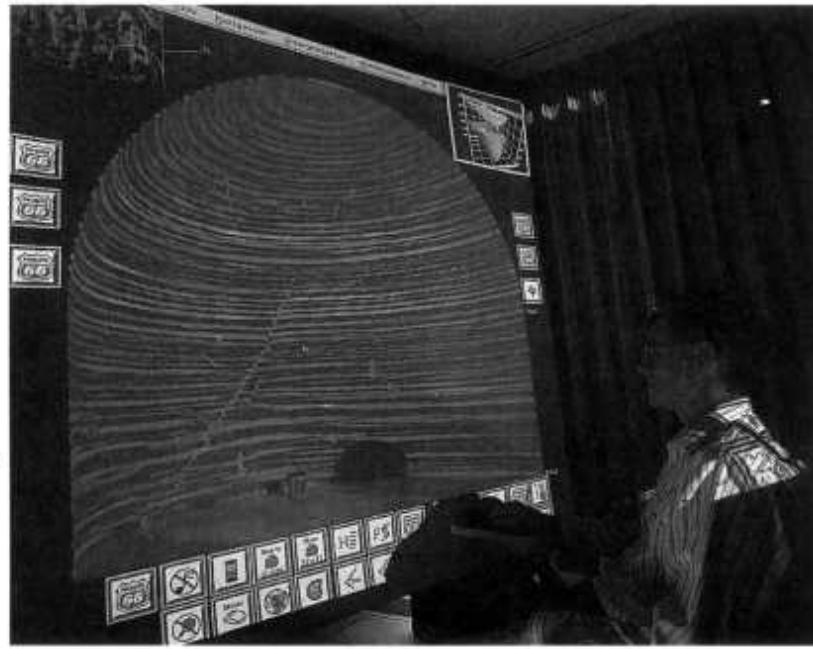
One of the sectors of the economy that has aggressively pursued ways to improve information visualization is the oil and gas industry. Their motivation to do so is the large financial stake associated with exploration and well management. The discovery of new oil reservoirs is based on visualization and interpretation of seismic data. Teams of geologists, geophysicists, and engineers need to sift through tens of gigabits of data looking for changes that would indicate the presence of oil underground. Exploratory holes costing \$10 million each are subsequently drilled to confirm their assumption [Bartling, 2002]. Not long ago, two of three such holes were dry (a well drilled in the wrong place), representing significant financial losses.

Seismic data are three dimensional, and thus well suited for visualization in VR. Another advantage of VE-mediated visualization is the ability to use displays larger than the desk-top CRT in order to better visualize the large volume of data involved. Scientists at Phillips Petroleum recently developed a display designed for unencumbered team viewing of seismic information [Neff et al., 2000]. The 1.83 m x 1.83 m Seismitarium represents a cross between a wall-type (flat) display and a CAVE, using a single 1280 x 1024 digital projector. As shown in Figure 9.18a, the Seismitarium has a curved central area consisting of a combination of a quarter sphere region placed on top of a semicylinder. This curved surface enhances the feel of depth when perspective-corrected graphics is projected on it, without the need for active

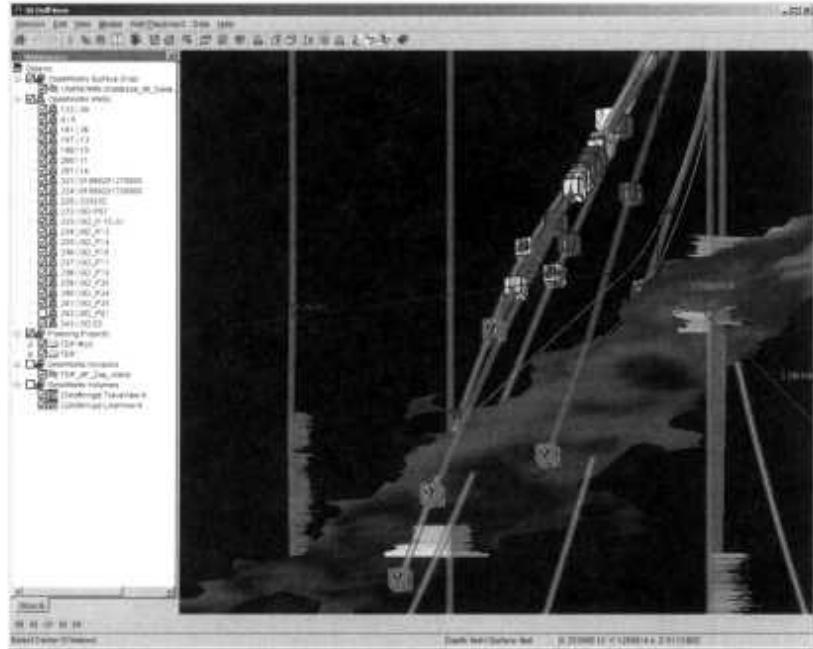
glasses. The curved portion of the display is surrounded by flat surfaces on the sides and bottom. These serve for projection of normally flat images such as GUI icons or 2D graphs. A hand-held paddle display is used when detail viewing is needed. This display is tracked (magnetically) such that the VR engine knows the paddle display location relative to the larger display. It then renders the corresponding image tile, but at much higher pixel resolution than on the rest of the curved display. The Seismitarium is marketed as the conCAVE and sold by FakeSpace Systems Inc. (Kitchner, Ontario, Canada). Its use in conjunction with seismic data visualization software has dramatically improved the odds of finding oil on the first exploratory hole. In fact, the hundreds of visualization centers using large displays that are in use today in the oil and gas industry have raised the success rate of exploration to an unprecedented 80%.

Finding reservoirs through seismic data visualization is a precursor to actual oil production. Oil is pumped through wells that are kilometers deep and need to be steered so they maximize output. This in turn requires careful planning to maximize porous sand surfaces (rich in oil) while avoiding water pockets or hard rock. The inefficient spread-sheet data formats available to drill engineers, combined with a lack of communication with the back-office geophysicists, traditionally had a negative impact on the quality of drilling. As a consequence, two thirds of the oil was not recovered, resulting in unnecessary drilling and negative impact on the environment. This situation has changed with the introduction of visualization software, such as Drill View (Landmark Graphics Co., Houston TX), to exploit well information. Realtime drilling data (such as pump rates, torque, drag, azimuth, inclination, rotation rates, and standpipe pressure) can provide valuable information on the nature of the geology being drilled. This can be logged into the same database that stores corresponding seismic subsurface information. It is then possible to recreate a 3D model that validates or corrects the database based on drilling data. As illustrated in Figure 9.18b [Sanstrom and Longorio, 2002], "knowledge attachments" represented by little cubes line the drilling trajectory and can be queried by a simple mouse click. The related information, including drilling incidents (such as a stuck pipe), is then retrieved instantaneously, allowing for much better collaboration between the back office and the rig

site. Use of such modern visualization techniques has increased production by 50%, as reported by BP Norway [Landmark Graphics Co., 2002]. The financial implications are obvious, with the resulting gains easily dwarfing the cost of the visualization system (hardware, software, and training).



(a)



(b)

Fig. 9.18 Visualization of seismic data in the oil industry. (a) The Seismitarium. From Neff et al. [2000]. Reprinted by permission. (b) "Knowledge attachments" correlating drilling events with the seismic subsurface model. From Sanstrom and Longorio [2002]. Reprinted by permission.

9.3.2 Volumetric Data Visualization

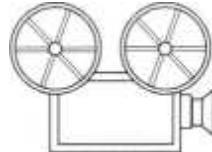
Underground seismic data are not only 3D in nature, but also volumetric, similar to data obtained from ultrasound, CT, or MRI imaging or from atmospheric and oceanographic modeling. What these types of data have in common is the absence of clear boundaries and the heavy dependence on visualization of a volume of space as a way to infer its physical structure [Wolfe and Liu, 1988].

Classic surface-based visualization techniques cannot present all the necessary information scientists require and the need arises for volumetric visualization. Volumetric visualization in turn implies volumetric graphics, volumetric hardware acceleration, special displays to present such data to the user in a volumetric way, and new human-computer interaction techniques. These topics make up the remainder of the book. We include them despite the fact that VR graphics is currently overwhelmingly surface-based. Rapid advances in volumetric rendering technology are making computers capable of real-time volumetric graphics and thus opening the door for its use in (future) VR systems.

9.3.2.1 Volumetric Graphics. This emerging subfield of computer graphics is concerned with the synthesis, rendering, and manipulation of scenes represented volumetrically [Kaufman et al., 1993]. Just as surface (or raster) graphics stores scenes in a frame buffer (discussed in Chapter 4), volumetric graphics stores them in a volume buffer. The scene information is stored in a discrete regular grid formed of voxels (abbreviation for volume cells). Voxels are elementary cubic volumes of space that are centered at a grid point. They represent the counterpart of pixels in raster graphics, but require much more memory to store. Since frame buffer memory is fixed, this requirement puts an upper bound on the resolution of the volumetric scene.

Kaufman et al. [1993] defined a taxonomy and data flow of volumetric graphics that starts with a geometric object model or sampled data from a real object. This sampled data could be, for example, 2D CT slices through the body anatomy. Whether data are sampled or computed, they need to be "voxelized" such that the voxel lattice stores information pertaining to the object. Finally, the object needs to be seen, and thus the volume needs to be rendered (illuminated and shaded). Figure 9.19 is an example of anatomy rendered volumetrically [TeraRecon Inc., 2002a]. It is apparent that this method affords a level of detail and a richness of visual information that is simply impossible to convey through surface graphics.

Apart from the ability to represent object interior and thus convey more visual information for nonisotropic (nonuniform) volumes, volumetric graphics has other advantages. First, unlike raster graphics, it is not affected by scene complexity since the voxelization stage is performed offline. Second, scan conversion is decoupled from scene refresh, unlike surface graphics, where rendering is done for every frame. Thus volumetric graphics is viewpoint-independent, and voxel representation facilitates approximation of object volume, area, distances, etc.



VC 9.7

There are also disadvantages associated with volumetric graphics, as it requires substantial amounts of memory and specialized processing hardware. More importantly, there is a space aliasing effect since object surfaces are approximated by voxels. This disadvantage becomes apparent when zooming in on a voxelized object. Thus, depending on the buffer resolution, object surfaces may appear too coarse and may require a separate geometrical data representation. Finally, since volumetric graphics is in its beginnings, the number of software vendors (for authoring and mostly medical application packages) is small. Table 9.1 summarizes the comparison of surface and volumetric graphics.



Fig. 9.19 Volumetric graphics representation of a body's internal organs and abdominal vasculature superimposed on the skeletal system. Image courtesy of TeraRecon, Inc.

9.3.2.2 Volumetric Graphics Hardware. This consists of rendering boards and volumetric displays providing interactive visual feedback to the user.

The level of interactivity depends of course on the application. Seismic data visualization may be content with slicing the voxel space and eliminating areas that block the view to the subsurface region of interest. In applications such as virtual colonoscopy, a minimum level of interaction is needed. This consists of object rotation and flythrough, which do, however, need to be accomplished in real time. To satisfy the resulting high computation load, modern volumetric simulators include a real-time volumetric rendering board.

VolumePro 1000 is a graphics accelerator offered commercially by TeraRecon Inc. (San Mateo, CA). Its main component is a Mitsubishi chip capable of rendering a lattice of 512 x 512 x 512 voxels at up to 30 frames/sec. In order to store all voxel information, each VolumePro board has 2 GB of memory, with larger models requiring the use of two (or more) boards. During rendering the entire 3D data are processed by casting rays from a view plane. The rays pick up color and opacity information as they pass through by trilinearly interpolating to the nearest lattice point. Subsequently, gradients are computed and lighting is assigned to the view plane image, which is then displayed by the computer. During flythrough the voxel space is reprocessed in rapid succession to allow interactive navigation. The VolumePro 1000 is also able to store surface geometry data in order to reduce the known space aliasing drawback of volumetric graphics. In doing so it works together with a surface graphics board installed on the same computer (PC or workstation). Application development is facilitated by the VolumePro 1000 Development Kit (which has sample code written in C++, image manipulation software, and system diagnostic programs) [TeraRecon Inc., 2002b]. One useful function of the software library is the programmer's ability to specify a particular VolumePro board to render a subvolume. This situation is encountered when multiple such boards reside on a single PCI bus.

TABLE 9.1. A Comparison Between Surface Graphics and Volumetric Graphics^a

Capability	Surface Graphics	Volume Graphics
Rendering performance	Sensitive to scene and object complexity	Insensitive to scene and object complexity
Memory and processing requirement	Variable, depends on scene and object complexity	Large but constant
Object-space aliasing	None	Frequent
Transformation	Continuous, performed on geometric definition of objects	Discrete, performed on pixel blocks (windows)
Scan conversion and rendering	Pixelization is embedded in viewing	Voxelization is decoupled from viewing
Rendering interior	No, surface only	Yes, rendering interior as well as surfaces
Viewpoint dependence	Requires recalculation for every viewpoint change	Precomputes and stores viewpoint-independent information

^a Adapted from Kaufman et al. [1993]. © 1993 IEEE. Reprinted by permission.

Displaying a volumetric rendered image on a standard CRT has limitations related to viewing angle and the need to use active glasses and head trackers. Autostereoscopic displays (discussed in Chapter 3) eliminate the need for active glasses, but limit the viewing angle to a relatively small portion of space. What is needed is a volumetric display that allows full 360° viewing angle without requiring any head gear or trackers. Volumetric displays typically use a moving element (either emitting or reflecting light) that displays slices through the voxel space. The brain then integrates the rapid succession of 2D slices into a 3D scene that appears to be floating in space (owing to the known inertia of the human visual system).

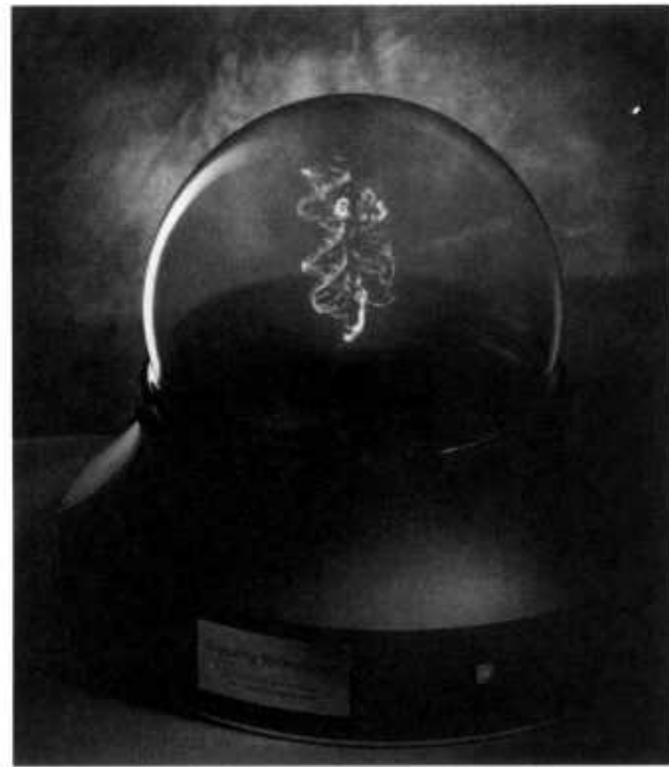
Early research into volumetric displays produced reciprocating multiplanar displays. These utilized a translating mechanism that moved an array of monochrome LEDs back and forth [Kameyama et al., 1992]. The image could be seen without any viewing apparatus on the head and from any angle. Drawbacks were related to noise (due to the translating structure), low image resolution (due to the LED's finite diameter), and the

inability to display more than one color. Furthermore, the room lights had to be dimmed due to low brightness of the volumetric display.

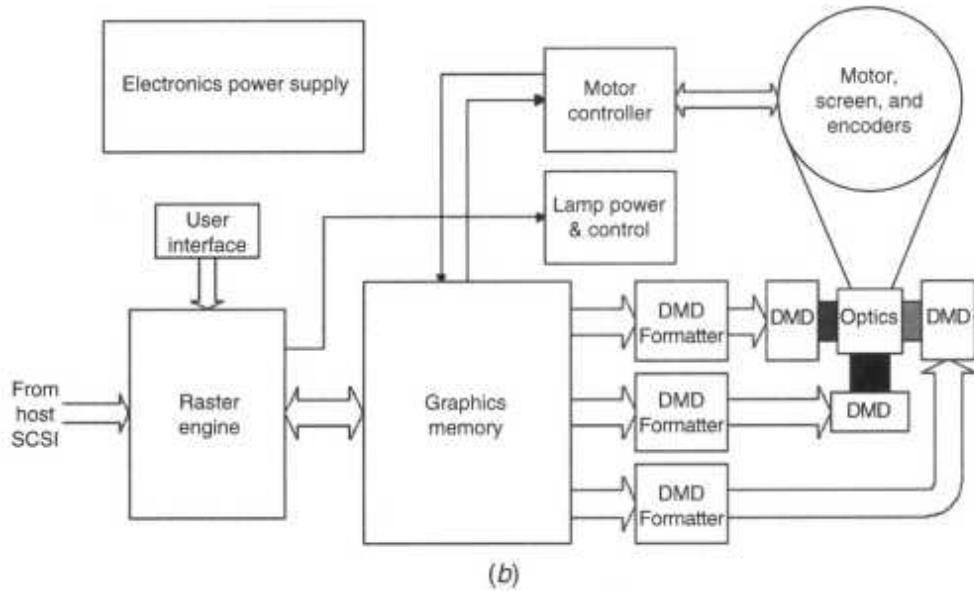
Modern volumetric displays, such as the Perspecta (Actuality Systems Inc., Burlington, MA), have solved some of the problems associated with reciprocating multiplanar displays. The design uses a Texas Instruments Digital Micromirror Device-based projector such that resolution is much higher. Furthermore, by using three DMDs it is possible to obtain eight colors (at high resolution) and many more in lower resolution (using color dithering). The image is projected through relay optics and onto a diffuse projection screen rotating at 730 rotations per minute. The rotating screen is enclosed in a spherelike structure of about 20 in. diameter, as shown in Figure 9.20a [Favalora et al., 2002]. Each slice in the voxel space corresponds to a little less than 2° of display rotation, such that the whole volume is rendered with a resolution of 768 x 768 x 198 slices. The limiting factor on the refresh rate (scenes/second) is the switching frequency of the Texas Instruments chip. At 5 kHz the projector limits the volumetric refresh rate to 24 scenes/sec. Another limitation associated with the projector is low image contrast (luminosity), such that the room lights still need to be dimmed to view the displayed volume.

One of the major design parameters for the Perspecta display was plug-and-pay compatibility with the PC host computer running the simulation. Thus geometric information from the host is sent to the volumetric display over a standard SCSI interface, as illustrated in Figure 9.20b. Here the embedded raster engine (a TMS320 DSP processor) converts the 3D data (lines or polygons) into a cylindrical voxel grid. These data are then stored in the volume buffer, which consists of 3 gigabits of DDR SDRAM. This large memory allows double buffering by storing two complete scenes (each with 198 slices of 768 x 768 imagery). The controller driving the rotating screen motor then sends memory pages in synchronization with the screen position data. Each memory page (representing slice data) is sent to three DMD formatters controlling the R, G, B DMD chips. Relay optics then recombines the color image prior to display on the rotating disk. This is somewhat similar, albeit at a smaller scale, to the digital projectors discussed in Chapter 3. Application development for the Perspecta is

facilitated by an OpenGL-based programming toolkit as well as an API that is currently under development. The intention is to extend the same windows metaphor (drag/drop, 3D pointers, and icons) to volumetric interactions.



(a)



(b)

Fig. 9.20 The Perspecta display: (a) the volumetric display optics; (b) the electrooptical system. From Favalora et al. [2002]. Reprinted by permission.

9.3.2.3 Interaction with Volumetric Displays. This form of interaction has additional capabilities besides those of familiar windows-based GUIs. These result from the 360° field of view by multiple users characteristic of multiplanar (dome-type) displays. Since volumetric displays are under active development, it is understandable that human factors (usability) data are lacking. One alternative is to build physical mockups of future volumetric displays instead and use these to develop new user interface techniques and input styles [Balakrishnan et al., 2001]. The researchers focused on user interface issues for volumetric displays, such as the Perspecta, where a physical (transparent) enclosure separates the user from the image. They constructed clear-plastic enclosures of various dimensions and placed them on a rotating physical base. Inside the enclosures they placed various objects to simulate the display of different data types. Finally, they used different physical objects (such as a stylus) to simulate input devices that may be used in conjunction with these volumetric displays.

One important interaction parameter is the mapping between the user's 3D input space and the volumetric scene space. A possibility is to display a virtual hand that is mapped to the user's hand wearing a tracked sensing glove. An alternative, illustrated in Figure 9.21 a, is to place the volumetric display upside down, such that the user views it through a half-mirror. This approach, reminiscent of the anastomosis simulator described in Chapter 8, gives the user a view of his or her hands operating within the domed area. The display enclosure could also become an input device if it detects where the user's fingers touch it. In addition to using the fingers as pointing devices, the user could stretch or push the dome surface if the enclosure is made of elastic material. This deformable membrane then becomes a de facto haptic interface, as illustrated in Figure 9.21b.

The ability of volumetric displays to present an omnidirectional viewable scene poses a problem when 3D menus are concerned. One solution is to map the menus on curved surfaces and place the same menu at several locations to allow direct view by the user (Fig. 9.21 c). An alternative is to rotate the dome base to bring a menu into view or rotate the scene using pushbuttons (or similar controls) placed on the display base ring.

Another interesting problem is navigation within the volumetric scene. Since the scene is rendered in cylindrical or spherical coordinates, nonlinear motion artifacts may appear as well as the known visual occlusion present in regular displays. One solution is to place the display on a turntable and rotate it such that the destination comes into direct view, as illustrated in Figure 9.21d. Other fundamental tasks, such as object selection, moving, and scaling, could also be facilitated by this arrangement. The user could place a stylus on the dome surface and a ray could be cast to intersect the object of interest. Other interaction techniques will likely be developed to allow the user to adjust the scene content in order to better interpret the 3D information it presents.

9.4 CONCLUSION

It is hoped that this chapter gave the reader a feel for where VR is starting to be applied, in some cases with spectacular results. Despite some negative media features on the technology described in this book, we have no doubt that VR is here to stay. The realization of its full potential depends on a workforce of talented and motivated professionals. It is the authors' hope that this book has contributed to this societal need. Now it is up to the reader to make a difference by developing new VR hardware and new programming techniques, improving system usability, and creating "must use" imaginative VR applications. We wish you success!

9.5 REVIEW QUESTIONS

1. In what manufacturing areas is VR applied?
2. What is virtual prototyping? How is FreeForm used for this?

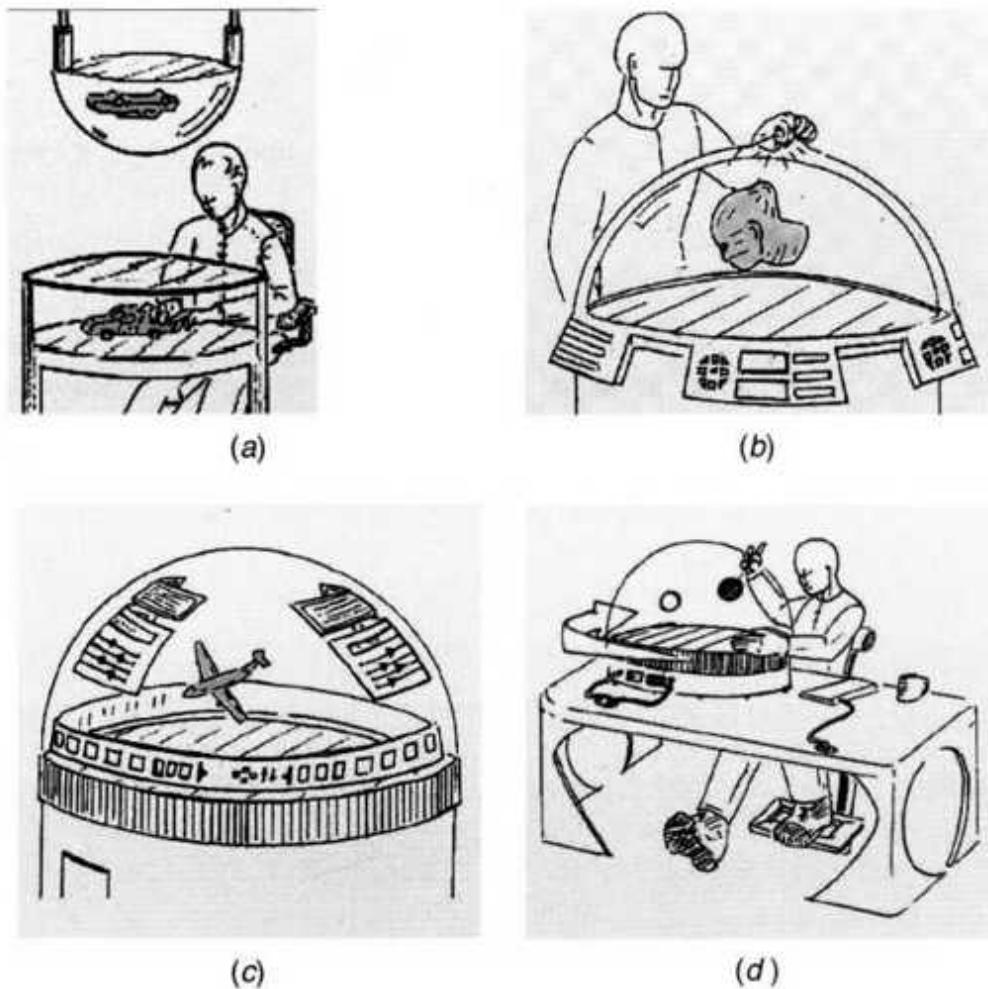


Fig. 9.21 Interaction with volumetric displays. From Balakrishnan et al. [2001]. © 2001 IEEE. Reprinted by permission.

3. How does VADE verify assembly? Make a drawing and explain.
4. In what ways is VR used in the automotive industry (both in prototyping and in ergonomic analysis)?
5. What differences exist between VirtualAnthropos and the Jack task analysis toolkit?
6. Give examples of the use of VR in the construction industry. Comment on differences depending on whether the user is an architect or an engineer versus a banker or an office executive.

7. How does augmented reality help maintenance tasks? Give examples.
8. Give examples of VR use in robot programming. Make a drawing and explain.
9. How does VR aid teleoperation in tasks where the visual feedback is corrupted?
10. How did NASA use VR and why?
11. Give examples of VR use in supervisory control. Comment on benefits. Make a drawing and explain.
12. How do data visualization tasks differ from other VR application domains?
13. What is the ConCAVE and why is it used in the oil exploration industry? Make a drawing and explain. Give examples of other related VR uses.
14. What is volumetric visualization?
15. What is volumetric graphics? Comment on its advantages and disadvantages relative to surface graphics.
16. Give examples of volumetric graphics hardware. Describe the Perspecta display and its electronics.
17. What user interaction techniques do you envision for domelike volumetric displays? Make a drawing and explain.

REFERENCES

Adams, A., 1996, "A Study of the Effectiveness of Using Virtual Reality to Orient Line Workers," Master's Thesis, DePaul University School of Learning, Chicago. Also online at www.adamsisolutions.com/download/tools/mu-study.pdf.

Anon, 1994, "Application: Motorola Trains Manufacturing Workers. Immersive VR Tests Best," CyberEdge Journal, Vol. 4(6), pp. 1-4.

Anon, 1999, "Industry Marketplace," Virtual Reality News, Vol. 8(1), online at www.vrnews.com/issuearchive/vrn080] .

Anon, 2000, "PSA Peugot Citroen Offers 3-D Virtual Automobile Showroom," Real Time Graphics, Vol. 8(7), p. 11.

AutoWeb.com.au, 1999, "Citroen Launches the Virtual Reality Showroom," 11 August 1999, online at www.autoweb.com.au/start_30/showall-/id_CIT/doc_cit990811/article.html.

Badler, N., M. Hollick, and J. Granieri, 1993, "Real-Time Control of a Virtual Human Using Minimal Sensors," Presence, Vol. 2(1), pp. 82-86.

Balakrishnan, R., G. Fitzmaurice, and G. Kurtenbach, 2001, "User Interfaces for Volumetric Displays," IEEE Computer, Vol. 34(2), pp. 37-45. Also online at oz.plymouth.edu/~wjt/HCI/ui1.pdf.

Battling, B., 2002, "Trends Driving Advancements in High-Performance Computing and Visualization," American Oil&Gas Reporter, 2002 (January), pp. 2-5. Also online at www.sgi.comVProducts/PDF/3303.pdf.

Bejczy, A., W. Kim, and S. Venema, 1990, "The Phantom Robot: Predictive Display for Teleoperation with Time Delay," in Proceedings of the IEEE Conference on Robotics and Automation, Cincinnati OH, pp. 546-551.

Bouzit, M., P. Richard, and P. Coiffet, 1993, "LRP Dexterous Hand Master Control System," Technical Report, Robotics Laboratory of Paris, France.

Bryson, S., 2002, "Information Visualization in Virtual Environments," in K. Stanney (Ed.), The Handbook of Virtual Environments Technology, Erlbaum, Mahwah, NJ, pp. 1101-1118.

Burdea, G., 1999, "Invited Review: The Synergy Between Virtual Reality and Robotics," IEEE Transactions on Robotics and Automation, Vol. 15(3),

pp. 400-4 10. Also online at
www.caip.rutgers.edu/vrlab/publications/papers/1999-ieee_tra.html.

Buxton, W., G. Fitzmaurice, R. Balakrishnan, and G. Kurtenbach, 2000, "Large Displays in Automotive Design," IEEE Computer Graphics and Applications, 2000 (July/August), pp. 68-75.

Davies, R., 2002, "Applications of Systems Design Using Virtual Environments," in K. Stanney (Ed.), *The Handbook of Virtual Environments Technology*, Erlbaum, Mahwah, NJ, pp. 10791100.

Deisinger, J., R. Breining, A. Robler, D. Ruckert, and J. Hofle, 2000, "Immersive Ergonomic Analyses of Console Elements in a Tractor Cabin," in *The 4th Immersive Projection Technologies Workshop*, Ames, IA, online at vr.iao.fhg.de/papers/ERGONAUT_iptw2000.pdf.

Electronic Data Systems, 2002, "Task Analysis Toolkit," Electronic Data Systems, Plano, TX. Also online at www.eds.com/products/plm/efactory/jack/tat.shtml.

Favalora, G., J. Napoli, D. Hall, R. Dorval, M. Giovinco, M. Richmond, and W. Chun, 2002, "100 Million-Voxel Volumetric Display," in D. G. Hopper (Ed.), *Cockpit Displays IX: Displays for Defense Applications*, SPIE, pp. 300-312. Also online at www.aetuality-systems.com/admin/publications/Actuality_Whitepaper_AeroSense_2002.pdf.

Flaig, T., K. Grefen, and D. Neuber, 1996, "Interactive Graphical Planning and Design of Spacious Logistic Environments," in Proceedings of the Conference of the FIVE Working Group, Scuola Superiore Santa Ana, Pisa, Italy, pp. 10-17.

Fraunhofer Institute for Computer Graphics, 1997, "Virtual Prototyping," company brochure, Darmstadt, Germany, 1997. Also online at www.igd.flig.de/www/igd-a4.

Fraunhofer Institute for Manufacturing Engineering and Automation, 1996, "VR4-Software Tool for Dynamic and Real-Time Oriented Virtual

Environments," company brochure, Stuttgart, Germany. Also online at www.ipa.fhg.de.

Gomes de Sa, A., and G. Zachmann, 1999, "Virtual Reality as a Tool for Verification of Assembly and Maintenance Processes," *Computer & Graphics*, Vol. 23(3), pp. 389-403.

Gruenbaum, P., W. McNeely, H. Sowizral, T. Overman, and B. Knutson, 1997, "Implementation of Dynamic Robotic Graphics for a Virtual Control Panel," *Presence*, Vol. 6(1), pp. 118-126.

Information Society Initiative, 2000, "UK Business Potential for Virtual Reality," Executive Summary, Department of Trade and Industry, London.

Institute for Robotics Research, 2002, "Projective Virtual Reality," online at www.irf.de/cosimir.eng/vr/ProjektiveVR/ProjectiveVR.htm.

Ipsos Reid, 2002, "Virtual Test Marketing-VirtualReid. The Most Realistic Measure of Consumer Behavior," company brochure, Ipsos Reid, New York. Also online at www.ipsos-reid.com/search/us/services/dsp_vr.cfm.

Jayaram, S., Y. Wang, and U. Jayaram, 1999, "A Virtual Assembly Design Environment," in Proceedings of IEEE Virtual Reality '99 Conference, Houston, TX, pp. 172-179.

Jet Propulsion Laboratory, 2002, "Long Range Science Rover," online at telerobotics.jpl.nasa.gov/tasks/scirover/.

Juster, N., J. Maxfield, P. Dew, S. Taylor, M. Fitchie, W. Ion, J. Zhao, and M. Thompson, 2001, "Predicting Product Aesthetic Quality Using Virtual Environments," *Journal of Computing and Information Science in Engineering*, Vol. 1(2), pp. 105-112.

Kameyama, K., K. Ohtomi, and Y. Fukui, 1992, "A Virtual Reality System Using a Volume Scanning 3-D Display," in International Conference on Artificial Reality and Teleexistence, Tokyo Japan, pp. 49-62. Also online at www.star.t.u-tokyo.ac.jp/ic-at/papers/92049.pdf.

Kanai, S., H. Takahashi, and T. Kishinami, 1997, "Networked Haptic Interfaces for Distant Redesign and Review of Free-Form Surfaces," in Proceedings of ASME Design Engineering Technical Conference, Sacramento CA, DETC 97/DFM 4366.

Kaufman, A., D. Cohen, and R. Yagel, 1993, "Volume Graphics," IEEE Computer, Vol. 26(7), pp. 51-64. Also online at www.cs.sunysb.edu/vislab/projects/volume/Papers/index.html.

Kheddar, A., C. Tzafestas, P. Coiffet, T. Kotoku, S. Kawabata, K. Iwamoto, K. Tanie, I. Mazon, C. Laugier, and R. Chellali, 1997, "Parallel Multirobot Long Distance Teleoperation," in Proceedings of ICAR '97, Monterey, CA, pp. 1007-1012.

Kheddar, A., K. Tanie, and P. Coiffet, 1998, "Detection of Discrepancies and Sensory-Based Recovery for Virtual Reality-Based Telemanipulation Systems," in Proceedings of IEEE 1998 International Conference on Robotics and Automation, Leuven, Belgium, Vol. 4, pp. 2877-2883.

Kheddar, A., R. Chellali, and P. Coiffet, 2002, "Virtual Environment-Assisted Teleoperation," in K. Stanney (Ed.), *The Handbook of Virtual Environments Technology*, Erlbaum, Mahwah, NJ, pp. 959-997.

Kim, W., 1996, "Virtual Reality Calibration and Preview/Predictive Display for Telerobotics," *Presence*, Vol. 5(2), pp. 173-190.

Kotoku, T., 1992, "A Predictive Display with Force Feedback and Its Application to Remote Manipulation System with Transmission Time Delay," in Proceedings of 1992 IEEE/RSJ International Conference on Intelligent Robots Systems, Raleigh, NC, pp. 239-246.

Landmark Graphics Co., 2002, "BP Norway Optimizes Geosteering with Real-Time Link to 3D Model," in *Solutions in Action*, Landmarks Graphics Co., Houston, TX. Also online at www.lgc.com/resources/sia-bpnorway.pdf.

Mechanical Dynamics, 2002, "Dynamic Designer," company brochure, Mechanical Dynamics, Ann Arbor, MI. Also online at www.dynamic.designermotion.com.

Nash, J., 1997, "Wiring the Jet Set," Wired Magazine, online at www.wired.com/wired/5.10/wirimg-pr.html.

Neff, D., J. Singleton, J. Grismore, J. Layton, and E. Keskula, 2000, "Seismic Interpretation Using True 3-D Visualization," Leading Edge, 2000 (June), pp. 523-525.

Neumann, U., and A. Majoros, 1998, "Cognitive, Performance, and Systems Issues for Augmented Reality Applications in Manufacturing and Maintenance," in Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS), Atlanta, GA, pp. 4-11.

Oyama, E., N. Tsunemoto, S. Tachi, and Y. Inoue, 1993, "Experimental Study on Remote Manipulation Using Virtual Reality," Presence, Vol. 2(2), pp. 112-124.

Rehg, J., 1997, Introduction to Robotics in CIM Systems, Prentice-Hall, Upper Saddle River, NJ.

Rosenberg, L., 1993, "The Use of Virtual Fixtures to Enhance Telemanipulation with Time Delay," in Proceedings of ASME WAM, Vol. 49, New Orleans, LA, pp. 29-36.

Rosenberg, L., 1994, "Virtual Fixtures: Perceptual Overlays Enhance Operator Performance in Telepresence Tasks," Ph.D. Thesis, Stanford University, Stanford, CA.

Sanstrom, B., and P. Longorio, 2002, "Application of an Innovative 3D Visualization Tool for Reservoir Development Projects," Oil & Gas Journal, Vol. 100(8), Online at www.lgc.com/resources/article-innovative_3d_visualization.pdf.

Scraft, R., J. Neugebauer, and K. Grefen, 1997, "Factory and Logistics Planning with Virtual Reality," in Proceedings of the 7th International Conference of Flexible Automation and Intelligent Manufacturing, Begell House, New York, pp. 958-968.

SensAble Technologies, 2000, "The Making of Saint Fruition," SensAble Technologies, Woburn, MA.

SensAble Technologies, 2002, "FreeForm Manual Version 5.0," SensAble Technologies, Woburn, MA. Also online at www.sensable.com/freeforml.

Shewchuk, J., K. Chung, and R. Williges, 2002, "Virtual Environments in Manufacturing," in K. Stanney (Ed.), The Handbook of Virtual Environments Technology, Erlbaum, Mahwah, NJ, pp. 1119-1141.

Short Brothers, 1998, "Application Briefing: Designing for Maintenance," Virtual Reality News, Vol. 7(6), pp. 15-18.

Strommer, W., J. Neugebauer, and T. Flaig, 1993, "Transputer-Based Virtual Reality Workstation as Implemented for the Example of Industrial Robot Control," in Proceedings of Interfaces to Real and Virtual Worlds Conference, Montpellier, France, pp. 137-146.

TeraRecon Inc., 2002a, "Medical Image Gallery," online at www.terarecon.com/gallery/med-gallery.html.

TeraRecon Inc., 2002b, "VolumePro 1000 OEM Development Kit," company brochure, TeraRecon Inc., Concord, MA. Also online at www.terarecon.com/downloads/products/vp1000_sdk.pdf.

Whyte, J., 2001, "Business Drivers for the Use of Virtual Reality in the Construction Sector," in Proceedings of the Conference on Applied Virtual Reality in Engineering & Construction. Applications of Virtual Reality Current Initiatives and Future Challenges, Gothenburg, Sweden, pp. 99-105.

Williams, T., The Purdue Enterprise Reference Architecture, Instrument Society of America, Research Triangle Park, NC.

Wolfe, R., and C. Liu, 1988, "Interactive Visualization of 3D Seismic Data: A Volumetric Method," IEEE Computer Graphics and Applications, 1988 (July), pp. 24-30.

Yanagihara, Y., T. Kakizaki, K. Arakawa, and A. Umeno, 1996, "Task World Reality for Human and Robot System-A Multimodal Teaching Advisor and Its Implementation," in Proceedings of 1996 IEEE International Workshop on Robot-Human Communication, Tsukuba, Japan, pp. 38-43.

INDEX

- 2D graph, 373, 374
- 2D image, 287
- 2D sound, 129

- 3D audio feedback, 92
- 3D audio system, speaker-based, 92
- 3Ball, 42
- 3D character, 194
- 3D digitizer, 160, 162
- 3D digitizer, HyperSpace, 160, 161
- 3D digitizer, MicroScribe 3D, 160
- 3D digitizer, work envelope, 161
- 3D geometry, 373
- 3D graphics, 11, 221, 304
- 3D icon, 251
- 3D measurement technique, 24
- 3D menu, 381
- 3D model, 121, 164, 174, 319–323, 351, 352
- 3D Mouse, 42, 43
- 3D object, 123, 160, 261
- 3D organ model, 288
- 3D position, 19
- 3D rendering chipset, 10
- 3D scanner, 160, 162
- 3D scanner, accuracy, 163
- 3D scanner, camera-laser assembly, 162
- 3D scanner, Cyrax 2500, 163
- 3D scanner, data conversion, 163
- 3D scanner, dot cloud, 163, 319
- 3D scanner, errors, 162

3D scanner, laser beam, 162
3D scanner, Polhemus FastScan, 162, 163
3D scene, 237
3D shape extraction, 287
3D slide, 4
3D sound, 8–10, 18, 58, 88, 299, 313, 316
3D sound card, 91, 110, 128
3D sound display, 57
3D sound generator, 16, 340
3D sound hardware 88, 89
3D sound processing, 92
3D sound processor, 89
3D sound source, 268
3D sound system, loudspeaker-based, 91
3D sound, virtual, 89
3D space, 17, 84
3D surface, 158
3D tracker, 329
3D transformation, 130, 136
3D vector, 42
3D view, 203, 251
3D viewer, 319
3D VR, 12
3Dlabs Inc., 10, 140, 142–145
3rdTech, 37

A-10 simulator, 339, 340
A/D converter, 23, 25, 49, 50, 52, 53,
 102
AAC, 70
abdomen, 288

Abrash, 132
absolute coordinates, 41
absolute position, 41, 42, 45, 173
absolute position control, 43
AC magnetic field, 24, 25, 31
accelerometer, 24, 38–40
accelerometer, bias, 39
accident re-creation scenario, 236
AccuTouch, 297, 298
acrophobia, 311
active glasses, 72, 73, 76, 78, 80, 83, 89,
 129, 130, 136, 142, 257, 272, 288,
 316–318, 374, 379
active glasses, CrystalEyes, 83
active glasses, incorporating tracker, 74
active glasses, IR controller, 73
active glasses, IR emitter, 72, 74, 83
active glasses, IR receiver, 74
active glasses, IR transmitter range, 75
active glasses, orthochromatic liquid
 crystal shutters, 73
active glasses, synchronization plug, 129
active glasses, wired, 75, 129
active glasses, wireless, 73
active matrix LCD, 62
active military unit, 334
activities of daily living, 304
actor, 29, 35
Actuality Systems Inc., 379
actuator, 93, 192, 263, 265, 279, 295, 297,
 306, 340, 353
actuator driver unit, 100
actuator interface, 99
Adams, 279, 360
Adams Consulting, 360
adaptation, 274, 275
adaptation, accelerated, 271
adaptation, distributed practice, 274
adaptation, error-corrective feedback, 274
addiction, 324
adduction–abduction angle, 53
Adelstein, 31, 189
Adept 1, 363
ADHD, 313, 314
ADHD, assessment, 313, 314
adjacent triangles, 158
ADSL, 150, 151
advance programming interface, 52
Advanced Visualizer, 134
after action review theater (AART), 334
aftereffects, 276
agent, 194–196

agent, autonomous, 195, 196
agent, behavior model, 195
agent, behavior rules, 195
agent, dance performance, 196
agent, Dexter, 195, 196
agent, emotion, 195
agent, emotion-based behavior, 195
agent, guided, 194
agent, programmed, 197
agent, reflex behavior, 195, 196
agoraphobia, 311
AI, 324, 328, 331
air bellow, 97
Air Force, 328, 340
Air Force Research Laboratory, 338–341
air humidity, 34
air temperature, 33
Air Warning System, 340
aircraft, 328, 329, 334, 338, 350, 360, 362
aircraft carrier, 326
aircraft maintenance, 360
aircraft manufacturer, 357
aircraft production, 362
aircraft trainer, 339
Airey, 202–204
airline industry, 311
airline maintenance personnel, 357
airplane, 6, 7, 251, 270, 305, 306, 311–313,
 326, 328, 338, 340, 358, 360, 362
airport, 290, 311, 313
airway secretions, 295
Akka, 74
Alias Wavefront, 322
alien, 324
alien attack, 324
Alphaworld, 279, 280
aluminum, 31, 32, 351, 352
aluminum mirror element, 77
Alzheimer’s disease, 313
Amato, 297, 298
American Association of Orthopedic
 Surgeons, 305
American Cancer Society, 296
American government, 7
American Heart Association, 294
American Society for Surgery of the Hand,
 46, 47
American Stroke Association, 308
Ames, 212
AMLCD, 62, 63
amphibious assault, 251
amusement park, 269

An, 95
analog current, 107
analog data, 66
analog display connector, 131
analog monitor, 131
analog signal, 89, 102
analog switches, 34
analysis of variation (ANOVA), 247
anastomosis, 299
anastomosis simulator, 299, 300, 381
Anatomic VisualizeR, 288, 289
anatomical database, 287
anatomical model, 288, 289
anatomical model, textured, 294
anatomical organ, 265
anatomical range limit, 268
anatomical structure, 289
anatomy, 287–289, 293, 294, 297
anatomy, mental image, 289
anatomy, teaching, 288
anesthesia, 295, 296
anesthesiologist, 296
angioplasty, 293, 294
animation, 4, 35, 52, 134, 226, 287–289,
 309, 315, 351
ankle, 268, 305, 306
Anon, 42, 44, 46, 51, 212, 287, 300, 357,
 360
antenna, 24, 25, 27
anteposition–retroposition, 46
antialiasing, 118, 124, 132, 135
antialiasing, subpixel region, 118
antisocial behavior, 279
anxiety management, 312
API, 221, 235
application, 3, 9, 29, 34, 44, 65
application developer, 231
application development, 285
application process, 232
application programming interface (API),
 210
application, porting, 211
application, requirements, 17
application-specific integrated circuit
 (ASIC), 136
applications, aerospace, 286
applications, emerging, 13
applications, entertainment, 13
applications, information visualization, 13
applications, manufacturing, 13
applications, medical care, 13
applications, military, 13

applications, robotics, 13
applications, see-through, 65
applications, visually demanding, 81
Applied Research Associates, 30
architect, 358
architectural blueprints, 321
architectural design, 160
architectural detail, 321
architectural modeling, 202
architectural planning, 286
architectural walkthrough, 200
architecture, 17
Argone E2 manipulator, 256
arm, 178, 196, 309, 314, 319, 357
arm actuator, 110
arm reach, 308
arm-reaching motion, 311
armored unit, 332
Army, 328, 329
Army Reserve, 334
Army, German, 329
Army, US, 331
art historian, 319
artificial intelligence, 277, 362
artificial reality (AR), 3, 65
artificial repelling force, 364
artillery, 332, 337, 342
artist, 1, 331
Arts, 286, 342
Ascension Technology Co., 27–30, 36, 39
ASCII, 32
ASIC, bus interface, 145
ASIC, geometry distributor, 138
ASIC, geometry engine, 136, 145
ASIC, HIP, 137
ASIC, rasterizer engine, 136, 145
assembler code, 211
assembly, 365, 371
assembly fit, 354
assembly language, 358
assembly line, 360
assembly operation, 357
assembly scene, 364
assembly sequence, 353
assembly task, 256, 260–262, 361
assembly tolerances, 355
assembly verification, 355
assembly worker, 362
associative memory, 360
astronaut training, 7
asymmetrical digital subscriber line
(ADSLs), 150

asynchronous communication, 373
ATI Technologies, 129, 130, 131
atmospheric attenuation factor, 123
atmospheric drag, 317
atmospheric effects, 118
atomic energy, 278
attention deficit, 313
audio coprocessor, 89
audio cue, 329
audio feedback, 128, 218, 262, 263, 332,
 364
audio system, multi-channel, 90
audio system, multi-speaker, 90
audio workstation, 89, 91, 110
audiovisual, 286
auditorium, 200–202, 204
auditory channel, 97
auditory cue, 265, 306
auditory depth perception, 18
auditory feedback, 16, 110, 262, 264,
 297
auditory localization acuity, 18
auditory mislocalization, 276
auditory system, 86, 88
auditory system, damage, 268
augmented reality, 1, 65, 360–362
Austria, 80
authoring pathway, 212
authoring task, 212
authoring tool, 160, 214, 215, 222
AutoCAD, 160, 161, 164, 214
Autodesk, 211
Autodesk Inventor, 354
Autodesys Inc., 160
automated forces, 328
automotive industry, 349
autostereoscopic adaptation coder, 70
autostereoscopic display, 68, 69
autostereoscopic display, active, 68,
 70
autostereoscopic display, active matrix
 back-lighting, 69
autostereoscopic display, back-lighting
 distance, 69
autostereoscopic display, barrier stripe
 film, 70
autostereoscopic display, dark subpixel, 70
autostereoscopic display, display mode
 selection, 70
autostereoscopic display, DTI 2018XL, 68,
 69
autostereoscopic display, Ecomo4D, 70–72

autostereoscopic display, illumination mechanism, 68
autostereoscopic display, image resolution, 68–70
autostereoscopic display, noisy operation, 70
autostereoscopic display, passive, 68
autostereoscopic display, prism mask, 70
autostereoscopic display, stereo perception area, 68
autostereoscopic display, stereo viewing zone, 69
autostereoscopic display, viewing angle, 70
autostereoscopic display, Virtual Window, 68–70, 72
autostereoscopic display, weight, 70
AutoWeb.com.au, 362
avatar, 18, 151, 177, 196, 235, 236, 238, 271, 279, 290, 291, 315–318, 329, 337, 338, 357, 370, 371
avatar, body posture, 196, 357
avatar, control, 29
avatar, palm, 290
avatar, skin, 290
aviation support unit, 334
aviation unit, 329
AWACS simulator, 340
AWadvs-04, 134

background noise, 34
Badique, 324
Badler, 357
Balakrishnan, 381, 382
ball, 84
ball, non homogeneous, 186
Barco, 76
Barco Baron, 78, 79, 83
Barner, 194
Barnes, 34
Bartling, 374
battalion, 328, 334
Batter, 6
battery, 329
battle, 342
battle drill, 332
battle master control (BMC), 334
battle scenario, 332
battlefield, 151, 251, 252, 290, 332, 333, 366
beamsplitter, 66
behavior, 194, 324, 357
behavior engine, 323

Bejczy, 1, 368
Bellemans, 80
benchmark, 133, 228, 295
benchmark, compound score, 135
benchmark, CPU, 134
benchmark, Indy3D, 133
benchmark, table, 133
benchmark, test, 133
benchmark, viewperf, 134, 135
benchmark, viewperf score, 135, 136,
 146
benchmark, viewset, 134, 135
bend sensor, 53
Bernardini, 319, 320
Berry, 321, 323
Bezier curve, 161
bicycle helmet, 351
biking, 276
billiard ball, 42
binary switch, 45
Biocca, 265, 266
biomechanics, 17
BioSimMER, 290
bioterrorism, 290, 291
biowarfare, 290
Bishop, 166
blackboard, 313, 314
bleeding, 295
Blinn, 168, 170
Blood, 27, 28
blood, 297, 299
blood clot, 308
blood vessel, 288, 299, 300
BMW, 361
body, 271, 276, 288, 293, 308
body anatomy, 376
body motion, 16
body orientation, 24
body position, 23, 95
body, discomfort, 357
body, interior, 293
body, sensory-motor control, 95
Boeing 737, 311
Boeing 747, 362
Boeing Co., 362
Boian, 227–230, 306, 309, 310
Bombardier Ltd., 357
bombing range, 339
bone, 289
Borriello, 17
Bosnia, 331
Boston Dynamics, 212, 236–238, 299

Boud, 260, 261
boundary of illusion, 265
bounding box, 181, 182, 213, 224, 365
bounding box, axis-aligned (AABB), 181, 182
bounding box, fixed size, 182
bounding box, oriented (OBB), 181, 182
bounding box, overlap test, 182
bounding box, projection interval, 182
bounding box, variable size, 182
Bouzit, 228, 308, 371
bowel, 299
Bowman, 246, 253–255
box, 255, 256
box, textured, 255
BP Norway, 376
Braille display, 262
brain, 58, 59, 64–66, 73, 83, 84, 86, 88, 94, 262, 268, 274, 275, 308, 326, 379
brain, depth perception, 73
brain, injury, 268
brain, pain response, 95
brain, plasticity, 308
brass, 31, 32
bridge, 332
Brill, 326, 327
Brinkley, 289
British Aerospace, 340
broadcast, 148
broadcast mode, 148
bronchial wall, 295
bronchies, 293
bronchoscopy, 293–295
Brooks, Jr., 5, 6, 285
Brutzman, 153, 227
Bryson, 373
Bui-Tuong, 122, 166
building, 202, 203, 236, 237, 332
building electrical mains, 142
building electrical wiring, 31
Burdea, 3, 4, 8, 9, 11–13, 17, 25, 26, 31, 33, 47, 51, 73, 84, 85, 92, 94, 120, 176, 186–188, 227, 229, 263, 292, 306–308, 363
Bürger, 304
burst transmission, 151
bus, Accelerated Graphics Port (AGP), 127, 130, 136, 144, 146
bus, AGP 8x, 128
bus, AGP bandwidth, 127
bus, AGP pins, 131
bus, AGP sideband address lines, 128

bus, AGP slot, 130
bus, AGP transfer rate, 128
bus, congestion, 146
bus, control, 136
bus, data, 136
bus, main system, 137
bus, PCI bandwidth, 127
bus, Peripheral Component Interface (PCI), 127, 128, 136, 144, 379
bus, UPA, 136
bus, vertex, 138
bus, xBox memory, 133
Bush, 267, 268
butterfly, 310
button, 22
Buxton, 351
Byzantine church, 321

C, 211
C&M Research, 100
C function, 10
C++, 322, 342, 379
C-17 aircraft, 360, 361
C/C++, 225, 235
cable, 107
cable backlash, 108
CAD, 17, 160, 164, 353, 354, 357
CAD authoring tool, 354
CAD file, 160
CAD model, 357
cadaver, 287, 288, 297
calendar, 194
calibration, 66, 157, 367, 369
calibration algorithm, 24, 27, 29
calibration signal, 25
Calvert, 278, 279
camera, 3, 4, 35, 70, 178, 179, 301, 367
camera, CCD, 35, 364
camera, coordinates, 178
camera, digital, 162, 306, 319
camera, fixed, 35
camera, focal point, 235
camera, infrared, 24
camera, miniature, 162, 268
camera, position, 296
camera, real, 370
camera, remote, 367
camera, view finder, 370
Canada, 211
Canadian Information Processing Society, 205
cancer, 293

cannon, 326, 337, 338
canonical view model, 179
canonical view volume, 180
capacitance, 51
capacitive bend sensor, 51
Capin, 151, 152
capstan drive, 102
capture time, 258
capturing task, 258
car, 123, 236, 301, 313, 350, 351, 354, 355, 358, 362
car body, 354, 355
car chassis, 364
car dealership, 362
car exterior, 354, 363
car interior, 363
car manufacturing, 364
car manufacturer, 350
car model, 362
car navigation, 17
car seat, 351
car surface, 355
car, dynamics, 276
carrier frequency, 25
Cartesian coordinates, 17, 85
Cartesian error, 95
casualty simulation module, 290
catheter, 297, 298
catheter tube, 297
catheterization, 297
CathSim, 297–299
Catmull, 168
CAVE, 78, 110, 140, 226, 252, 317, 351, 356, 357, 374
CAVE, cost, 80
CAVE, cubical structure, 78
CAVE, floor display, 78
CAVE, floor panel, 80
CAVE, four-projector configuration, 80
CAVE, graphics pipeline signal, 78
CAVE, Immersive WorkRoom, 80
CAVE, mirror assembly, 78
CAVE, modular viewing configuration, 80
CAVE, PC CAVE, 80
CAVE, PC cluster, 80
CAVE, RAVE, 80
CAVE, ReActor, 80
CAVE, retro projection, 78
CAVE, size, 80
CAVE, variants, 80
CAVE, vertical panel, 78
CAVE, visual artifacts, 80

CAVE, wall-like configuration, 80
cavity wall, 293
ceiling fan, 160
cell segmentation, 205
cell, pre-fetched, 204
center of projection, 180
Central Europe, 332
central nervous system, 270, 271
chair, 200, 313
Chapel Hill, 37
Chapin, 89
character animation, 29
characteristics, 51
chemistry, 314
Chen, 95
child, 314, 317
child, behavior, 314
child, injured, 331
China, 321
Cholewiak, 94
Christ, 319
Chuang, 213
circle, 351
circular buffer, 225
circulatory system, 293
Citröen, 362, 363
Clark, 1
claustrophobia, 311
clay, 350
client, 153, 220, 221, 289
client, authenticated, 220
client–server architecture, 220, 306
client–server communication, 221, 227
clipping, 117, 179, 214
clipping plane, 234
clock, 194, 213
close air support, 332, 334
Close Combat Tactical Trainer (CCTT),
 329, 334, 335
Close Combat Tactical Trainer (CCTT),
 aviation, 334
Close Range Weapons Simulator, 338
Close-range naval artillery, 337
cockpit, 338–340
cockpit dials, 334, 340
Codella, 3
cognitive aid, 362
cognitive assessment, 314
cognitive deficit, 313
cognitive gain, 318
cognitive load, 310, 324
cognitive skill, 334

cognitive task, 250, 360
Cohen, 182, 183
Coiffet, 12, 13, 17, 33, 73, 84, 85
Colgate, 188, 189
collapsed lung, 290
Collins, 94
collision, 353, 368
collision detection, 107, 126, 143, 180,
 182, 183, 219, 224, 234, 293, 354,
 357, 365
collision detection, 3D extent, 181
collision detection, AABB test, 182
collision detection, approximate, 181, 182
collision detection, bounding box, 181, 182
collision detection, computation time, 183
collision detection, cost, 182
collision detection, exact, 181–183
collision detection, multi-body, 183
collision detection, OBB-based, 182
collision detection, optimization, 182
collision detection, test, 182
collision detection, undetected collisions,
 182
collision force, 6, 126
collision response, 183, 184, 219, 234
colon, 295, 296
colon, interior, 295
colon, 3-D model, 295, 296
colon, cancer, 295
colon, cancer screening, 296
colon, polyp, 296
colon, virtual, 296
colon, wall, 295, 296
colonoscopy, 293, 295, 296
colonoscopy, optical, 296
colonoscopy, virtual, 295, 296
color, 3
color buffer, 119
color scene, 64
Colt, 295
combat zone, 338
combined processor memory switch, 136
commander, 329
communication bottleneck, 16
communication delay, 332, 369, 370
communication latency, 271
communication line, 20, 24
communication loop, 20
communication protocol, 221
communication rate, 53
communication skill, 340
communication time delay, 20

compiler, 122
compliance, 235
components of a VR system, 12
compound gesture sequences, 49
compound matrix, 175
computational engine, 356
computational load, 117, 123, 378
computer, 2, 4, 8, 16, 18, 20, 22, 34, 65, 72, 75, 78
computer architecture, special-purpose, 13
computer failure, 93
computer graphics, 1, 2
Computer Graphics Systems Development, 169
computer interface, 58
computer keyboard, 66
computer scientist, 331
computer screen, 160
computer virus, 153
computer, networked, 75
computer-aided manufacturing (CAM), 352
computing hardware, 116, 337
computing power, 135, 153
conCAVE, 374
concept design, 350
conductive fiber patch, 48
conductive ink flex sensor, 9
conductive polymer, 51
conductivity, 32
connectivity, 164
constraint force, 194
construction scheduling, 360
contact force, 93, 126, 184, 186, 190, 192, 193, 235, 262, 263, 265, 369, 370
contact force, constant, 94
contact force, direction, 191
contact force, model, 187
contact force, target, 107
contact geometry, 191
contact point, 191, 192
contact surface geometry, 93
continuous motion, 256
control algorithm, 247
control bandwidth, 97, 102
control box, 49, 263, 357
control loop, 93, 101, 102, 366
control panel, 186, 187, 316, 363
control server, 146, 147
control unit, 100
controller, 364
convex polyhedra, 183
conveyor, 358

convolving computation, 89
convolving engine, 89
convolving process, 91
Convolvotron, 89
Convolvotron, block diagram, 90
cooling fan, 131
copper, 32
cortex, 97
counter, 39, 40
CPU, 10, 33, 40, 89, 92, 117–119, 121, 126, 128, 133, 136, 142–144, 182, 214, 231
CPU, Pentium 4, 126
CPU, performance, 10
CPU, processing load, 122
CPU, speed, 126
Crane, 340, 341
crawling, 236
cross-modal enhancement, 265
crowd, 194, 196
crowd, behavior, 194, 196
crowd, guided, 196
crowd, leader, 196
crowd, level of autonomy, 196
crowd, social hierarchy, 196
CRT, 4, 64, 65, 68, 75, 140, 142, 145, 226, 257, 272, 371, 374, 379
CRT, Barco projector, 78
CRT, display, 22, 72
CRT, Electrohome projector, 78
CRT, electromagnetic radiation, 65
CRT, fast green coating, 76
CRT, graphics resolution, 74
CRT, high voltage, 64
CRT, image brightness, 74
CRT, magnetic field, 76
CRT, monitor, 31
CRT, mutual interference, 75
CRT, projector, 76–78
CRT, projector high-end, 76
CRT, side-by-side, 72, 75, 142
CRT, tube, 76
CRT, unshielded, 75
Cruz-Neira, 78
Crystal River Engineering, 89
CT, 287, 296, 376
CT, artifact, 288
CT, helical, 295
CT, image, 287
cube, 308
Currell, 84
current amplifier, 101
current source, 27, 100

Cutkosky, 95, 96
Cutt, 92
cutting plane, 43
CyberEdge Information Services,
 286
CyberForce, 109, 110
CyberForce, actuator unit, 109
CyberForce, cost, 110
CyberForce, maximum force, 109
CyberForce, orientation degrees of
 freedom, 109
CyberForce, work envelope, 109
CyberGrasp, 107, 108, 192, 231
CyberGrasp, actuator, 109
CyberGrasp, actuator housing unit, 107
CyberGrasp, back plate, 109
CyberGrasp, backpack FCU, 108
CyberGrasp, cable, 107
CyberGrasp, control bandwidth, 108
CyberGrasp, Force Control Unit (FCU),
 107, 109
CyberGrasp, maximum finger force, 108
CyberGrasp, mechanical bandwidth, 108
CyberGrasp, work envelope, 108, 110
CyberMath, 315
cybersickness, 266, 267, 269–272, 275,
 280
cybersickness, aftereffects, 276
cybersickness, age dependency, 271
cybersickness, age-induced resistance,
 271
cybersickness, aggravating factors, 269,
 271
cybersickness, alcohol influence, 271
cybersickness, exposure duration, 273
cybersickness, flashbacks, 276
cybersickness, gender influence, 271
cybersickness, guidelines, 276, 280
cybersickness, head spinning, 276
cybersickness, headaches, 276
cybersickness, immersion duration, 272,
 273
cybersickness, incidence, 269
cybersickness, influence of user
 characteristics, 271
cybersickness, influence of user degree of
 interactivity, 272
cybersickness, model, 269
cybersickness, repeated exposures,
 272–274
cybersickness, severity, 274
cybersickness, symptoms, 269

cybersickness, temporal aspects, 272
cybersickness, total severity score,
 272–274
cybersickness, under-reporting, 274
cybersickness,vection, 271
CyberTouch, 99, 100, 126, 231, 353, 361
cylinder, 315, 351, 353, 374
cylindrical bump, 190, 191
Cyra Technologies, 163

D/A conversion, 120
D/A converter, 25, 100, 102, 131, 149
D/A/D card, 105
Daeyang, 63
damper, 189, 190, 193
damper, directional, 190
damping, 126, 235
dark environment, 93
Das, 248
data acquisition, 161
data analysis, 247
data analysis, ANOVA, 247
data collection, 246
Data Explorer (DX-06), 134, 135
data extraction process, 373
data extraction time, 373
data packet, 221, 238
data processing, 163
data sample, 250
data validity, 247
data visualization, 349
database, 116, 117, 125, 157, 160, 164,
 169, 177, 198, 246, 247, 287–289,
 292, 294, 297, 301, 304, 306, 307,
 311, 321, 332, 333, 336, 339, 353,
 362, 365, 374
database traversal, 142
DataGlove, bending sensor, 25
DataGlove, electro-optical circuitry, 25
dataset, 17, 21
Daventry Associates Inc., 301
Davies, 359
daylight mission, 333
DC brushed motor, 105
DC current, 100
DC electrical actuator, 99, 102
DC magnetic field, 24, 31
De-mining training system, 330
dead reckoning, 221, 238, 332
DEC, 7
decoupled sensor, 53
Dede, 316, 317

Deering, 223, 225
Defense Advanced Research Projects Agency (DARPA), 332, 333
deformable membrane, 381
degrees of freedom (DOF), 17, 45, 46, 93, 256
Deisinger, 356, 357
Delaney, 12, 80, 321
DeLeon, 321, 323
demonstration itinerary, 196
Denavit, 172
depth perception, 256, 258
design, 278, 353, 356, 357, 359
design cycle, 252
design iteration, 357
Design Review (DRV-07), 134, 135
design review, 353
design revision, 350
design studio, 351
designer, 351–354, 357, 358
desk, 93, 98, 100, 102, 109
desk-supported display, 68
desk-supported display, autostereoscopic, 68
desk-top interface, 22
desk-top PC, 1
desk-top VR, 360
desktop monitor, 252
detail design, 359
Deutsch, 305, 306
dexterous manipulation task, 100
diagnosis, 287
diagnosis accuracy, 291
diagnostic procedure, 296
diagnostic skill, 289
diagnostic speed, 293
diagnostician, 289
diathermy, 303
die, 275
die manufacturer, 352, 353
differential algebra, 315
differential amplifier, 25, 27
differential geometry, 315
diffuse reflection coefficient, 123
Digital Anatomist, 289
Digital Anatomist Interactive Atlas, 289
digital audio signal, 89
digital clay, 351, 352
digital display, 130
digital encoder, 102
digital map, 250, 252, 254

Digital Micromirror Device (DMD), 77, 379, 381
Digital Neuroscientist, 289
digital projector, 374
digital radiography, 287
digital rectal examination, 291
digital signal processing, 25, 89
digital signal processor, 89
digital sketch, 351
digital-to-analog conversion, 138
Digitalo Design Co., 321
Dimension International, 10, 211
Dimension Technologies Inc., 69
dinosaur, 121, 164, 326
Dinsmore, 120, 121, 186
direct free-form deformation, 184
direct kinematics, 22, 45, 66
direct vertex manipulation, 159
Direct3D, 221, 235
DIS, vehicle simulation, 238
disaster relief, 328
disk, 260–262
dismounted infantry, 334
Disney, 324, 326
Disney Quest, 326
disorientation, 269
Displaced Temperature Sensing System, 100
display, 4, 116, 332, 374
display, analog, 145
display, CAVE-like, 81
display, composite, 146
display, CRT, 74
display, dimensions, 43
display, excessive weight, 67
display, flicker, 119
display, genlock, 76
display, hardware, 138
display, horizontal table-size, 78
display, inertia, 68
display, large volume, 140, 154
display, light-tight enclosure, 76
display, list, 137–139
display, loop, 20
display, master, 142
display, out-the-window, 76
display, panel, 68
display, projector based, 78, 83
display, PV290, 75
display, resolution, 75, 84, 121, 125, 133, 146

display, shape, 83
display, side-by-side, 140, 145
display, slave, 142
display, spherical, 81
display, three-panel, 75
display, tiled, 75, 140, 146, 147, 154
display, vertical blank period, 142
display, video, 118
display, wall, 140, 252, 271, 331, 351
display, wall-size, 81
display, window, 125, 172, 180
displayed image, 58
Dissimilar Air Combat Tactics, 340
distance learning, 314
distributed architecture, 90, 117
distributed computing architecture, 29
Distributed Interactive Simulation (DIS),
 151, 238, 239, 333, 334
Distributed Mission Training, 341
distributed simulations, 278
distributed VR, 315
DIVE, 315
Division Ltd., 9, 10, 211, 329, 330
DiZio, 276
dizziness, 117
DMA, 138
Doherty, 285
dome, 83, 88, 270, 329, 338, 381
dome, dimensions, 81
dome, FOV, 81, 83
dome, military flight training, 83
dome, multiple projectors, 81
dome, projector locations, 83
dome, rear-projection screen, 81
dominant hand, 43
Donovan, 10, 12
door, 203, 272, 313, 362
door, autonomous, 195
door, guided, 194
door, lock, 361
doorway, 272
Dragon, 250–253
DRE simulator, 293
DRE training, 292
Dresden 3D, 71
drill engineer, 374
Drill View, 374
drilling event, 375
drilling trajectory, 376
driver, 270
driving, 276, 324
drowsiness, 269

drug addict, 298
drug store, 362
DSP, 25, 89, 133
DSP card, 89
DSP chip, 89, 91
DTSS, 100, 101
dual-hand operation, 43
dual-processor architecture, 129
Duda, 85, 87, 91
dungeon, 278
Durlach, 59
DVD, 133
dynamic art, 319
dynamic bar menu (dynabar), 351
Dynamic Designer, 354

ear, 4, 84, 86–88, 91, 288
ear, canal, 84
ear, distant, 86
ear, eardrum, 88
ear, HRTF, 89
ear, inner, 87, 88, 270, 288, 289
ear, intensity of sound, 86
ear, otolith organs, 270
ear, outer, 88
ear, pinna, 87, 88
ear, semicircular canals, 270
Earth’s DC magnetic field, 24, 27, 39
eating disorder, 311
economy of motion, 301
eddy current, 26, 27, 31, 32
eddy current, magnetic field, 26
edge, 199
edge, shared, 158
Edison, 4
education, 17, 286, 319, 342
educational software, 317
Eichenlaub, 69
EKF, 41
elastic deformation, 180
elbow, 29
electric field, 316
electrical actuator, 97, 104, 107, 109
electrical charge, 316
electrical contact, 48
electrical wiring, 362
electrocardiogram, 250
electroencephalography, 250
electromyography, 250
electron, 316
Electronic Data Systems, 357, 358
electronic interference, 64

electronic unit, 9, 18
Electronic Visualization Laboratory, 78
electrostatic force, 316
elementary school, 317, 318
elevator, 272
Elsa Ag., 70, 71
Elsa Gloria III, 131, 133
Elsa Gloria III, fill rate, 133
embedded microprocessor, 27
embedded Pentium, 143
emergency medical response, 291
emergency medical technician (EMT), 290
emergency room, 290
Emering, 195, 196
Emory Health Sciences, 313
Emory University, 311
end-effector, 106
endoscope, 293–295, 301
endoscope, virtual, 294
endoscopic examination, 293
endoscopic exploration, 289
endoscopic procedure, 294, 295
endoscopy skill, 294
enemy force, 324, 332
energy transfer, 267
engineer, 8, 243, 361, 374
engineering, 3
Engineering Animation, 221
England, 321
enhanced reality, 1
entertainment, 278, 314, 324, 326, 342
entertainment applications, 286, 324
environment, computation model, 89
environmental conditions, 20
Epic Unreal, 322
Epley, 277, 279
ERGONAUT, 357
ergonomic analysis, 350, 356, 357
ergonomic analysis tool, 357
ergonomic assessment, 357
ergonomic toolkit, 357
Ethernet, 150
Ethernet line, 89, 107
Europe, 31
European Aeronautic Defense and Space
Company, 339, 340
European COVEN project, 153
European Surgical Institute, 304
evacuation simulation, 196
Evans and Sutherland, 4, 7
Everquest, 324
exerted force, 95

exoskeleton, 107, 109
experimental data, 247, 257
experimental design, 247, 293
experimental protocol, 245, 247, 248
experimental protocol, control group, 245, 293, 295, 304, 314
experimental protocol, data sampling frequency, 246
experimental protocol, experimental group, 245, 293, 295, 299, 302–304, 313, 314
experimental protocol, post-trial measurements, 246
experimental protocol, rest period, 245, 258
experimental protocol, session, 245, 336
experimental protocol, subject, 246
experimental protocol, trial, 245–248, 258, 293, 299, 300, 303, 370
experimental task, 248, 295
expert system, 287
exploratorium, 315
exploratory hole, 374
explosive disposal site, 366
exposure therapy, 311, 312
extendable library, 211
external force, 233
external sensor, 194
eye, 7, 58–60, 63, 66, 68, 83, 176, 268, 269
eye, binocular overlap, 58
eye, cataract, 268
eye, convergence angle, 58, 59
eye, corneal burn, 268
eye, corresponding image column, 68
eye, corresponding view, 70
eye, depth cue, 59
eye, depth perception, 58, 59
eye, exit pupil diameter, 60
eye, field of view (FOV), 58, 60
eye, fixation point, 58
eye, focus area, 58
eye, focusing distance, 62
eye, fovea, 58
eye, gaze, 236, 268
eye, image parallax, 59
eye, image tracking mechanism, 271
eye, injury, 268
eye, inter-pupillary distance (IPD), 59, 226
eye, lost focus, 75
eye, motion, 58
eye, occluded, 73
eye, photoreceptors, 58
eye, pixel/surface area ration, 81
eye, reflex, 290

eye, retina, 268
eye, retinal burn, 268
eye, sampling rate, 256
eye, space, 178
eye, stereopsis, 58, 59, 73, 74
eye, stereoscopic vision, 58, 59
eye, strain, 75, 254, 269, 271
eye, successiveness threshold, 94
eye, tracking system, 268
eye, tracking technology, 58
eye, viewing area, 59
eye, viewing axis, 58
eye, viewing field, 58
eye, viewing volume, 58, 59
EyePhone, 9

F-15, 340
F-16, 340
FA receptor, 94
Fabiani, 262, 264
face-mounted display, 62
facial expression, 236
factory, 359, 362
factory blueprints, 359
factory planning, 358
Fakespace Labs Inc., 22, 66–67
Fakespace Systems Inc., 78, 80–81, 374
false ceiling panel, 38
family life, 278
fast serial bus, 29
Favalora, 379
Faxon, 277, 279
FCS Control Systems, 106, 107
fear of flying, 311, 312
feedback, 16
feedback actuator, 105
feedback channel, 262
feedback force, 93, 105, 185–188, 190,
 191, 331
feedback interface, 57
feedback loop, 100
feedback modality, 250, 264
feedback quality, 260
feedback structure, 102
feedback type, 260
femur, 123, 124
fence, 253
ferrite, 31, 32
ferromagnetic cube, 24
ferromagnetic metal, 31
fiber loop, 49
fiber-optic sensor, 8, 49

field artillery, 332
field of view, 58, 74, 194, 338, 356, 381
Fifth Dimension, 46, 50, 51
fighter simulators, 340
file format, 3DS, 214, 222
file format, ASCII, 161
file format, DXF, 160, 161, 214, 222
file format, FLT, 214
file format, NFF, 215, 216, 222
file format, OBJ, 214
file format, PSS, 236, 238
file format, Swivel 3-D, 161
file format, TIFF, 169
file format, Wavefront, 161
file format, WRL, 215, 222, 224
fill rate, 131
finger, 8, 49, 95, 99, 107, 178, 213, 262, 265, 268, 275, 309, 310, 381
finger, abduction, 46
finger, attachment, 102
finger, bending, 9, 49, 52
finger, bone, 100
finger, closed, 95
finger, configuration, 49
finger, contact, 49
finger, contact force, 95, 107
finger, dimension, 178
finger, distal joint, 50
finger, extended, 52
finger, flexion, 46, 49, 51, 53
finger, force exertion bandwidth, 97
finger, grasping force, 306
finger, heat flow, 100
finger, inter-phalangeal joint, 52
finger, joint angles, 46, 48, 49
finger, link, 178
finger, motion, 46, 47, 53
finger, position, 46
finger, position data, 107, 109
finger, posture, 50
finger, proximal joint, 52
finger, range of motion, 49
finger, segment, 178
finger, strength, 306, 308
finger, sub-structure, 178
finger, un-flexed, 51
fingertip, 93–95, 100, 101, 107, 191, 193, 306
fingertip, area of influence, 192
fingertip, force, 107
fingertip, geometry, 191, 192
fingertip, haptic mesh, 191

fingertip, position, 178
fingertip, temperature, 101
fingertip, temperature feedback, 100
fingertip, touch sensor, 93
finite impulse response filter, 88
finite-state automata, 290
FIR, 88, 89
Fire GL2, 126, 129–131, 133–135, 140,
 143, 144
Fire GL2, block diagram, 130
Fire GL2, fill rate, 130
firing range, 340
first aid, 290
Fischer, 24
fish-eye lens, 329
Fisher, 7, 326
Fitts, 259
Fitts index, 259
Flaig, 365
flashlight, 290, 319
flat barrier, 340
flat panel display, 68, 70, 131
flat-shaded surface, 8
flex sensor, 50, 51
flexion angle, 50, 53
flight helmet, 7
Flight Helmet, 11
flight scenario, 312
flight simulator, 6, 270, 276, 290, 312, 338
flight stick, 252, 340
floating-point angle, 66
floating-point core, 138
floor, 88, 197, 256
floor plan, 203, 204
floor-supported display, 60, 66
floor-supported display, Boom3C, 66, 67, 72
floor-supported display, CCSV, 66
floor-supported display, counterbalanced
 CRT-based stereoscopic viewer, 66
floor-supported display, CRT, 66
floor-supported display, dead zone, 66
floor-supported display, excessive weight,
 68
floor-supported display, field of view, 66, 67
floor-supported display, graphics
 resolution, 66
floor-supported display, LEEP optics, 67
floor-supported display, optics, 66
floor-supported display, overhead
 counterbalanced arm, 67
floor-supported display, pendulum
 oscillation, 68

floor-supported display, supporting arm, 66
floor-supported display, WindowVR, 67, 68, 72
floor-supported display, work envelope, 66, 67
Florentine Pietà, 319, 320
fluorescent lamp, 69
flyby, 4
flying, 311, 324
flying speed, 46
FMD, 62
FMD, consumer-grade, 63
FMD, controller, 62
FMD, cordless TV, 62
FMD, cy-visor, 62, 63, 72
FMD, ergonomic advantages, 62
FMD, field of view (FOV), 62
FMD, free-form lens, 62
FMD, image, 62
FMD, image distortion, 62
FMD, image resolution, 63
FMD, IPD, 62
FMD, LCD-based, 62
FMD, Olympus Eye-Treck, 62, 63, 72
FMD, optical super resolution, 62
FMD, optics, 62, 63
FMD, polarization, 62, 63
FMD, power consumption, 63
FMD, resolution, 62, 63
FMD, Sony Glastron, 62
FMD, stereo FOV, 63
FMD, SVGA resolution, 63
FMD, weight, 62, 64
focal surface, 315
fog, 270, 333, 338
Foley, 122, 159, 165, 166, 168, 169, 172, 177
foot, 24, 306
foot pedal, 45, 46, 303
footstep positional information, 24
force, 44, 45, 100, 353, 367
force calculation, 180, 184
force control, 44
force control bandwidth, 95
force discontinuity, 190
force exertion, 95
force exertion, maximum, 95
force exertion, sustained, 95
force feedback, 13, 93, 102, 105, 184, 262–265, 310, 361, 367, 369, 370
force feedback arm, 104, 108

force feedback glove, 102, 107, 108, 192, 263, 268
force feedback interface, 102
force feedback interface, control bandwidth, 102
force feedback interface, grounded, 102
force feedback interface, mechanical bandwidth, 102
force feedback structure, 102
force feedback system, 103, 104, 108, 109
force field, 6
force isodyne, 357
force magnitude, 95, 190, 262
force mapping, 126, 180
force rendering, 143
force sensor, 106
force shading, 190, 191
force smoothing, 126, 180, 191
force target, 95, 109, 143
force threshold, 297
force tracking, 95
force tracking resolution, 95
force–surface deformation dependence, 185
forearm, 93, 297, 357
forearm, virtual, 298
form-Z, 160
forward air controller, 340
Foster, 89
Fourier transform, 88
Foxlin, 17, 18, 24, 33, 39–41
fps, 133, 135, 143, 203
fracture, 304, 305
fragment generator, 138
frame, 4, 21, 149, 228
frame buffer, 119, 120, 128, 130, 131, 133, 136, 141, 142, 145, 146, 149, 376
frame buffer, swap command, 142
frame buffer, swapping synchronization, 142
frame of reference, 17
frame refresh rate, 92, 120, 121, 123, 141, 146, 148, 149, 200, 203, 204, 214, 228, 243, 256–260, 268, 299, 319, 321, 358, 360, 365
frame refresh rate, user-specified, 200
frame rendering time, 200–202, 204, 205, 210, 228, 229, 256
frame rendering time, fixed, 201
frame rendering time, maximum, 201, 202
frame rendering time, variability, 202, 205, 229, 230

France, 8, 371
Franzen, 278
Frauenkirche, 321
Fraunhofer Institute, 358
Fraunhofer Institute for Computer
Graphics, 167, 350, 361
Fraunhofer Institute for Industrial
Engineering, 357
Fraunhofer Institute for Manufacturing
Engineering and Automation, 364
freedom of motion, 46
FreeForm, 351, 352
FreeForm, 3-D icon, 351
FreeForm, add clay mode, 351
FreeForm, digital prototype, 351
FreeForm, file export, 352
FreeForm, GUI, 351, 352
FreeForm, mirror all mode, 351
FreeForm, pencil tool, 351
FreeForm, Saint Fruition, 351
FreeForm, sketch plane, 351
FreeForm, smooth area mode, 351
FreeForm, symmetry plane, 351
FreeForm, tool, 351
FreeForm, tug mode, 351
FreeForm, wire cut, 351, 352
French Defense Department, 329
friction, 194, 265, 370
friction force, 126, 193
Fritz, 194
Fröhlich, 43
frustrum, 179, 180, 234
Fu, 172
Fuchs, 117
fuel filler flap, 355
Fukui, 262
fulcrum effect, 301
full-body motion, 29
Funkhouser, 200–202, 205
furniture, 88, 194, 197
Future Lab, 80

Gabbard, 250, 251
gall bladder, 300, 301
Gallager, 303
game, 9, 278
game, adventure, 324
game, console, 9, 132
game, interactive, 325
game, multi-player, 324
game, online, 324
game, PC-based, 324

game, platform, 129, 131
game, role-playing, 324
game, strategy, 324
game, VR, 324, 326
gardening, 318
gaze tracking, 253
GE Force 2, 338
GE Force 2, chip, 131
GE Force 2, fill rate, 131
GE Force 2, GPU, 131
GE Force 2, GTS, 146
GE Force 2, screen refresh rate, 131
GE Force 2, texture fill rate, 131
GE Force 3, 132
Gembicki, 334
General Electric, 324
General Electric Research and Development Center, 287
generation lock (genlock), 20, 142, 143, 145
geologist, 374
geometric algebra, 316
geometric modeling, 158
geometric primitive, 233
geometric transformation, 158
geometrical model, 117, 205
geometry, 13, 315
geometry chip, 129
geometry clipping, 136
geometry engine (GE), 119, 122, 125, 130, 136, 138, 169
geometry processor, 10
geometry processor unit (GPU), 131–133, 171
geophysicist, 374
Georgia Institute of Technology, 311
germ warfare, 290
German National Computer Science Institute, 78
Germany, 167, 304, 321, 332, 357, 358, 361, 364, 370
gesture, 2, 8, 51
gesture duration, 49
gesture input, 16, 370
gesture interaction, 49, 361
gesture interface, 46
gesture interpretation, 51
gesture library, 51
gesture mapping, 252
gesture recognition, 10, 316
gesture recognition interface, 53
GHOST, 231, 293

GHOST, Abstract node, 233
GHOST, C+ classes, 233
GHOST, callback function, 232, 234
GHOST, child node, 234
GHOST, collision detection, 234
GHOST, contact force, 232
GHOST, Data Type, 233
GHOST, Dynamic Property node, 233
GHOST, Effects class, 233
GHOST, Effects node, 233
GHOST, Geometry node, 233
GHOST, gstPHANToM node, 234
GHOST, Haptic Interface Device node, 233
GHOST, Haptic Interface node, 233
GHOST, haptics effects, 232
GHOST, interaction forces, 232
GHOST, interaction point, 232
GHOST, leaf node, 232
GHOST, Manipulator node, 233
GHOST, parent node, 232
GHOST, PHANToM Dynamic node, 233
GHOST, root node, 232
GHOST, Scene class, 233
GHOST, scene graph, 232–234
GHOST, scene graph traversal, 232
GHOST, separator node, 232
GHOST, sub-tree node, 232
GHOST, surface contact point, 233, 234
GHST, gstPHANToM, 234
Gifu University, 321
gimbal, 102
gimbal mechanism, 22, 102
gimbal rotational encoder, 22
gimbal sensor, 22
Girone, 143, 305
Glasstron, 11
global battlefield scenario, 342
global synchronization time, 221
global transformation matrix, 177, 178
Gloria III, 129, 131, 144
glove, 1, 7, 13, 361
glove dynamic range, 50
glove-embedded sensor, 48
Gold Standard Multimedia, 288
Goldberg, 163
Golgi organ, 95
Gomes de Sa, 361
Gomez, 124, 161
Gouraud, 123, 165, 256
Gouraud-shaded, 10, 228, 229
government, 10
GPS transducer, 17

graphic feedback, 370
graphic object, 3
graphical programming, 10
graphical programming environment, 235
graphical user interface (GUI), 296, 302,
 322, 351, 358, 359, 365, 374, 381
graphics accelerator, 4, 10, 121, 124, 126,
 129, 135, 136, 150, 169, 211, 222, 338
graphics accelerator, 3Dlabs Wildcat II, 133
graphics accelerator, Elite3D, 136
graphics accelerator, Expert3D, 136, 137,
 145
graphics accelerator, Fujitsu Celsius, 133
graphics accelerator, multi-pipeline, 143
graphics accelerator, NEC TE4E, 133
graphics accelerator, single-pipeline, 144
graphics accelerator, VolumePro 1000, 378
graphics accelerator, Wildcat 4110, 227
graphics accelerator, Wildcat II 5110, 145
graphics accelerator, Wildcat quad-pipe,
 145
graphics API, 225
graphics architecture, 137
graphics board, 120, 123, 130, 131, 166
graphics card, 126–128, 133, 134, 140,
 142, 146, 149
graphics card, ranking, 133
graphics computation, 61
graphics computer, 7, 70, 74, 78
graphics computer, multi-pipe, 81
graphics controller, 128
graphics display, 43, 57–60, 84, 89, 100,
 226, 252, 370
graphics display, auto-stereoscopic, 59, 60,
 379
graphics display, CRT, 58
graphics display, definition, 58
graphics display, Dome, 81
graphics display, ergonomic, 59
graphics display, ergonomic factors, 58
graphics display, field of view, 58
graphics display, hand-supported, 65
graphics display, image characteristics, 58
graphics display, image resolution, 58, 64
graphics display, large volume, 70
graphics display, LCD, 58
graphics display, monoscopic, 58, 67
graphics display, PanoWall, 81
graphics display, personal, 58, 60, 65, 70, 72
graphics display, stereoscopic, 58, 59
graphics display, V-Dome, 81
graphics display, wall type, 35, 81

graphics display, weight, 66
graphics display, WorkWall, 81
graphics engine, 89, 90
graphics feedback, 18, 84, 110, 125, 219,
 231, 243, 251, 260, 262, 263, 266,
 270, 271, 300, 301, 361, 364
graphics feedback, multi-channel
 synchronization, 141
graphics hardware, 133, 153, 211, 246
graphics hardware performance, 133
graphics image, 190
graphics language, 210
graphics library, 164, 177, 221
graphics loop, 234
graphics mode, 252, 253, 256–259
graphics monitor, resolution, 64
graphics output port, 140
graphics performance, 8, 11
graphics pipeline, 58, 120, 121, 123, 126,
 133, 138, 143, 145, 146, 154, 167,
 168, 179, 180, 183, 199, 205, 214,
 229, 231, 256, 373
graphics pipeline, application stage, 117,
 118, 121, 122, 136, 141, 158, 167,
 198, 199, 214
graphics pipeline, application stage
 bottleneck, 121
graphics pipeline, application stage tasks,
 117
graphics pipeline, asynchronous, 76
graphics pipeline, bottleneck, 121, 122
graphics pipeline, bottleneck in the
 rasterizing phase, 125
graphics pipeline, bottleneck stage, 120
graphics pipeline, CPU limited, 121
graphics pipeline, double buffering, 119,
 379
graphics pipeline, FIFO buffer, 120, 138
graphics pipeline, fill limited, 121, 125
graphics pipeline, geometry engine, 119
graphics pipeline, geometry stage, 117,
 118, 122, 131, 133, 157, 168, 171,
 178, 180, 201, 214
graphics pipeline, geometry stage
 bottleneck, 123
graphics pipeline, geometry stage
 computational load, 123, 129, 138
graphics pipeline, geometry stage FIFO
 buffer, 119
graphics pipeline, highly-parallel, 136
graphics pipeline, independent, 145
graphics pipeline, multiple, 75, 140

graphics pipeline, optimization, 121
graphics pipeline, output rate, 120
graphics pipeline, output signal conversion,
 61
graphics pipeline, programmable, 132
graphics pipeline, rasterizer unit, 119
graphics pipeline, rasterizing stage, 118,
 119, 124, 131, 133, 136, 168, 201
graphics pipeline, rasterizing stage
 computation load, 118
graphics pipeline, rasterizing stage FIFO
 buffer, 119
graphics pipeline, rendering speed, 182
graphics pipeline, RGB signal, 61
graphics pipeline, several, 140
graphics pipeline, software
 synchronization, 141
graphics pipeline, stages, 118
graphics pipeline, starved stage, 120
graphics pipeline, Sun Blade 1000, 136
graphics pipeline, synchronization, 76,
 141, 143, 154, 168
graphics pipeline, throughput, 89, 120, 198
graphics pipeline, transform limited, 121,
 125, 166
graphics pipeline, unbalanced, 125
graphics pipeline, video synchronization,
 142
graphics pipeline, view-frustum culling,
 214
graphics platform, 133
graphics programming, 212, 369
graphics programming language, 160
graphics quality, 20
graphics realism, 123, 127
graphics renderer, 9
graphics rendering, 117, 236, 256, 373
graphics rendering, field-sequential, 142
graphics rendering, stages, 117
graphics simulation, 356
graphics simulator, 364
graphics texturing, 299
graphics workstation, 10, 18, 66, 89, 146
graphics workstation, multi-pipe, 80
graphics, field-sequential, 83
graphics, monoscopic, 83
graphics, over-imposed, 65
graphics, temporal-interlaced format, 72
grasp configuration, 95
grasp type, 97
grasp-strengthening exercise, 310
grasping, 196, 256, 310

grasping force, 95, 263
grasping geometry, 95, 96
grasping gesture, 178
grasping strength, 310, 311
gravitational acceleration, 39, 317
gravity, 316
gravity compensation, 105
Greenhaigh, 153
Griffith, 326
Grimsdale, 9
grip strength, 309
grocery store, 362
Groen, 275
gross national product, 350
Gruenbaum, 363
GT1000, 130
gunner, 329, 338
gunner director, 337, 338
gunner director platform, 338
Gypsy, 23, 29
gyroscope, 24, 38–40
gyroscope, bias, 39
gyroscope, drifted data, 39
gyroscope, misalignment, 39

Haines, 117–119, 121, 122, 168, 171, 178,
 180, 199, 212, 214
half-mirror, 62, 275, 299, 381
Hall effect sensor, 24
hallway, 197
hand, 29, 33, 49, 93, 97, 109, 175, 176,
 178, 263, 275, 297, 299, 301, 309,
 310, 381
hand gesture, 16, 364
hand haptic sensing, 49
hand measurement, 53
hand motion, 47, 311
hand segment, 216
hand size, 8, 48, 50
hand strength, 95
hand-supported display, personal, 67
hand tracking, 10
hand tremor, 162
hand, 3D model, 177
hand, 3D position, 107
hand, closure, 108
hand, non-dominant, 313
hand, orientation, 176
hand, sensorial bandwidth, 93
hand, virtual, 100, 117
hand–eye coordination, 275, 276, 301, 306,
 308

hand-supported display (HSD), 60, 65, 66, 68
Hannaford, 249
Hanrahan, 137, 139, 146, 147
Hansen, 36, 37, 39
Hanson, 334
hapthai, 92
haptic, 231
haptic arm, 102
haptic bump, 99, 192
haptic click, 187, 233
haptic clipping, 181
haptic command, 99
haptic control loop, 105
haptic cue, 265
haptic degrees of freedom, 104
haptic display, 57, 126, 184
haptic effect, 231, 326
haptic event, 102
haptic feedback, 16, 92, 93, 110, 118, 128, 218, 219, 243, 249, 260, 262, 263, 265, 266, 293, 294, 299, 301, 309, 329, 338, 351
haptic feedback, illusion, 265
haptic feedback, interface, 58, 93
haptic fidelity, 235
haptic glove, 53, 184, 191, 192, 231, 308, 371
haptic illusion, 267
haptic information, 266
haptic interaction, 353
haptic interface, 92, 93, 95, 97, 99, 105, 106, 109, 110, 150, 184–186, 188, 191, 193, 231, 232, 234, 250, 262, 268, 297–299, 353, 363, 381
haptic interface characteristics, 126
haptic interface controller, 143
haptic interface point (HIP), 138, 139, 191
haptic interface point, ideal (IHIP), 192
haptic interface, control, 188
haptic interface, instability, 189
haptic interface, maximum force capability, 185
haptic interface, saturated, 185
haptic interface, time delay, 190
haptic mesh, 191, 192
haptic pop, 297
haptic rendering, 180, 189
haptic ridge, 192
haptic scene, 233
haptic software, 105
haptic stimuli, 97

haptic system, 271
haptic technology, 6
haptic texture, 99, 192, 193
haptic texture, directional, 192
haptic texture, sinusoidal function, 193
haptic texturing, 126, 180, 192
haptically enabled window border, 99
HapticMaster, 106, 107
HapticMaster, admittance control, 106
HapticMaster, control bandwidth, 106
HapticMaster, inertia, 106
HapticMaster, maximum output force, 106
HapticMaster, mechanical bandwidth, 106
HapticMaster, stiffness, 106
HapticMaster, work envelope, 106
haptics, 5, 117, 125, 326, 369
haptics computation, 140
haptics environment, 232
haptics loop, 231, 233, 234
haptics output display, 126
haptics process, 231
haptics programming, 212
haptics rendering pipeline, 117, 125, 126,
 143, 154, 180, 181, 184, 192, 231, 373
haptics rendering pipeline, first stage, 125,
 182
haptics rendering pipeline, force limited,
 126
haptics rendering pipeline, optimization,
 126
haptics rendering pipeline, second stage,
 126
haptics rendering pipeline, stages, 126
haptics rendering pipeline, third stage, 126
haptics technology, 126
haptics, passive, 340
harbor model, 336
harbor pilot, 336
hard disk, 128, 133
hard drive, 127
hardness, 13
hardware, 7, 9, 10, 57, 68, 80, 121
hardware architecture, 290
hardware filter, 45
hardware latency, 256
hardware setup, 229
Harm, 269, 271
Harp, 214
Hartenberg, 172
Hash, 272
Hatada, 58, 59
Hattori, 1, 2

Hays, 336
head, 8, 29, 33, 74, 84–86, 88, 178, 270,
290, 314, 318, 334, 379
head, height, 65
head, model, 86
head, motion, 78
head, movement, 196
head, position data, 89
head, radius, 85, 86
head, restraint, 62
head, tracker, 18, 62, 70, 379
head, tracking information, 89
head, translation, 88, 270
head, trauma, 290
head-attached system of coordinates, 85
head-mounted display (HMD), 1, 4, 6, 7,
9–11, 13, 18, 24, 33, 37, 59, 60,
65–68, 74, 84, 89, 128, 157, 175, 176,
212, 218, 219, 226, 238, 247, 261,
268, 270, 290, 311, 313, 316, 324,
326, 328, 329, 336–338, 340, 353,
354, 360, 361, 363, 364
head-mounted television, 4
head-related impulse response, 88
head-related transfer function (HRTF), 88,
89, 91
headphones, 18, 84, 89–91, 128, 263, 264
headset, 88
hearing, 299
heat, 101
heat effect, 328
heat pump, 100, 101
heat sink, 100, 101, 130, 131
heat source, 100
heat transfer, 101
heel, 29
Heilig, 3–6
helicopter, 331, 332, 334, 342
helicopter flight simulator, 274
Helsel, 285
Hentschke, 70
Hercules 3D Prophet, 149
heritage site, 321
Hewlett-Packard, 119, 135, 184, 214
Hewlett-Packard 9000, 8, 10, 120, 121
Hewlett-Packard J500, 75
hierarchical structure, 177
high frequency stimuli, 94
high-speed electronics, 65
Hightower, 17
hinge, 357
hip, 23, 95

hip proprioceptive sensor, 95
Hirshfeld, 294
Hirzinger, 44
Hix, 250, 251
HMD, AMLCD display, 63
HMD, blurred pixels, 60
HMD, comfort, 61
HMD, consumer grade, 60
HMD, control unit, 61
HMD, controller, 61
HMD, CRT, 65
HMD, CRT display, 60
HMD, CRT-based, 21, 64, 65, 329
HMD, cy-visor, 62, 63, 72
HMD, Datavisor HiRes, 64, 65, 67, 72
HMD, difuser sheet, 60
HMD, display, 61
HMD, display pixel, 60
HMD, display technology, 60
HMD, electromagnetic radiation, 66
HMD, ergonomics, 11
HMD, eye relief, 63
HMD, EyePhone, 61, 316
HMD, field-sequential video input, 64
HMD, focal distance, 65
HMD, FOV, 60, 61, 63, 65, 74
HMD, granularity, 60
HMD, graphics resolution, 66
HMD, head support, 61, 63
HMD, headphones, 61, 336
HMD, I-glasses, 75, 364
HMD, image panel, 60
HMD, image resolution, 11, 12, 60, 61, 65
HMD, integration, 61
HMD, IPD, 63
HMD, IPD adjustments, 65
HMD, LCD, 65
HMD, LCD display, 60
HMD, LCD panel, 63
HMD, LCD shutters, 64, 65
HMD, LCD-based, 61, 62, 74
HMD, low-cost, 39
HMD, low-resolution, 60, 275
HMD, misaligned optics, 271
HMD, monochrome CRT, 64
HMD, monoscopic, 61, 254, 255
HMD, NTSC video input, 61
HMD, Olympus, 110
HMD, Olympus Eye-Trek, 62, 72
HMD, optics, 60, 61, 65
HMD, PAL video input, 61
HMD, panel display, 61

HMD, panel image, 60
HMD, professional-grade, 61, 63
HMD, ProView XL35, 63–65, 72
HMD, ProView XL50, 337
HMD, RGB video input, 61
HMD, RGB-to-NTSC/PAL conversion, 61
HMD, safety concerns, 65
HMD, semi-transparent, 360
HMD, stereo, 16, 61, 64, 265, 299, 334
HMD, stereo FOV, 63
HMD, stereoscopic viewing, 61
HMD, tracked, 35, 271, 326, 331, 361, 362
HMD, tracker, 18
HMD, undistorted optical path, 65
HMD, Virtual Research FS5, 361
HMD, visual artifacts, 63
HMD, weight, 61, 64, 65, 361
HMD, x-cube optical path, 63
HMD, XGA resolution, 63
HMD, XVGA resolution, 60
Ho, 191, 192
Hodges, 278, 311, 312
Hokkaido University, 353
Holden, 308, 309
hole, 307
Holland, 329
Holloway, 175, 176
home theater electronics, 90
homogeneous transformation matrix, 172, 177
homogeneous transformation matrix, compounded, 172
homogeneous transformation matrix, inverse, 172, 174
homogeneous transformation matrix, scaling, 174
homogeneous transformation matrix, time dependent, 175
Hooke’s law, 126, 185, 188, 190, 263
hospital, 298, 300, 304
hospital-acquired infections, 297
host computer, 7, 18, 20, 22–25, 29, 33, 34, 37–41, 44, 45, 48–54, 89, 98, 100–102, 105, 107, 109, 143, 338, 379
host interface processor (HIP), 137
host PC, 23, 306, 364
house, 3
Howe, 95, 96, 248
HRIRs, 88
HSD, graphics resolution, 66
HSD, miniature CRT, 65
HSD, on/off button, 65

HSD, optics, 65
HSD, optics weight, 66
HSD, virtual binoculars, 65, 66, 72, 336,
 337
Hsu, 184
HT Medical Systems Inc., 294, 297
Hughes, 314
Hughes Aircraft Co., 324
human anatomy, 197, 287
human behavior, 194, 244
human body, 288
human error, 251, 338
human eye, 198
human face, 162
human factors, 93
human factors issues, 13
human factors research, 243, 244, 250,
 276, 280
human factors study, 74, 116, 244–247,
 250, 253, 276, 292, 293, 295, 299,
 315, 317, 334, 342, 370, 381
human factors study, between-subjects
 design, 246, 252, 255, 257, 263, 272,
 336
human factors study, case study, 245
human factors study, consent form, 246
human factors study, control study, 245
human factors study, experimental
 protocol, 244
human factors study, Institutional Review
 Board (IRB), 246
human factors study, principal investigator,
 246
human factors study, stages, 245
human factors study, subject, 245
human factors study, subject privacy, 246
human factors study, subject recruitment,
 246
human factors study, subject screening,
 246, 247
human factors study, subjective evaluation,
 246, 265, 272, 290, 315, 357, 361
human factors study, task debriefing, 246
human factors study, testbed, 253–255
human factors study, within-subject design,
 246, 252, 255, 258, 259, 261, 266, 336
human haptic characteristics, 93
human haptic sensing, 100
human haptic system, 93
human vision system, 58, 379
human visual perception mechanism, 58
human–computer interaction, 16

human-computer interaction technique, 376
human-computer interface, 1
human-machine interaction, 244
humanoid model, 356, 357
Humphreys, 137, 139, 146–148
Hunter, 95
Huron 20, 89–91
Huron bus, 89
hybrid tracking approach, 41
hyperactivity, 314
hysteresis, 34

I/O card, 10, 89
I/O communication, 117, 157
I/O controller hub, 128
I/O device, 13, 16, 45, 46, 127, 128, 151, 157, 177, 211, 212, 214, 219, 226, 239, 243, 253, 364
I/O device, mapping, 157
I/O hub, 133
I/O processor, 10
i860, 10
IBM, 319
IBM GT1000, 129
IBM RC1000, 129, 130
icon, 10, 97, 99, 374, 381
ID8 Media, 23
identity matrix, 174
IEEE-488, 32
illumination, 122, 351
illumination, global, 164–168
illumination, global radiosity, 167, 355
illumination, local, 164–166, 190, 199
illumination, model, 166
illumination, Phong model, 122, 123
illumination, poor, 367
Illusion Systems Inc., 326
image, 1, 7, 11, 18, 21, 58, 59
image, adaptation, 70
image, aliased, 118
image, anti-aliasing effect, 118
image, antialiased, 118
image, artifacts, 75
image, autostereoscopic, 70
image, binocular, 60
image, blending, 81, 82, 146
image, boudary subpixel, 70
image, brightness, 81
image, color uniformity, 81
image, column interlaced format, 68, 70
image, composite, 75, 81

image, correctly aligned, 76
image, depth cue, 59
image, DVD, 75
image, exaggerated response, 74
image, flicker-free, 74
image, floating, 60, 68
image, four-corner region, 81
image, frame-sequential mode, 76
image, generator, 83
image, generator array, 81
image, granularity, 63, 65, 81
image, high-resolution, 59, 76, 78
image, high-resolution stereo, 78
image, horizontal shift, 58
image, large surface, 81
image, luminance, 76
image, luminosity, 81
image, misalignment, 76
image, monocular, 60
image, monoscopic, 60, 70
image, occlusions, 59
image, overlap zone luminosity, 81
image, parallax, 68
image, perceived quality, 84
image, perspective, 59
image, pre-distorted, 83
image, quality, 81, 118
image, realism, 168
image, recognition stage, 70
image, red green blue components, 65
image, resolution, 81, 83, 131, 136, 146,
 256, 379
image, seamless matrix blending, 81
image, shadows, 59
image, slice, 288
image, spatial multiplexing, 70
image, spatially sequenced, 59
image, stereo, 60, 68, 69, 70, 76, 83, 229
image, stereo effect, 76, 83
image, stereo pair, 68, 69, 72, 76, 83
image, stereo tiled, 142
image, stereo viewing cone, 69
image, surface texture, 59
image, synchronization, 76
image, texture, 127
image, three primary colors, 77
image, tile, 81, 82, 146, 149, 374
image, tile overlap, 81
image, tile pixel intensity, 81
image, time sequenced, 59
image, update, 66
image, VCR, 75

image, virtual, 60, 62, 65
image-understanding, 256
imagination, 3, 4
ImersaDesk, 317
immersion, 3, 4, 18, 275, 285, 317, 319,
 324
Immersion Co., 45, 46, 53, 99, 192, 193,
 294, 304
Immersion Medical, 297, 298
Immersion Probe, 45
immersive VR, 360
Immersive Workbench, 18
Imperial College of Medicine, 302
implicit function, 183
impulse response, 89
incipient malignancies, 293
incremental position control, 42
index, 216
index finger, 292
index fingertip, 97, 292
index of refraction, 49
indexed motion, 43
individual isolation, 278
industrial design application, 134
industrial engineer, 358
industrial equipment, 358
inertia, 13, 193, 265, 276, 364
inertial force, 98, 270
infantry vehicle, 334
Infinite Reality, 124, 125, 137–139, 146
information retrieval, 360
Information Society Initiative, 349
information visualization, 371, 374
information visualization pipeline, 373
initial contact cue, 264
input device, 54, 74, 116, 225, 226, 373,
 381
inspection room, 355
inspection task, 371
Institute for Creative Technologies, 331
Institute for Robotics Research, 370
Institute of Electrical and Electronics
 Engineers, 8, 151
instructional software, 317
instructor, 329, 332, 336, 338–340, 342,
 360
instrument display, 340
instrumented object, 261
integer arithmetic, 211
integration issues, 9
Intel 486, 126
Intel 820/850 chipset, 128

Intel Co., 127, 128
intelligent agent, 289, 317, 332, 357
intelligent behavior, 13, 213
interaction, 3, 4, 8, 46, 244, 254
interaction technique, 246, 250, 253–256,
 266, 276, 280, 381
interaction technique, gaze directed, 254
interaction technique, Go-Go, 253, 255,
 256
interaction technique, HOMER, 253
interaction technique, manipulation-based,
 253
interaction technique, pointing, 254
interaction technique, ray casting, 256
interaction technique, steering-based, 253
interaction technique, target specification,
 253
interaction, collaborative, 149
interaction, cooperative, 149, 151
interaction, dextrous, 47
interaction, gesture based, 46
interaction, mouse-based, 97
interaction, natural, 16, 46
interaction, single point, 47, 190
interaction, trackball-mediated, 65
interactive graphics, 3
interactive sound, 329
interactivity, 2, 3, 16, 54, 203, 285, 299,
 314, 318, 319
interface, 8–10, 16, 44, 289
interface, actuator, 102
interface, box, 49
interface, chip, 119, 120
interface, controller, 143
interface, design, 247
interface, device, 33
interface, electronic unit, 40
interface, functionality, 16
interface, handle, 234
interface, hardware design, 17
interface, hardware functionality, 17
interface, hardware limitations, 17
interface, stylus-based, 191
interface, viscosity, 233
Interfaces for Real and Virtual Worlds, 8
interference, 29
internal organ, 377
Internet, 10, 153, 277–279, 287, 304, 306,
 314, 319, 324, 353
Internet2, 308
interpolated joint motion, 238
InterSense, 39, 41

intravascular catheterization simulator, 297
intravenous procedure, 296, 297
invariant, 173
invasion of privacy, 279
Ipsos Reid, 362
Isdale, 62, 75
ISDN, 353
IV insertion, 298
IV procedure, 297

Jack, 309
JACK, 357, 358
Jandura, 97
Japan, 1, 321, 353, 365, 367, 369, 371
Java, 221, 227
Java 3D, 13
Java 3D, 3D view model, 224
Java 3D, API, 225
Java 3D, applet, 227
Java 3D, branch group, 224
Java 3D, branch graph, 224
Java 3D, Canvas3D object, 226
Java 3D, class, 222, 223
Java 3D, DistanceLOD class, 227
Java 3D, file importing, 222
Java 3D, frame refresh rate, 228
Java 3D, garbage collection, 229
Java 3D, group node, 224
Java 3D, Input Device Interface, 225
Java 3D, input device support, 225
Java 3D, leaf node, 224
Java 3D, loader, 222
Java 3D, Locale node, 223
Java 3D, LOD class, 227
Java 3D, method, 222, 225
Java 3D, networking, 227
Java 3D, object behavior, 226
Java 3D, object scheduling bounds, 226
Java 3D, platform independence, 225
Java 3D, portability, 226
Java 3D, rendering speed, 228
Java 3D, scene graph, 222, 223, 226, 227
Java 3D, scene graph traversal order, 224
Java 3D, Screen3D node, 226
Java 3D, sensor, 225
Java 3D, Shape3D node, 224
Java 3D, subgraph, 224
Java 3D, Switch node, 224, 227
Java 3D, system latency, 228
Java 3D, system response, 228
Java 3D, time-stamped data, 225

Java 3D, View model, 226
Java 3D, View object, 224, 226
Java 3D, ViewPlatform activation radius,
 226
Java 3D, ViewPlatform node, 224, 226
Java 3D, virtual device, 225
Jayaram, 353, 354
Jebens, 1
Jense, 329
Jensen, 340
Jet Propulsion Laboratory, 368, 369
jogging, 236
John Deere Co., 357
Johnes, 95
Johnson, 195, 196
joint, 45
joint angle, 46, 53
joint force commander, 342
joint position, 23, 95
joint position, just noticeable difference,
 95
joint resolution, 46
joint sensor, 46
joint sensor, resolution, 22
Joint Simulation Systems, 342
joint tolerance, 46
joint velocity, 95
Jones, 287, 295
joystick, 1, 47, 102, 104, 106, 107, 128,
 129, 272, 370
joystick, actuator-gimbal assembly, 102
joystick, base, 102
joystick, button, 102
joystick, degrees of freedom, 102
joystick, force feedback, 129
joystick, jolt, 102
joystick, onboard electronics, 102
joystick, WingMan Force 3D, 102, 103
Jules Verne, 4
Julesz, 58
JumpZone, 326, 327
 Dark Ride, 326
Juster, 355

Kaiser Electro-Optics, 11, 12, 63, 64
Kalawsky, 58, 86, 87, 94, 95, 340
Kalman filter, 38, 41
Kameyama, 379
Kanai, 353
Kaufman, 376
Kehres, 339
Kennedy, 272–274

Kestenbaum, 268
keyboard, 8, 16, 324
Kheddar, 367, 370–372
Kim, 367, 368
kinematic model, 22, 205
kinematics modeling, 172
kinematics transformation, time dependent, 176
kinesthesia, 95
kinesthesia, 93
kinesthetic receptor, 100
kinesthetic sensing, 95
knee, 29, 313
Knerr, 328
knowledge attachment, 374, 375
knowledge retention, 314, 316, 317
Koechling, 238, 239
Kogan, 39
Kontarinis, 248
Kopp, 308
Kotoku, 369
Kraemer, 91, 92
Kramer, 52, 53, 108
Krieg, 25, 26
Krueger, 3
Kuwait Theater database, 333
Kuwait Theater of Operations, 332

Lackner, 276
Lake Technology, 89, 91
Lampton, 246, 253
LAN, 107, 109, 129, 146, 148–151, 153, 246, 290, 340
LAN, bottleneck, 149
LAN, communication, 129
LAN, Mytrinet, 146, 149
LAN, pier-to-pier, 152
LAN, traffic, 146, 148
Landmark Graphics Co., 374, 376
Lanier, 7, 8
laparoscope, 301, 303
laparoscope, virtual model, 301
laparoscopic procedure, 303
Laparoscopic Surgical Workstation, 304
laparoscopy, 301, 303
Lara Croft, 324, 325
large-screen visual channel, 76
large-volume display, 11, 13, 16, 81, 83
large-volume display, CAVE, 72
large-volume display, comparison, 83
large-volume display, CRT-based, 72, 73
large-volume display, display wall, 72

large-volume display, Dome, 72, 81
large-volume display, Immersive Workbench, 78
large-volume display, projector based, 72, 78
large volume display, stereo, 72
large volume display, work volume, 72
large volume display, workbench, 72, 79
laser, 268, 319, 363
laser pointer, 268
laser range finder, 364
laser, class 1, 268
laser, class 3a, 268
Latency, 20
latency, 117, 218
latency, acceptable level, 149
lateral-effect photodiode, 38
Lathan, 269
Laurel, 262
Laviola, 47
LCD, 7, 9, 11, 12, 18, 69, 74
LCD display, 63, 76
LCD flat panel display, 67
LCD panel, 67, 69
LCD, TFT, 69
LCOS, 62, 63, 65, 66
LCOS display, 65, 66
LCOS panel, 66
leadership training, 328, 331, 342
Learjet 45, 357
learning, 278, 314
learning, constructionist, 314, 318
learning, constructivist, 314
learning, curve, 258
learning, situated, 314
learning, VR-based, 318
least-squares algorithm, 367
Lecuyer, 265, 267
LED, 24, 35, 37–39, 44, 49, 63, 105, 263, 264, 379
LED, infrared, 39, 268
LED, pulsed, 38
left-eye view, 68
leg, 178, 314, 319
Leifer, 53
LEPD, 38
level flight, 328
level of detail, 205, 224, 365
level-of-detail optimization, 158
Lewiss, 306
lieutenant, 331
ligament sprain, 305

light, 228, 229, 299
light, ambient, 76, 122, 123, 135
light, beacon, 35
light, column source, 69
light, diffuse, 215
light, direct, 122
light, energy flow, 167
light, external, 63
light, guide, 69
light, high intensity, 268
light, high-intensity source, 77
light, infrared, 35, 268
light, intensity, 122, 164, 166, 170
light, intensity interpolation, 165
light, intensity value, 166
light, IR, 36
light, IR beam, 37
light, IR filtering lens, 38
light, LED source, 66
light, local source, 132
light, loss, 62
light, maximum number of sources, 123
light, measurement, 321
light, multiple sources, 122, 123
light, narrow source, 69
light, plane, 36
light, point source, 122, 123
light, pulsed, 268
light, reflected, 122, 123
light, reflection, 355
light, shutter filtered, 74
light, single source, 122
light, source, 63, 121, 123, 130, 136, 164,
 166, 170–172, 228–230, 268, 319, 355
light, source type, 118
light, specular reflection, 168
light, spot, 38, 170
light, striped, 355
light, transmitted, 49
light, virtual sun, 168
Light-04, 134
lighting, 224
lighting computation, 117, 118, 130
lighting model, 118
Lightscape, 134
limb motion, 271
Lin, 181–183
Lindheim, 331
liquid crystal layer, 63
liquid-crystal-on-silicon, 62
Liu, 376
live dancer, 196

live fire training, 336, 337
live reckoning, 238, 239
load distribution, 373
local area network (LAN), 107, 127
local client, 153
local magnetic North, 39
local simulation, 153
location-based entertainment, 326
locomotion, 17
LOD management, 200, 205
LOD management, adaptive, 199, 200,
 202, 204
LOD management, alpha blending, 199
LOD management, cycling, 199
LOD management, discrete geometry, 198
LOD management, edge collapse, 199
LOD management, hysteresis buffer, 199
LOD management, morphing, 199
LOD management, popping artifact, 199
LOD management, static, 198–200, 202
LOD segmentation, 201
Loftin, 316
Logitech Co., 97, 98, 102, 103
Longorio, 374, 375
look-up table, 50, 81, 89, 91, 148, 184
Lorensen, 287, 288
loudspeaker, 88, 90, 91
low-frequency stimuli, 94
low-level control, 358
Lowe, 277
lower abdomen model, 292
lower back, 357
lower intestine, 293
lower precision arithmetic, 122
LRP glove, 371
Lubell, 77
LucasArts Entertainment, 324
Luebke, 198
lumbar anesthesia, 297
lumen, 76, 78
Lycra suit, 23

Macedonia, 153
machine reasoning, 365
Machover, 11, 12
magnetic field, 24, 31, 46, 66, 98, 140
magnetic field, induced, 31
magnetic interference, 329
magnetic particle brake, 297
magnetic permeability, 31
magnetic resonance imaging, MRI, 287
magnetic sensor, 18

magnetometer, 39
mailbox, 308, 309
mains frequency, 31
maintenance task, 360
maintenance training, 361
maintenance worker, 360
Majoros, 360, 361
Mako aircraft, 339, 340
Management, Command and Control
 workstation (MCC), 332, 333
manipulation, 41–44
manipulation, Cubic-Mouse, 43
manipulation, interface, 41–43
manipulation, task, 100, 275
manipulator, 363, 364
mannequin, 295
manual digitization process, 160
manual force exertion capability, 93, 95
manual recording, 246
manual skill, 293
manually controlled action, 176
manufacturer, 350
manufacturing, 285, 349, 350, 358, 362,
 364
manufacturing applications, 286
manufacturing facility, 353
manufacturing training, 361
Map View, 203
marble, 319
Marchese, 296
marching-cubes algorithm, 287
marketing, 286, 350, 351, 358, 360, 362
marketplace conservatism, 285
Mars, 368, 369
Mars Pathfinder, 369
Martens, 69
Martin, 98
Mary Magdalene, 319
Massachusetts Institute of Technology, 308
Massie, 104
Massimino, 256
master, 29, 142
master arm, 107
material flow simulation, 358
material texture, 321
Mathematica-to-DIVE converter, 315
matrix, 177
matrix product equation, 177
matrix transformation, 315
mausoleum, 320
Mavor, 59
Maxwell World, 316, 317

Maya, 322
McAllister, 166
McDonough, 332, 333
McDowall, 46, 48, 66, 67
Mead, 22
measurement accuracy, 18
measurement range, 17
mechanical amplifier, 107
mechanical arm, 4, 21, 22, 24, 45, 66, 67,
 109, 110, 160, 329, 362
mechanical arm, inertia, 24
mechanical arm, motion interference, 24
mechanical arm, oscillations, 24
mechanical arm, weight, 24
mechanical assembly, 357
mechanical bandwidth, 102
Mechanical CAD, 134
mechanical design, 140, 160
Mechanical Dynamics Inc., 354
mechanical support structure, 66
mechanical texturing, 143
mechanical training box, 303
mechanism simulation, 357
mechanoreceptor, sensing bandwidth, 97
media control processor (MCP), 133
medic, 324
medical education, 286, 287
medical informatics, 287
medical student, 299
medical training, 287
medicine, 3, 8, 286, 287, 342
Medicine Meets Virtual Reality, 8
MedMCAD-01, 134
Meissner corpuscle, 94
memory, 116, 144, 169, 364, 376–378
 memory, bus speed, 131
 memory, clock, 131
 memory, controller, 133
 memory, controller chip, 128
 memory, direct access, 138
 memory, direct burst, 136, 145
 memory, double-data rate (DDR), 130,
 131, 133, 379
 memory, frame buffer, 136, 170, 227, 376
 memory, graphics, 127
 memory, hard disk, 197, 202
 memory, multiple calls, 136
 memory, page, 381
 memory, page fault, 205
 memory, primary cache, 136
 memory, RAM, 202
 memory, SDRAM, 379

memory, secondary video, 128
memory, shared, 133, 147, 167, 168
memory, swap, 197, 202
memory, system RAM, 127, 128, 138, 205,
 214
memory, texture, 136, 145, 169, 227
memory, texture cache, 120
memory, transfer, 127
memory, TwinBank architecture, 133
memory, unified, 136
memory, video, 128
Mentice Medical Simulations, 301
merge tree, 199
Merkel disk, 94
message passing, 177
metal, 27, 31, 32, 35, 39
metal, interference, 32, 62
metallic object, 26, 31
metallic shininess, 355
Mi-24 Hind, 342
Michelangelo, 319, 320
microelectromechanical systems (MEMS),
 38
micromirror, 78
microphone, 33, 39, 88, 89
microphone array, 306
micropin array, 97
microprocessor, 27, 29, 36, 37, 48, 98
MicroScribe 3D, 45, 46
MicroScribe 3D, calibration, 160
MicroScribe model, 46
Microsoft Co., 132, 324
microswitch, 24
MIG-29, 342
migraine, 268
mild steel, 31, 32
military, 3, 6, 7
military applications, 286
military branch, 328
military command and control, 140, 197,
 250, 251, 253
military commander, 251
military hardware, 328
military planner, 333
military simulation, 212, 285
military training, 235, 246, 268, 286, 328
milling, 352
milling machine, 351, 352
mine, 331
mine, 3-D model, 331
mine, detection, 329
miniature wearable display, 268

Minimally Invasive Surgical (MIS) Trainer, 301, 304
Ministry of International Trade and Industry, 367
Minsky, 193
Mira Imaging Inc., 160
missile, 329
missile tracking, 17
mission debriefing, 334, 340
mission performance, 342
Mission Rehearsal Exercises, 331
mission scenario, 336
MIST-VR, 301–304
MIST-VR, training, 304
MIST-VR, validation studies, 302
MIT Media Laboratory, 183, 195
model complexity, 125, 197
model management, 157, 197
model management, cell segmentation, 197, 203
model management, database management, 197
model management, level of detail (LOD), 164, 198
model transformation, 117
model, 2D, 170
model, airplane, 164
model, automatic segmentation, 202, 295
model, candle surface, 170
model, cell, 202
model, cell segmentation, 203, 204
model, clay, 160
model, complex, 160
model, database management, 204, 205
model, hierarchy levels, 177, 178
model, high LOD, 205
model, house, 160
model, kinematics, 178
model, mesh simplification, 199
model, monolithic, 204
model, opacity, 199
model, optimized, 157
model, partition priority, 203
model, partitioning, 202
model, polygonal, 163
model, preexisting, 160
model, purchasing price, 164
model, scaling, 160
model, segmentation, 202, 222
model, segmented, 178, 323

model, several LODs, 199
model, shaded, 164
model, spline-based, 163
model, transparency, 199
model, tree, 170
model, wireframe, 164
modeling, 157
modem, 38, 150, 151, 338
molecular docking force, 6
Möller, 117–119, 121, 122, 168, 171, 178,
 180, 199, 212, 214
Mon-Williams, 276
monitor, 11, 72, 74, 75, 84, 91, 140, 142,
 144, 145, 261, 293, 297, 301, 370
monitor, CRT, 74, 84, 142
monitor, external synchronization signal,
 142
monitor, horizontal line retracing, 142
monitor, master, 142
monitor, refresh rate, 74
monitor, resolution, 131
monitor, scan rate, 72
monitor, side-by-side, 75
monitor, stereo, 31, 75, 129
monitor, stereo mode, 129
monitor, stereo-ready, 72
monitor, vertical line retracing, 142
monitor, vertical refresh rate, 31
monitor, video logic circuitry, 142
Monkman, 100
monoscopic graphics, 258, 259, 324
Montpellier, 8
Montrym, 137, 138
Moore’s law, 10
Morgenbesser, 190
Mortensen, 166
Moshel, 314
mosque, 321
motherboard, 140
motion, 3
motion capture, 29, 35
motion capture suit, 23
motion command, 369
motion parallax, 59
motion platform, 268, 270, 326, 338
motion restriction board, 293
motion sequence, 323
motion sickness, 269, 271
motion trajectory, 308
motor, 381
motor shaft, 99
motor skill, 301, 304

motorcycle ride, 3
motorized unit, 328
Motorola, 360
mouse, 8, 13, 16, 42, 43, 66, 97, 117, 129,
 160, 192, 225, 235–237, 261, 279,
 314, 315, 324, 376
mouse, actuator, 192
mouse, base, 98
mouse, button state, 97
mouse, electrical actuator, 97
mouse, haptic, 193
mouse, iFeel, 97, 98, 100, 192
mouse, optical, 98
mouse, optical sensor, 98
mouse, pad, 98
mouse, position, 97
mouse, processor, 99
mouse, standard, 97
mouse, tactile, 97, 110
mouse, translation, 98
mouse, 3D, 33, 261
movie industry, 256
MRI imaging, 376
Müller, 1
multi-user interaction, 117, 211
multi-user networking, 285
multicast communication, 151, 153, 290
multimedia simulation, 12
multimodal interface, 243
multimodal teaching advisor, 364, 365
multimonitor display, 140
multipanel display, 75
multipipeline synchronization, 141
multiple-instructions, multiple-data
 parallelism (MIMD), 138, 158
multiplexer, 27, 49, 52
multiplexing chip, 48
multitasking environment, 136
multiuser environment, 90
multiuser simulation, 149, 239, 373
multiview cable, 142
multiview display, 140
muscle, 93
muscle contraction, 95
muscle fatigue, 95
muscle fiber, 95
muscle length, 95
muscle ligament, 93
muscle pain, 268
muscle spindle, 95
muscular exertion force, 94
mutual interference, 50

n-vision Inc., 64, 65, 72
Nadeau, 224, 226
Naeve, 315
Naimark, 319
Nash, 362
National Aeronautics and Space Agency (NASA), 7, 10, 66, 68, 88, 89, 316, 367, 369
National Guard, 334
National Institute of Safety and Health, 357
National Library of Medicine, 287, 288
National Rural Health Association, 304
natural motion, 46
nausea, 269
naval artillery, 334
Naval Command, Control and Ocean Surveillance Center (NCCOSC), 334
Naval Postgraduate School, 153
Naval Research Laboratory, 251–253
Naval Submarine School, 334
navigation, 16, 17, 41–44, 252–254, 270, 279, 360, 378, 381
navigation, active control, 272
navigation, active-passive control, 272
navigation, algorithm, 296
navigation, degrees of freedom, 252, 272
navigation, egocentric, 251, 252
navigation, exocentric, 251, 252
navigation, fly-by, 43, 44
navigation, indirect, 254
navigation, interface, 41, 42, 47, 252
navigation, maze, 272
navigation, metaphor, 252
navigation, motion dynamics, 270
navigation, obstacle, 253, 272
navigation, open-loop, 97
navigation, passive control, 272
navigation, passively-experienced, 271
navigation, technique, 271
navigation, under user control, 271
Navy, 328
Navy, British Royal, 336–338
Navy, US, 334
needle, 297–299
needle, holder, 299
needle, insertion, 297
needle, placement accuracy, 299
needle, virtual, 297
Neely, 51
Neff, 374, 375
Netherlands, 106
network, 149, 150, 238, 332, 333

network, architecture, 151
network, area of interest management (AOIM), 153, 227
network, bandwidth, 150, 151, 153
network, breakdown, 332
network, buffer, 148
network, central server, 151, 153
network, central server bottleneck, 151
network, client peers, 153
network, computer security, 153
network, congestion, 152
network, delays, 238
network, DiveBone, 153
network, eavesdropping, 333
network, hybrid, 153
network, logical connection, 150, 152
network, message traffic, 153
network, Multicast Backbone (MBone), 153
network, packet wrapping, 153
network, physical connection, 150, 151
network, pier-to-pier, 153
network, proxy server, 153
network, ring, 152
network, router, 153
network, traffic, 151, 332, 333
network, type, 150
network, unmatched servers, 152
networked computer, 116, 149, 373
networked simulation, 338, 340
networked virtual environment, 227
networking, 212
Neumann, 360, 361
neural circuitry, 268
neural conflict, 270
neural network, 314
neural pathway, 269
neuro-muscular stimulation, 97
neurohormonal substance, 271
neuroscience, 288
New Jersey Institute of Technology, 309
New York, 3
New Zealand, 30
newborn, 298
Newell, 168
Newton World, 316
Newtonian motion, 316
Newtonian Physics, 316
NICE, 317
Nintendo, 3, 9
Nixon, 30–32
non-ferromagnetic metal, 31, 32

nondominant hand, 43
nonisotropic volume, 376
nonlinear calibration algorithm, 367
normal force, 234
North, 311
North America, 31
Northern Ireland, 303
nose, 85
Notre Dame Cathedral, 321–323
NTCS, 18
NTT, 364
nurse, 296
nurse, IV skill, 297
nursing staff, 298
NVIDIA Co., 129, 131, 133, 146, 171,
 172, 338
NVIDIA graphics card, 80
NVIS Inc., 65, 72

O'Toole, 299, 300
oar, 326
object, 22, 34, 42, 158, 182, 212, 214, 381
object, 3D, 78, 181, 182
object, alpha blending, 198
object, appearance, 157, 164, 172, 192,
 212, 215, 222, 223
object, away from viewer, 59
object, axes of symmetry, 172
object, battleship, 163
object, behavior, 157, 168, 180, 194, 205,
 212, 219, 233
object, center of gravity, 172, 173
object, centroid, 183
object, child, 172, 177, 178, 210, 212
object, class, 211
object, clipped, 180
object, coexisting LODs, 199
object, cold, 192
object, compliance, 180
object, contact, 182, 234
object, coordinate space, 168
object, deformation, 186
object, deletion, 152
object, deselected, 43
object, detail, 59
object, dextrous manipulation, 95
object, diffuse color, 123
object, digitized, 45
object, discrete geometry, 198
object, distant, 59, 163, 180, 198, 256
object, dynamic, 218
object, elastic, 184–186

object, elastically deformed, 187
object, fast-moving, 182
object, file, 160, 174
object, furniture, 163
object, geometry, 164, 183, 198, 212, 215,
 222
object, grasped, 95, 176, 184, 254,
 258–260, 265
object, hard, 105
object, harder interior, 186
object, hardness, 157
object, hierarchy, 177
object, highlighted, 255
object, homogeneous, 185
object, horizontal shift, 59
object, house, 161
object, immovable, 106
object, in contact, 180
object, inertia, 93, 108, 157, 180, 265
object, initially deformed, 188
object, insertion, 260
object, instance, 177
object, intelligence, 194, 290
object, inter-reflections, 165, 166
object, interactive, 194
object, interdependencies, 165, 166
object, jittery, 43
object, kinematic constraints, 157
object, kinematics, 172, 184
object, level of autonomy, 194
object, level of detail, 160, 164, 197, 198,
 227
object, location, 172
object, manipulation, 17, 41, 47, 253, 260,
 265
object, material identification, 100
object, material properties, 215
object, metallic, 32
object, model, 376
object, modeling, 13
object, monolithic model, 176
object, morphing, 198
object, moving, 17, 20, 21, 24, 25, 41, 160,
 182, 183
object, non-rigid, 183
object, non-uniform hardness, 186
object, not homogeneous, 185
object, occluded, 92
object, orientation, 17, 39
object, orientation rate of change, 38
object, palm, 213
object, parallax, 59

object, parametric coordinates, 168
object, parent, 172, 177, 178, 210
object, parent-child dependency, 222
object, parent-child hierarchy, 177, 224
object, parent-child relation, 172
object, perspective projection, 180
object, physical characteristics, 125, 157,
 180, 184
object, physical model, 192
object, placement, 254
object, placement task, 258, 260
object, plastic, 187
object, plastically-deformed, 184
object, polygonal mesh, 184
object, position, 17, 20, 22, 42, 44, 173
object, positioning, 255
object, primitive, 119
object, radiosity, 166
object, real, 160, 261, 311, 357
object, real-time position, 35
object, relative velocity, 182
object, rendering cost, 201
object, rigid, 93
object, rotation, 17, 172
object, scaling, 172, 174
object, scanned, 162, 164
object, selection, 41, 251, 253–255
object, shaded, 168
object, shadow, 166
object, shape, 123, 157–160, 222
object, shared group, 221
object, shiny surface, 123
object, size, 256
object, slippage, 93
object, slippery, 192
object, smoothness, 166
object, state update, 221
object, static, 173, 218
object, stationary, 20, 184
object, statue, 161, 163
object, stiffness, 185
object, surface, 45, 122, 160, 162, 164,
 173, 183
object, surface compliance, 93
object, surface geometry, 232
object, surface properties, 168
object, surface reflectivity coefficient, 164
object, surface roughness, 93, 180
object, surface temperature, 93, 100
object, symmetrical, 160
object, teacher, 308, 309
object, teapot, 166

object, tessellated, 123, 190
object, thermal characteristic, 100
object, thermal conductivity, 100
object, thermal diffusivity, 100
object, thermal signature, 100
object, tracked, 35, 36, 39
object, translation, 17, 172–174
object, transparency, 223
object, traversal, 301
object, value, 201, 202
object, vertices, 183
object, virtual, 93, 97, 100, 149, 311
object, visible, 212
object, voxelized, 378
object, weight, 93, 108, 126, 157, 180, 265
object-oriented function, 211
object-oriented programming, 221
objective measure of performance, 303
obsessive-compulsive disorder, 311
obstacle, 363, 364
occlusion, 364
occlusion factor, 204
oculomotor problem, 269
off-centered mass, 99
office, 197
office room, 339
Officer on Deck training, 334
offline processing, 35
oil and gas industry, 374
oil exploration, 140, 349, 373
oil production, 374
oil reservoir, 374
oil tanker, 276
Okino Inc., 161, 162
Olympus Co., 62
online medical education, 287
online society, 279
Onyx, 124
Onyx2 R10,000 CPU, 139
open kinematic chain, 46
open kinematic link, 66
open-loop control, 93
OpenGL, 118, 129, 134, 136, 147, 148,
 164, 178, 210, 221, 231, 235, 292, 381
OpenGL, encoded command, 148
operating system, 169
operating table, 299
operational amplifier, 100
operator, 364, 367
operator, novice, 365
operator, training, 367
optical distortion, 270

optical encoder, 105
optical sensor, 301
optics, 4, 7, 315
opto-electronic connector, 49
opto-mechanical shaft encoder, 66
optoelectronics circuitry, 49
organ malignancy, 291
organ, behavior, 299
organ, removed, 298
Ouh-young, 104
outer space, 7, 368
outpatient clinic, 306
output device, 16, 110
overlaid graphics, 1
oversampling technique, 25
Oyama, 367

Pacinian corpuscle, 94, 95
packet transmission, 151
paddle display, 374
Pager Assembly Line, 360
pallet, 358
palm, 46–48, 50, 52, 53, 94, 95, 98, 99,
 178, 216, 265, 290, 306, 309
palpation, 289
palpation, force feedback, 293
Panoram Technologies, 75, 81, 82
PanoWall, 81, 146
PanoWall, processor array, 81
PanoWall, single projector, 81
PanoWall, three projectors, 82
PanoWall, visual artifact, 81
Papagiannakis, 321, 322
Pape, 78, 80
parachute, 326, 328
parachute simulator, 327
parallel architecture, 9, 118
parallel kinematic mechanism, 102
parallel kinematic structure, 22
Paramont Pictures, 324
Paris Air Show, 340
Pasquier, 329, 330
passenger, 270
path, 194, 195, 236, 237
path, waypoint, 236
patient, 290, 291, 293–300, 304, 306, 308,
 310–312
patient, anatomy, 297
patient, anesthesia, 293
patient, appearance, 290
patient, baseline, 306
patient, body, 301

patient, chronic post-stroke, 308, 309
patient, cognitive deficit, 308
patient, complications, 295
patient, confidentiality, 311
patient, database, 287
patient, discharged, 304
patient, geriatric, 298
patient, home, 306
patient, orthopedic, 306
patient, performance, 307, 309, 311
patient, phobia, 311
patient, range of motion, 306
patient, reactions, 294, 298
patient, risk, 297
patient, stroke, 308, 310
patient, virtual, 287, 290, 295, 297, 298
pattern recognition, 331
PC, 11, 81, 89–91, 99, 126–129, 135, 139,
 140, 144, 148, 211, 289, 294, 295,
 297, 299, 301, 306, 310, 311, 315,
 324–326, 338, 342, 353, 360, 379
PC bus, 89
PC, architecture, 127
PC, available AGP slots, 146
PC, Chromium cluster, 146, 147, 149
PC, client, 151
PC, cluster, 146–149, 154
PC, game port, 128
PC, graphics, 11
PC, graphics accelerator, 10, 138
PC, graphics architecture, 136
PC, graphics board, 10
PC, graphics card, 129, 131
PC, graphics hardware, 137
PC, hardware, 10
PC, high end, 129
PC, host, 105, 143
PC, local, 151
PC, low-cost, 146
PC, master, 142
PC, motherboard, 130, 146
PC, operating system, 140
PC, parallel port, 128
PC, programmable I/O (PIO) port, 128
PC, remote, 151
PC, screen, 9
PC, serial port, 128
PC, server, 149
PC, slave, 142
PC, sound card, 92
PC, USB port, 128
PDP 11-40, 7

peacekeeping incident, 331
peacekeeping operation, 328, 331
peak force, 300
peg, 306, 307
peg-in-hole insertion, 262, 370
pegboard exercise, 307, 308
Peltier principle, 100
Peltier pump, 100
pencil, 234, 351
pendulum, 316
penetration distance, 187, 190, 192
Pentium, 227
Pentium 4, 133
Pentium II, 143
Pentium III, 131, 133, 146, 149
Pentium, dual processor, 338
Pentland, 183
people, 236–238
PeopleShop, action bead, 236, 237
PeopleShop, action palette, 236
PeopleShop, AI-controlled character, 332
PeopleShop, character, 236, 238, 331
PeopleShop, character actions, 236
PeopleShop, character behavior, 237, 331
PeopleShop, character emotion, 332
PeopleShop, character path, 236, 237
PeopleShop, civilian, 236
PeopleShop, control palette, 236
PeopleShop, decision bead, 237
PeopleShop, DI-Guy, 236
PeopleShop, display palette, 236
PeopleShop, Embedded Run-Time
Module, 238
PeopleShop, end bead, 237
PeopleShop, external process, 238
PeopleShop, interactivity, 237
PeopleShop, library of models, 236
PeopleShop, LOD, 236
PeopleShop, networking, 238
PeopleShop, script bead, 237
PeopleShop, sensor, 237
PeopleShop, sensor detection volume, 237,
238
PeopleShop, soldier, 236
PeopleShop, soldier actions, 236
PeopleShop, stealth display, 238
PeopleShop, vehicle, 236
perception, 316
perceptual-motor coordination, 274
Pere, 150
performance, 253, 256
performance degradation, 256, 258

Perl script, 237
Personal Haptic Interface Mechanism,
 104
personnel training, 350
Perspecta, 379–381
perspective, 169
perspective projection, 179
Peugeot, 362
PHANToM, 104–106, 184, 191, 193, 194,
 231–234, 262, 265, 292, 293, 299,
 329, 351, 352, 363
PHANToM, actuator, 105
PHANToM, backdrivability, 105
PHANToM, continuous force, 105
PHANToM, control bandwidth, 105
PHANToM, control electronics, 105, 232
PHANToM, controller, 143
PHANToM, degrees of freedom, 105
PHANToM, Desktop, 104, 105
PHANToM, handle, 265, 266
PHANToM, impedance control, 106
PHANToM, inertia, 105
PHANToM, mapped object, 234
PHANToM, maximum output force, 105
PHANToM, mechanical bandwidth, 105
PHANToM, model 1.5/6.0, 105, 106
PHANToM, price, 105
PHANToM, servo loop, 231
PHANToM, thimble, 292
PHANToM, workspace, 106, 234, 235
phantom ship, 326
PHIGS, 164
Phillips Petroleum, 374
phobia, 311
phobia-inducing stimulus, 311
Phong, 123
photo sensor, 35, 44
photodiode, 35
phototransistor, 49
physical constraint, 370
physical mockup, 357, 381
physical modeling, 13, 180, 205, 297
physical simulation model, 126
physical therapy, 306
physician, 294
physician, certification, 294
physician, experienced, 295
physician, novice, 295
physician, skill, 295
physician-in-training, 294
physics, 314
physics laboratory, 316

Phywe Systeme, 100, 101
piano, 310
Piantanida, 256
Picasso, 362
pick-and-place task, 311
Picture System 2, 7
PID, 101
pier-to-pier communication, 227
pilot, 4, 270, 273, 274, 276, 297, 328,
 338–340, 342
Pimentel, 18, 61, 121, 202, 203
pinch grasping, 95
pipeline architecture, 117
pipeline synchronization, graphics-haptics,
 144
pipeline throughput, 124
pirate ship, 326
Pirates of the Caribbean, 326
piston, 22
pitch, 17, 38, 53
pixel, 9, 11, 58, 60–65, 69, 70, 75, 76, 78,
 81, 83, 118, 120, 125, 131, 132, 136,
 138, 169, 170, 172, 310, 374, 376
pixel, cache, 130
pixel, color, 119, 120, 169
pixel, fill rate, 133
pixel, normal, 170, 190
pixel, shader, 133
pixelation effect, 11
placement task, 246, 259
planet, 324
plant, 317
plant design, 358, 359
plant simulation, 358
planting task, 317
plastic deformation, 180, 188
platoon, 328
platoon sergeant, 331
Play Station, 9
player, 326, 328
pneumothorax, 290
point of subjective equality, 266, 267
pointing, 253
pointing device, 381
polarized glasses, 83
polarizer, 66
political demonstration, 196, 197
polling algorithm, 49
polygon, 4, 10, 120, 121, 123, 124, 142,
 149, 158, 159, 161, 164, 168–170,
 184, 190, 197, 199–204, 211, 214,
 227–230, 256, 316, 321, 323, 379

polygon, aliased, 118
polygon, anti-aliased, 118
polygon, decimation, 163, 288, 352
polygon, edge, 118
polygon, Gouraud-shaded, 120, 126, 130,
 131, 137
polygon, light intensity, 166
polygon, non-textured, 130
polygon, normal, 123, 165, 166
polygon, shaded, 166
polygon, shading mode, 123, 165
polygon, split, 203
polygon, textured, 161, 169, 170, 358
polygon, type, 125
polygonal mesh, 165
Popescu, 117, 125, 126, 191, 306, 307
portability, 93
portal, 204
position measurement technology, 39
position measurement, noncontact, 24,
 32, 35
position, accuracy, 17
position, data, 35
position, measurement, 38, 135
position, measuring point, 35
position, sensor, 42, 109, 265
position, trackers, 16
position, tracking, 4
position-adapting backlighting, 71
position-feedback actuator, 106
posttraumatic stress disorder, 311
postural instability, 269
potentially visible set, 203, 204
potentiometer, 4, 23
potentiometer, accuracy, 23
power amplifier, 105
power grasp, 95, 176
power plant, 367
power putty, 306
power supply, 48
PowerGlove, 9
prebuilt world, 314
precision, 17
precision grasp, 95
predictive control, 34
predictor display, 368
PreOp, 295
PreOp bronchoscopy simulator, 294
presence, 250
Princeton University, 149
prism, 351
Pro/Designer, 134

Pro/Engineer, 214
probe, 45–47, 54
probe, accuracy, 46
probe, actual position, 46
probe, calibration, 46
probe, degrees of freedom, 45
probe, kinematic model, 45
probe, latency, 46
probe, measured position, 46
probe, tip, 45, 46
probe, workspace, 46
ProCDRS, 134
processing electronics, 29
processor, 10, 167
processor array, 81
product design, 250
product development cycle, 252
product redesign, 252
product usability, 252
programmable logic controller, 358
programmer, 365
programming environment, 221
programming language, 342
programming task, 211
programming time, 239
programming, task level, 235
projection distance, 74
projection matrix, 180
projection screen, 1
projective virtual reality, 370
projector, 76, 146, 148, 149, 329, 379
projector, array, 81, 82, 149
projector, Barco, 83
projector, calibration, 81
projector, digital, 77, 381
projector, Digital Micromirror Device
(DMD), 77, 78, 379
projector, DMD array element, 78
projector, DMD chip, 77
projector, luminance, 77
projector, micromirror, 77
projector, overlap zone, 81
projector, polarized, 83
projector, side by side, 81
projector, special optics, 83
projector, synchronized, 81
projector, visual signature, 81
proportional-integrative-derivative
controller, 101
proprioception, 95, 308, 310
proprioception cue, 43
proprioceptive adaptation, 276

proprioceptive control, 275
proprioceptive feedback, 265
proprioceptive mapping, 276
proprioceptive resolution, 95
proprioceptive sensing, 95, 275
proprioceptive sensorial channel, 271, 275
proprioceptive system, 270, 271, 275,
 276
proprioceptive system, calibration, 275
proprioceptor, sensing bandwidth, 97
proprioceptive mapping, left-right reversal,
 301
prostate, 291, 292
prostate, 3D model, 292
prostate, advanced malignancy, 291, 293
prostate, anatomical landmark, 292
prostate, cancer, 291
prostate, circular geometry, 293
prostate, enlarged, 291, 293
prostate, incipient malignancy, 291, 293
prostate, model, 292
prostate, normal, 293
prostate, palpation, 292
prostate, rubber model, 293
prostate, simplified model, 293
prostate, surface, 292
prostate, virtual, 293
prostate, wireframe, 293
prototype, 350, 355
prototype, assembly verification, 353
prototype, clay, 351
prototype, disassembly, 353
prototype, physical, 352, 354, 356
prototype, realization, 350
prototype, virtual, 350, 353
Provision 100, 9
proximity sensor, 194, 323, 324
pseudo-haptic piston, 265, 266
pseudo-haptic spring, 267
psychiatric disorders, 278
psychological deficit, 304
psychologist, 243
psychology, 277
pulley, 105
pulsing sound, 268
purchasing decision, 362
Push 1280, 22
pushbutton, 42–44, 45, 57, 65, 67, 102,
 161, 186, 187, 225, 252, 255, 315,
 316, 329, 364, 381
pushbutton, binary, 44
pushbutton, return spring, 186

pushbutton, travel limit, 186
Push display, 22
puzzle, 371

quadrangle, 124
Quadro2 Pro, 129, 133–135, 140, 144, 171
quantum physics, 316
Quantum3D, 76, 141
Queen’s University, 303

radar, 342
radar time-of-flight data, 163
radiosity, 134, 321, 322
radiosity, computation load, 167
radiosity, lighting, 321
radiosity, modeling, 167
radiosity, process, 168
radiosity, rendering, 315
raft, 326
rain, 338
Raindrop Geomagic, 163
RAM, 77, 197, 227
Ranadive, 256
range, 40, 41
rapid-fire cannon, 337
raster engine, 379
raster graphics, 376
rasterizer, 119
rasterizer chip, 129
ray casting, 253–255, 378
RB2 Model 2, 9
reach envelope, 356
ReachIn Technologies, 231
reaching motion, 309
reaction force, 98, 184
real environment, 276
real time, 2, 3, 12, 13, 70, 89, 116, 152,
 153, 218, 332, 367, 378
Real Time Graphics, 212
real world, 219
real-time computation, 116
real-time constraints, 125
real-time feedback, 110
real-time graphics, 127, 136, 373
real-time interaction, 116, 157
real-time motion, 309
real-time position, 17, 24, 25, 32, 42, 46
real-time rendering, 122, 129, 316, 321
real-time simulation, 160
real-time tag, 49
realistic simulation, 47
Realworld Imagery Inc., 169, 170

rearward vehicle, 332
rebirth of VR, 10
reciprocating multiplanar display, 379
reconfigurable virtual environment, 80
rectangle, 351
Reddy, 184, 185
redundant feedback, 265
Reed, 170
Rees, 337
reflector, 69
reflex behavior, 324
refresh rate, 117, 123, 146, 166, 169, 256
rehabilitation, 304, 309
rehabilitation, at home, 304
rehabilitation, clinic, 308
rehabilitation, cognitive, 304
rehabilitation, device, 306
rehabilitation, exercise library, 306
rehabilitation, functional, 306
rehabilitation, hand, 310
rehabilitation, handball exercise, 306
rehabilitation, neurological, 304
rehabilitation, orthopedic, 304, 305
rehabilitation, outpatient clinic, 304
rehabilitation, pegboard exercise, 306
rehabilitation, psychological, 311
rehabilitation, session, 309
rehabilitation, stroke, 308
rehabilitation, virtual, 304
rehabilitation, VR-augmented, 304, 308,
 311
rehabilitation, VR-based, 304, 305, 308,
 311, 312
Rehg, 364
Reichert, 329
relative coordinates, 41, 42, 44
relative motion, 256
relative position, 42, 45, 178
remote computer, 140
remote consultation, 287
remote environment, 1, 366, 367, 370
remote node, 149
remote player, 324
remote sensing, 225
remote simulation, 328
remote site, 367, 368, 371
remote workspace, 370
rendering architecture, 243
rendering chip, 131
rendering hardware, 324
rendering mode, 121, 201, 223, 229
rendering performance, 134

rendering pipeline, 117, 119, 140, 154, 171, 243
rendering pipeline, co-located, 143
rendering pipeline, performance, 118, 125, 172
rendering server, 146–148
rendering server, state, 148
rendering speed, 124, 182
rendering, parallel, 149
rendering, stereo, 129
rendering, wireframe mode, 135, 315
Research Center for Historical Monuments, 321
resistive force, 190, 192, 194
resistive losses, 23
resolution, 9, 11, 53
responsiveness factor, 74
Restle, 51
RGB, 64, 130, 131, 145
RGB input, 75
RGB monitor, 136
RGB output port, 128
RGB signal, 73
RGB subpixel arrangement, 70
RGB, subpixel, 70
Rich, 324
Richard, 257–259, 262
Rigamonti, 288, 289
right-eye view, 68
rigid body, 233
river, 326
Rizzo, 313, 314
road, 3, 332
Roadrunner ’98, 340–342
Robertson, 1
Robinett, 59, 60, 175, 176, 319
robot, 1, 232, 362–367, 370, 371
robot, Adept 1, 363
robot, arm, 363
robot, CAD design, 363
robot, cylindrical, 106
robot, firefighting, 367
robot, hidden, 370, 372
robot, human operator, 367
robot, industrial, 363–365, 371
robot, joints, 363
robot, manipulator, 362, 363, 365
robot, manual programming, 364
robot, master, 368–370
robot, master arm, 367
robot, Mitsubishi PA10, 364
robot, move-and-wait strategy, 368, 369

robot, multimodal teaching advisor, 364, 365
robot, off-line programming, 364, 366
robot, offline, 364
robot, operator, 368–370
robot, phantom, 368, 369
robot, programmer, 364
robot, programming, 364
robot, programming language, 364
robot, real, 364
robot, remote, 367, 368, 370, 371
robot, rover, 369
robot, sensors, 363–365
robot, service, 366
robot, slave, 367–371
robot, structured environment, 366
robot, task modeling errors, 365
robot, task-level programming, 364
robot, teach pendant, 364
robot, teaching time, 365
robot, teleoperation, 367, 368
robot, teleprogramming, 368, 369
robot, trajectory, 365, 368
robot, transparent teleoperation, 367
robot, unstructured environment, 366
robot, virtual, 364, 367, 368
robot, work envelope, 363
robotic arm, 6, 13, 367
robotic assembly line, 360
robotic assembly station, 353
robotic graphics, 363
robotic interface, 294, 295
robotic language, 364
robotic platform, 306
robotic programming, 364
robotics, 17, 287, 349, 363, 364
Robotics Laboratory of Paris, 370
Roehl, 210
role playing, 314
roll, 17, 38
Rolland, 59, 60
roller-coaster, 269
roof repair, 276
room, 17, 91, 202–204, 355
room temperature, 33
Rosen, 297
Rosenberg, 45, 97, 98, 102, 103, 189, 370
Rosenblum, 10
rotary potentiometer, 105
rotation, 17, 19, 117
rotation submatrix, 172, 174

Rousseau, 334
Roussos, 317, 318
rover, 368
Rover R75, 355
Royal Institute of Technology, 315
RS-485, 29
RS232, 32, 49
RS232 line, 22, 37, 39, 41, 44, 45, 52, 53,
 89, 100, 101, 107, 143, 301
RS232 serial port, 48
RU, 119, 120, 125
Rudrajit, 149
Ruffini corpuscle, 94, 95
run-time environment, 210
run-time software, 212
running shoe, 351
RUs, 119
Rutgers Ankle, 143, 144, 305, 306, 311
Rutgers Master I, 263, 264
Rutgers Master II, 228, 265, 306, 310
Rutgers Master II-ND, 308
Rutgers University, 292, 305, 306,
 308–310

SA receptor, 94
safety switch, 110
Saint Fruition, 352
Saint Sophia Cathedral, 321
Salcudean, 190
Salisbury, 104, 194
sampling interval, 250
San Diego, 8
Sandia National Laboratories, 290
Sanstrom, 374, 375
Satava, 287, 295, 297, 299
satellite servicing task, 367
Sato, 295
scaling, 117, 234, 381
scaling, factor, 174, 364
scaling, non-uniform, 180
scaling, undesirable effect, 235
scan conversion, 376
scan frequency, 65
scanner, 319, 321
scanner control, 36
scanner rotational speed, 36
scenario control interface, 337, 338
scenario visualization, 237
scene, 4, 18, 20, 58, 124, 157, 173, 178,
 202, 213, 227–229, 238, 254, 256,
 262, 267, 271, 275, 290, 299, 311,
 321, 338, 360, 361, 376, 379, 381

scene graph, 212, 213, 215, 216, 223, 224, 233
scene graph, change, 213, 226, 234
scene graph, collision detection, 214
scene graph, external node, 212
scene graph, haptics, 231, 232
scene graph, hierarchical culling, 214
scene graph, internal node, 212, 213, 232
scene graph, leaf node, 213, 222, 224
scene graph, node, 215
scene graph, node attributes, 213
scene graph, node bounding volume, 213, 226, 233
scene graph, parent node, 215
scene graph, root node, 212, 213, 215
scene graph, sibling node, 215
scene graph, sub-tree bounding volume, 214
scene graph, subtree, 214
scene graph, traversal, 214, 215, 231, 232, 234
scene graph, tree structure, 212
scene graph, update, 213, 214
scene graph, viewpoint node, 215
scene rendering, 33
scene traversal, 138
scene, 2D, 117
scene, 3D, 70, 78, 180, 379
scene, coherence, 167, 182, 214
scene, colliding structures, 126
scene, complex, 256, 268
scene, complexity, 120, 121, 124, 139, 200, 211, 227–229, 299, 316, 376
scene, concentric zone, 198
scene, constant refresh rate, 199
scene, content, 268
scene, database, 138, 198
scene, detail, 166
scene, discontinuity, 76
scene, floating, 271
scene, frozen, 197, 229
scene, generator, 4
scene, geometry, 119, 124, 146, 167
scene, globally illuminated, 167
scene, hierarchy, 232
scene, illumination, 164
scene, imagery, 77
scene, interactivity, 168
scene, level of detail, 125
scene, low resolution, 254
scene, monoscopic, 257
scene, multiple views, 215

scene, patch, 167
scene, perspective control, 74
scene, photo-realistic, 166, 355
scene, polygonal count, 170, 228, 229
scene, projection, 117
scene, realism, 125, 164, 169, 332
scene, realistically-looking, 164, 165
scene, refresh, 377
scene, refresh rate, 120, 121, 139, 256
scene, remote, 367, 370
scene, rendering cost, 201
scene, shaded, 118
scene, static, 74
scene, stereo, 78, 83, 120, 257, 258
scene, stereo refresh rate, 121
scene, view, 176, 177, 351
scene, viewpoint, 167
scene, virtual, 60, 65, 149, 309
scene, visible object, 215
scene, volumetric, 376, 381
scene, zooming, 66
Scheffel, 129–131, 133, 134
Schlaroff, 183
Schmult, 1
Schöffel, 167, 168
school desk, 313
school environment, 314
Schott, 63
Schraft, 358, 359
Schroeder, 279, 280
Schwartz, 287
scientific community, 1
scientific visualization, 12, 199
scissors, 290, 301
Scott, 119
SCP, 234
screen, 3
screen mapping, 179, 180
screen refresh rate, 44, 131
screen resolution, 131
script, 331
SCSI interface, 379
sculpting, 351
sculpting tool, 351
sculpture, 319, 351
seam welding, 364
secondary cache, 136
Sega, 324
segmentation algorithm, 295
seismic data, 374–376
seismic data visualization, 374, 378
seismic information, 374

seismic subsurface information, 374
seismic subsurface model, 375
Seismitarium, 374, 375
selection task, 246
selection time, 256
self-motion, 271, 276
Semi-Automated Forces (SAFOR),
 332–334
semiconductor, 100
SensAble Devices, 187, 193
SensAble Technologies, 105, 212, 231,
 232, 234, 235, 351, 352
Sense8 Co., 10, 118, 133, 212, 214, 216,
 219–221
sensing element, 41
sensing equipment, 358
sensing glove, 1, 7, 8, 16, 24, 46–49, 53,
 54, 57, 74, 117, 129, 157, 175, 176,
 178, 212–214, 228, 238, 257, 262,
 268, 290, 316, 340, 353, 354, 364,
 370, 373, 381
sensing glove, 5DT Data Glove, 46, 49–52
sensing glove, calibration, 48, 50, 52, 53
sensing glove, CyberGlove, 46, 52, 53, 99,
 100, 107, 310
sensing glove, Didjiglove, 46, 51, 52
sensing glove, electronics box, 48
sensing glove, latency, 52
sensing glove, no calibration, 49
sensing glove, optical sensor, 25
sensing glove, performance comparison, 53
sensing glove, Pinch Glove, 46, 48, 49,
 252, 265
sensing glove, PowerGlove, 8, 9
sensing glove, readings, 51
sensing glove, sampling rate, 46, 53
sensing glove, sensors, 46, 53
sensing glove, tethered, 46
sensing glove, tracker, 46
sensing glove, VPL DataGlove, 8, 9, 25,
 49, 195, 263, 265, 275, 299, 316
sensing glove, wireless, 46, 50
sensing glove, work envelope, 47
sensing suit, 16, 196, 235
sensor, 20, 21, 23, 36–38, 49, 53, 66, 93,
 117, 248, 306, 332, 358, 363
sensor, 3D magnetic, 25
sensor, analog signal, 23
sensor, bending, 51
sensor, coupling, 44
sensor, data, 20, 225
sensor, electrode, 51

sensor, IR, 39
sensor, latency, 117
sensor, noise, 20, 53, 247
sensor, optical, 38
sensor, position, 104, 105
sensor, proprioceptive, 94
sensor, range, 25
sensor, rapidly adapting, 94
sensor, rate of discharge, 94
sensor, receptive field, 94
sensor, relative, 42
sensor, resolution, 46, 66
sensor, slow adapting, 94
sensor, spatial resolution, 94
sensor, tactile, 94
sensor, thermal, 94
sensor, triangular, 37
sensor, uncertainty, 364
sensor-intensive task, 365
sensor-processing electronics, 29
Sensorama Simulator, 3, 5
sensorial channel, 3, 16, 57, 58, 265, 269
sensorial channel requirement, 19
sensorial conflict, 265–267, 269–272,
 274
sensorial conflict, magnitude, 273
sensorial data, 195, 270
sensorial domain, 262
sensorial feedback modality, 57
sensorial illusion, 266, 269
sensorial interaction, 324
sensorial modalities, 3
sensorial overload, 262, 324, 334
sensorial redundancy, 262, 263, 265
sensorial substitution, 262–264
sensorial transposition, 262, 263
sensorial transposition, complex, 262
sensorial transposition, force–(force,
 sound, image), 262
sensorial transposition, force–(sound,
 image), 262
sensorial transposition, force–image, 262
sensorial transposition, force–sound, 262
sensorial transposition, force-force image,
 262
sensorial transposition, simple, 262
sensorial transposition, sound–(force,
 image), 262
sensorial transposition, sound–image, 262
sensorial transposition, sound–sound force,
 262
sensorized exoskeleton, 23

sensorized glove, 310
sensorized joint, 22
sensory disarrangement, 275
sensory rearrangement, 274
sensory–motor control loop, 93
Seow, 94
Séquin, 200–202
serial feedback arm, 105
serial kinematic structure, 22
serial line, 21, 66, 81
serial port, 128, 129
SERIAL1, 219, 220
server, 151–153, 220, 311
servo loop, 106, 232–234
sex, 279
sex industry, 277, 279
SGI, 10, 135–138, 214, 292, 316
SGI 4D/310 VGX, 9
SGI 4D/320 VGX, 316
SGI computer, 80
SGI graphics supercomputer, 11
SGI Infinite Reality, 326, 358
SGI Onyx2 IR, 135
SGI Reality Engine, 10, 137, 200, 326
SGI workstation, 11, 288, 336
shading, 165
shading, flat, 123, 166, 211
shading, force, 126
shading, Gouraud, 123, 124, 165, 168
shading, mode, 123, 125
shading, Phong, 123, 166, 168
shading, rasterizer, 171
shading, smooth, 160
shading, wireframe, 123
shadow, 306
shared virtual environment, 321, 373
shared virtual world, 314
Sheingold, 33
Sheridan, 1, 256
Shewchuk, 350
Shimoga, 94, 97
Shimojo, 262
ship, 251, 326, 338
shoe, 358
Short Brothers, 357
shoulder, 23, 95
shoulder flexion, 309
shoulder proprioceptive sensor, 95
shutter glasses, 59, 72, 75
Siever, 237
SIGGRAPH 2000, 351
sight, 57

sighting display, 337
sign language, 262
signal-to-noise ratio, 25
Sillion, 166, 167
SIMI, 35
SIMI Motion Capture 3D, 35
simulated casualty, 290
simulated environment, 44
simulated world, 44, 59
simulation, 2–4, 6–8, 11, 18–20, 42, 45,
 54, 58, 75, 194, 234
simulation computational load, 149
simulation cycle, 42, 182
simulation effect, 267
simulation effect, direct, 267
simulation effect, indirect, 267
simulation efficiency, 13
simulation environment, 194
simulation gain, 44
simulation interactivity, 373
simulation kernel, 211
simulation latency, 117, 221
simulation loop, 74, 219
Simulation Network (SIMNET), 329, 332,
 333
simulation quality, 117, 143, 197
simulation realism, 18, 49, 57, 65, 84, 92,
 100, 170, 195, 205, 299, 313, 326,
 328, 338
simulation response, 54, 228
simulation session, 50
simulation sickness, 20, 66, 117, 141, 142,
 243–245, 265, 269, 317, 324
simulation sickness, disorientation, 272
simulation sickness, headaches, 20
simulation sickness, nausea, 20, 254, 272
simulation sickness, oculomotor distortion,
 272
simulation sickness, questionnaire, 272
simulation sickness, vertigo, 20
simulation, cockpit view, 76
simulation, coherence, 238
simulation, dynamic response, 21
simulation, fast tilting maneuver, 76
simulation, feedback, 57
simulation, frame rate constancy, 197
simulation, goodness, 244
simulation, interactivity, 74, 229
simulation, latency, 33
simulation, level of autonomy, 195
simulation, performance, 244
simulation, reset, 44

simulation, running, 23
simulation, site, 93
simulation, speed, 197
simulation, survivability, 149
simulation, system, 117, 125, 252
simulation, task, 149, 220, 248, 262
simulation, view, 22, 74, 117, 157, 178, 220
simulation, walking, 23
simulator, 6, 7
Singhal, 151, 152
single logical screen, 146
single-constraint-at-a-time algorithm, 38
single-instruction, multiple-data
 parallelism (SIMD), 138
single-soldier simulator, 328
single-user system, 176
Sitterson Hall, 204
situation awareness, 334, 342
skeletal articulation, 95
skeletal structure, 288
skeletal system, 84, 377
skin, 94, 101, 297, 298, 352
skin, contact temperature, 95
skin, dermis, 94
skin, disease, 268
skin, electric pulses, 97
skin, epidermis, 94
skin, mechanoreceptor, 94, 100
skin, mechanoreceptor spatial resolution,
 94
skin, nociceptor, 94, 95
skin, receptor density, 94
skin, spatial resolution, 94
skin, stretch, 94, 297
skin, tactile receptor, 97
skin, temperature sensing, 94
skin, temporal resolution, 94
skin, thermoreceptor, 94
skin, thermoreceptor spatial resolution, 94
skin, vibrations, 94, 100
skull, 288
slave, 29, 142
sliding door, 194
small-batch production, 350
smell, 3, 4
smoke, 367
snake, 311
social interaction, 278, 279
sociological study, 244
sociology, 277
soda can, 184
Soda Hall, 200, 204

software, 6–10, 13
software driver, 51
software filter, 44
software gain, 44
software optimization techniques, 122
software tool, 210
soldier, 151, 237, 324, 328–330
solenoid valve, 360
Soler, 166, 167
Solid System Modeler, 316
Song, 184, 185
SonicFury 3D, 92
Sony, 7, 9, 11
sound, 57, 90, 91, 97, 268
sound, 3D, 84–86, 89–92
sound, 3D cue, 91
sound, 3D source, 89
sound, 5.1 surround format, 90, 91
sound, binaural, 84
sound, compound, 89
sound, cone of confusion, 87
sound, convolved, 88, 89
sound, cross-talk cancellation, 91
sound, cross-talk effect, 91
sound, direct, 84, 87, 88
sound, direct/reflected ratio, 88
sound, direct/reverberated ratio, 88
sound, display, 84, 226
sound, Dolby Digital, 91
sound, effect, 133
sound, faint, 88
sound, far away source, 88
sound, feedback, 84, 219, 231, 243, 260,
 367
sound, frequency, 86, 87, 263
sound, front-back confusion, 87
sound, hardware, 89
sound, head-shadow effect, 86
sound, high frequency, 86
sound, high-energy source, 88
sound, input, 89
sound, intensity, 86
sound, interaural intensity difference (IID),
 86, 87
sound, interaural time difference (ITD), 86,
 87
sound, interference, 87
sound, loudness, 88
sound, low frequency, 86
sound, maximum ITD, 86
sound, monoaural, 84
sound, motion parallax, 88

sound, occlusion, 89
sound, path variation, 87
sound, pinna-reflected, 87
sound, pinnal transform, 88
sound, propagation velocity, 86
sound, psychoacoustic information, 84
sound, quad format, 90
sound, range cue, 88
sound, real source, 89
sound, recorded, 84
sound, reflected, 84, 88
sound, reflection path, 87
sound, reverberated, 86, 88
sound, simulated source, 89
sound, source, 18, 85, 87–89, 268
sound, source azimuth, 85, 86, 88
sound, source azimuth localization, 86, 91
sound, source elevation, 85, 87, 88
sound, source image, 89
sound, source in far field, 88
sound, source localization, 84
sound, source location, 85, 86, 88
sound, source location cue, 87, 88
sound, source position, 84, 85
sound, source prior knowledge, 88
sound, source range, 85, 88
sound, source spatial recognition rate, 88
sound, spatially localized source, 88
sound, speed, 86
sound, stereo, 3, 61, 84, 85, 90, 91
sound, synthesized, 89
sound, temporal cue, 86
sound, transients, 86
sound, virtual, 84
sound, wave, 86
source, azimuth, 88
source, elevation, 88
source, frequency, 88
source, in far field, 88
Sowizral, 34, 212, 223–226
space aliasing effect, 377
Space Ball, 219
space exploration, 366
space repair task, 368
space teleoperation, 368
Sparcstation II, 136
spatial view, 18
SPEA Fire, 126
speaker, 33, 84, 90, 91
speaker, phantom, 91
speaker, virtual, 88
specular reflection, 123, 166

specular reflection coefficient, 123
speed of sound, 33
sphere, 199, 200, 227, 233, 292
spherical coordinates, 85, 381
spider, 311
spine, 29
spline, 158, 159, 168, 184, 236, 299
spline, NURBS, 163, 164
spline, waypoint, 236
spring deformation law, 44
spring-damper, 234
springlike force, 102
squad, 331
squadron, 338, 339
Srinivasan, 95, 97, 190
SS. Sergius and Bachhus, 321, 322
stainless steel, 31, 32
stairway, 197
Stampe, 211
Standard Performance Evaluation Co.
(SPEC), 134
standing, 236
Stanford Medical School, 306
Stanford University, 139, 308, 370
Stanney, 243, 244, 269, 271, 272, 276
Stansfield, 290, 291
StarCraft, 324, 325
state change, 332
state graph, 311
state machine, 312
State University of New York (SUNY) at
 Stony Brook, 295
static art, 319
static friction, 186, 265
statue, 319, 320, 351, 352
Stealth Vehicle, 333
steel, 100
steering wheel, 326, 357
stepper motor, 70
stereo, 7
stereo display, 10, 22
Stereo Display Manager, 316
stereo glasses, 33, 74, 75, 214
stereo glasses, 3D glasses, 75
stereo glasses, CrystalEyes3, 75
stereo glasses, passive, 83
stereo glasses, transmittance rate, 74, 75
stereo headphones, 84
stereo image, 9
stereo viewing hardware, 59
StereoGraphics, 75
stereophonic sound, 4

stereoscopic graphics, 66, 67, 258, 259
Stewart platform, 306
stiffness, 266
Stinger, 329
Stinger Trainer, 329
Stinger, mock-up, 329
Stockburger, 247
Stoffregen, 270, 276
Stone, 301, 302, 329, 330, 337
Stonehenge, 321
stopwatch, 247
store interior, 362
strain gauge, 53
stress, 312
stroke, 304, 308, 309, 313
stroke survivor, 308
Strommer, 364, 366
structural information server, 289
student, college level, 315
student, high school level, 316
Sturman, 176
stylus, 39, 105, 161, 191, 351, 381
SU-27, 342
subject, 248, 252, 253, 255–258, 260, 262,
 263, 265, 266, 272, 275, 290, 293,
 295, 302–304, 312–315, 317, 336,
 357, 360–362, 370
subject, actions, 246, 247
subject, adapted, 275, 276, 293
subject, ADHD, 314
subject, age, 253
subject, average contact force, 248
subject, baseline, 246
subject, body, 250
subject, brainwave, 250
subject, breathing rate, 250
subject, cognitive demand, 250, 254
subject, cumulative contact force, 249
subject, error averaging, 248
subject, experienced, 336
subject, expert, 247, 295, 299, 302
subject, force tracking ability, 97
subject, force variance, 249
subject, heart rate, 250
subject, motor movement, 314
subject, muscle exertion, 250
subject, novice, 247, 295, 299, 302
subject, peak contact force, 249
subject, performance, 248, 253, 254, 256,
 258, 262, 264–266, 302, 334
subject, performance measure, 247–250
subject, physical demand, 250

subject, preference, 247
subject, prior experience, 247, 336
subject, prior knowledge, 249, 253
subject, prior learning, 248
subject, prior mental image, 248, 336
subject, prior VR experience, 262
subject, reaction time, 247
subject, recognition rate, 88
subject, right hand-dominant, 252
subject, skill, 302
subject, targeting, 275
subject, task completion time, 247, 248,
 252, 256, 258, 260–262, 299–301,
 303, 369
subject, task error rate, 247, 248, 252
subject, task learning, 248, 249, 292, 293,
 299, 300, 336
subject, task learning time, 247, 248, 258,
 259
subject, work load, 250
subjective evaluation questionnaire, 313
submarine, 334, 336
submarine, virtual, 336
subpixel, 78
subwoofer, 90
successiveness threshold, 94
surgery, open, 301
sun, 317
Sun, 369
Sun Blade 1000, 136, 137
Sun Blade 1000, fill rate, 136
Sun Blade 1000, system architecture, 137
Sun Microsystems, 135–137, 214, 221
super-scalar architecture, 135
Superscape PLC, 10
supervisory control, 370
support troops, 332
surface, 158
surface, bumpy, 158
surface, color, 117, 158, 169
surface, compliance, 125, 353
surface, compression, 185
surface, control point, 160, 184
surface, control point lattice, 184
surface, curvature, 191
surface, cutting, 184
surface, deformation, 159, 183–187, 191,
 192, 263–265
surface, degree of deformation, 126
surface, detail, 170
surface, displacement map, 193
surface, free-form, 353

surface, friction, 194, 234
surface, frictionless, 190, 192
surface, geometry, 215, 379
surface, gradient, 193
surface, graphics, 376–379
surface, height, 170
surface, highly curved, 158
surface, illumination, 158
surface, lighting information, 199
surface, local deformation, 192
surface, material properties, 118
surface, normal, 166, 190
surface, parametric, 158–160, 164, 168,
 184
surface, patch, 160
surface, plasticity, 13
surface, polygonal, 159, 190, 198
surface, polygonal mesh, 184, 185
surface, reflection coefficient, 123
surface, ridges, 353
surface, rigid, 189
surface, roughness, 193
surface, single-point contact, 184
surface, smooth, 190
surface, smoothness, 126, 159, 353
surface, static, 160
surface, sticky, 193
surface, temperature, 126
surface, texture, 158, 164, 215
surface, topological change, 184
surface, velvet-like, 192
surface, weighted normal, 192
surgeon, 297–299, 301
surgeon, experienced, 303
surgeon, hand, 301
surgeon, learning curve, 297
surgeon, MIS skill, 301
surgeon, novice, 297, 303
surgeon, trainee, 303
surgeon, vascular, 299
surgery, 287, 294, 296, 297, 300
surgery, abdominal, 299
surgery, endoscopic, 301
surgery, eye, 297
surgery, hand, 305, 308
surgery, laparoscopic, 301
surgery, minimally-invasive (MIS), 297,
 299–301
surgery, MIS trainer, 301
surgery, open, 297, 299, 300
surgical area, 299
surgical dexterity, 301

surgical instrument, 299–301
surgical planning, 297
surgical procedure, 299
surgical simulator, 197, 297, 299
surgical skill, 287, 299, 303
sustained force, 106
Sutherland, 4, 5, 21
Swan, 252
Swartout, 331
sweating, 269
Sweden, 153, 301, 315
swept volume, 353, 354
switch box, 34
Switzerland, 278, 321
synthetic environment, 3
synthetic scene, 58, 65
synthetic world, 2, 18, 58
syringe, 290, 297, 299
system bus interface, 130
system clock, 247
system lag, 271
system latency, 20, 21, 117, 128, 141, 228,
 229, 243, 247, 253, 270, 271, 274, 301
system latency, constant, 274
system of coordinates, 17, 20, 23, 41, 172,
 173, 175, 176, 178, 179
system of coordinates, Cartesian, 173, 178
system of coordinates, object, 172, 173
system of coordinates, orthonormal, 172
system responsiveness, 252, 256, 258–260,
 266
system responsiveness, variability, 258
system stability, 243

tactile cue, 43
tactile feedback, 8, 97–100, 265, 340
tactile feedback device, 97
tactile feedback interface, 102
tactile feedback pattern, 100
tactile feedback system, 98
tactile receptor, 94
tactile sensing, 95
Taffinder, 302
Tan, 95
tangential force, 299
tank, 251, 328, 332, 334
tank, ammunition consumption, 333
tank, battle, 328, 333
tank, battle damage, 333
tank, commander, 332
tank, crew, 332
tank, M1, 332

tank, simulator, 329, 332
tank, track, 332
tank, virtual, 332
target, 253, 254, 256
target acquisition, 301, 329
target area, 256, 259
target diathermy, 301
target engagement, 338
target force, 95
target spotting, 329
target viewpoint, 338
target volume, 258–260
task, 107, 261–263, 265, 364
Task Analysis Toolkit, 357, 358
task, accuracy requirement, 246
task, difficulty, 248
task, index of difficulty, 259, 260
task, intermediate steps, 261
task, learning, 285
task, performance, 260, 265, 280, 293
task, sensor-intensive, 364
task, taxonomy, 253
task, throughput, 259, 260
taste, 3
Taxén, 315
Taylor, 100, 287
TCP/IP, 147, 151–153, 221, 353
teacher, remote, 315
teaching by imitation, 308, 309
team training, 334, 340
Technology Education Center, 360
Tehrani, 52
Teixeira, 18, 61, 121, 202, 203
Tektronix, 64
tele-rehabilitation, multiplexed, 304, 309,
 311
tele-rehabilitation, orthopedic, 306, 307
tele-rehabilitation, stroke, 311
telemanipulation application, 104
teleoperation, 256, 363, 366, 367, 369, 370
teleoperation, intuitiveness, 370
teleoperation, multiplexed, 371, 372
teleoperation, one-to-many, 371
teleoperation, supervisory, 370, 371
telepresence, 1
Telepresence Research Inc., 7
telerehabilitation, 304, 309, 310
telerobotics, 1
telescopic piston, 22
temperature differential, 101
temperature feedback, 97, 100
temperature feedback glove, 101

temperature sensor, 194
Temple University, 319
tendonitis, 268
tennis racket, 193
TeraRecon Inc., 376, 378, 379
Terracotta Warriors, 321
terrain model, 329
terran troops, 324
tether, 268, 361, 362
Texas Instruments, 77
text-to-speech conversion, 332
text-to-speech voice generation, 332
texture, 13, 164, 169, 223, 321
texture, force, 193
texture, 3D spatial cues, 169
texture, accelerator, 119
texture, array, 169
texture, bitmap, 169
texture, blending cascade, 170, 171
texture, building, 169
texture, bump map, 170, 171, 193
texture, chip, 119, 120
texture, coordinates, 169
texture, engine, 130
texture, force, 193
texture, frequency component, 192
texture, grayscale, 170
texture, ImageCEL, 169
texture, JPEG, 215
texture, landscape, 169
texture, library, 169
texture, light map, 170–172, 321, 322
texture, manipulation, 215
texture, mapping, 169
texture, monochrome, 170
texture, multi-texturing, 170, 171, 192, 321
texture, normal map, 171
texture, people, 169
texture, perspective corrected, 119
texture, RGB, 215
texture, scaling, 169
texture, scanned photograph, 169
texture, texel, 130, 131, 169, 170
texture, texel color, 169
texture, texel normal, 171
texture, TGA, 215
texture, tree, 169
texture, vehicle, 169
textured graphics, 360
texturing, 123, 124, 169, 236, 322
texturing, blending cascade, 170
texturing, corresponder function, 168

texturing, operation, 133
Thalmann, 194–197
The World of Virtual Reality, 1, 2
theme park, 324, 326
therapist, 304, 306–308, 311
thermal actuator, 100
thermal conductor, 100
thermocouple, 101, 194
thermode, 100, 101
thermode control diagram, 101
thermoelectric heat pump, 100
thermoresistor, 194
thin-film transistor (TFT), 69, 70, 75
thorax, 288
thumb, 8, 46, 49, 51, 97, 216, 297, 298, 306
thumb, anteposition, 53
tile overlap computations, 149
tiled display, 139, 340
tilt sensor, 49, 50
time delay, 20, 27, 32, 66, 252, 256, 276,
 279, 368–370
time dilation, 316
time of flight, 33
time-of-arrival detector, 39
timing circuit, 48
tissue burning, 301
tissue damage, 299, 300
TMS320 DSP processor, 379
TNO Physics and Electronics Laboratory,
 329
Todorov, 308, 309
toe, 29, 95
Tomb Raider, 324, 325
toolkit, 10, 52, 133, 157, 160, 161, 206,
 210–212, 214, 227, 239, 365, 381
toolkit, 3D Studio, 160, 164, 214, 215, 321
toolkit, 3D Studio Max, 52, 133, 134
toolkit, Cyberspace Developer Kit (CDK),
 211
toolkit, editor, 160
toolkit, form-Z, 160
toolkit, General Haptics Open Software
 Toolkit (GHOST), 212, 231, 239, 351
toolkit, general-purpose, 211
toolkit, Java 3D, 212, 221, 222, 229, 231,
 232, 235, 239
toolkit, Java 3D 1.3 Beta 1, 227, 228, 229,
 230
toolkit, Lightwave, 222
toolkit, low end, 211
toolkit, Maya, 52
toolkit, Multigen, 214

toolkit, NuGraph, 161
toolkit, PeopleShop, 212, 235, 236, 238,
239, 331
toolkit, proprietary, 212
toolkit, public domain, 212
toolkit, RealTexture, 169
toolkit, Rend386, 211
toolkit, Softimage, 52
toolkit, special-purpose, 212, 239
toolkit, Starbase, 164, 184
toolkit, User Interface Toolkit, 231
toolkit, VCToolkit (VCT), 211
toolkit, Virtual Reality Toolkit (VRT3), 211
toolkit, VirtualHand, 231
toolkit, VR4, 365
toolkit, VRML, 212, 215, 222, 224, 232,
235, 239, 321, 322
toolkit, Wavefront, 214, 222
toolkit, WorldToolKit (WTK), 10, 13, 212,
214, 216, 221, 222, 224, 225, 227,
229, 231, 232, 235, 239, 306
toolkit, Wrap, 163
toolkit, WTK Release 9, 227–230
toolkit, WTK/Java3D performance
comparison, 227
torque, 44, 45, 293, 306, 357
torque target, 97
torso, 29, 178
torso tracking, 253
total latency, 117
touch, 3, 5, 57, 92
touch feedback, 93, 97, 126, 353, 361
touch receptor, 93
Tower of Hanoi, 260–262
toy, 351
trachea, 293
trackball, 1, 44–46, 47, 54, 57, 67, 74, 117,
177, 219, 370, 373
trackball, Logitech Magellan, 44
trackball, support, 44
trackball, work envelope, 47
tracked object, 20–22
tracker, 7, 13, 17, 19–22, 24, 26, 32, 35, 36,
38, 39, 41–43, 54, 57, 63, 65–67, 74,
78, 89, 90, 100, 109, 117, 128, 129,
150, 157, 178, 212–214, 219, 225,
226, 228, 238, 247, 252, 257, 261,
270, 290, 313, 329, 340, 364
tracker performance parameters, 18
tracker, 3D Bird, 39
tracker, 3D non-contact, 24
tracker, AC, 31

tracker, AC magnetic, 24, 26, 29
tracker, AC magnetic field, 27, 31
tracker, accuracy, 19, 20, 22, 26, 27, 29, 31,
 32, 37–39, 41, 142, 160, 162
tracker, active area, 74
tracker, ambient interference, 31
tracker, analog signal, 25
tracker, application, 18
tracker, Ascension, 361
tracker, Ascension Extended Range
 Controller, 28, 29
tracker, Ascension Extended Range
 Transmitter, 29
tracker, backpack, 29
tracker, backpack electronics, 29
tracker, backpack unit, 29
tracker, base station, 29
tracker, beacon array, 37, 38
tracker, calibration, 32, 33, 39
tracker, ceiling interface board, 38
tracker, ceiling-HiBall interface, 38
tracker, control box, 34
tracker, control unit, 33
tracker, controller, 74
tracker, daisy-chained, 38
tracker, data, 18, 20, 84
tracker, data format, 32
tracker, data packet, 21, 24
tracker, data set, 21
tracker, DC, 31, 32
tracker, DC magnetic, 24, 27–29, 142
tracker, DC magnetic field, 27, 31
tracker, DC magnetic pulse, 27
tracker, DC magnetic receiver, 29
tracker, direct line of sight, 34
tracker, distance from transmitter, 30
tracker, drift, 19, 20, 39, 41
tracker, dynamic response, 31
tracker, echo, 33
tracker, electronic interface, 21
tracker, electronic unit, 24, 25, 32
tracker, electronics, 20, 21
tracker, emitter, 27, 74, 160
tracker, ergonomic concerns, 38
tracker, ergonomic drawbacks, 24
tracker, error, 20, 270
tracker, Extended Range Controller, 29
tracker, Extended Range Transmitter
 (ERT), 29
tracker, external synchronization signal,
 142
tracker, Fast Bird Bus, 29

tracker, Fastrack receiver, 162
tracker, Flock of Birds, 27–29, 31, 32, 42
tracker, four receivers, 129
tracker, functionality, 42
tracker, HiBall 3000, 37, 38, 41
tracker, hybrid, 20, 37, 39, 40
tracker, inaccuracy, 20
tracker, InertiaCube, 39, 40
tracker, inertial, 38, 39, 67
tracker, information, 18
tracker, inside-looking-out, 35–37
tracker, interference, 31, 34, 35
tracker, interference from magnetic fields,
 22
tracker, interference from metallic
 structures, 22
tracker, InterSense, 78
tracker, InterSense 300, 67
tracker, InterSense IS-900, 39–41, 337
tracker, InterTrax2, 39, 42
tracker, jitter, 19, 20, 22, 25, 39, 66, 74
tracker, joint sensors, 22
tracker, laser scanner, 36, 37
tracker, laserBIRD, 36, 37
tracker, latency, 19, 20, 22, 25, 31, 32, 35,
 37–41, 66, 67, 270, 271
tracker, lateral shift, 275
tracker, line of sight, 35
tracker, line-of-sight constraint, 39
tracker, line-of-sight requirement, 74
tracker, Logitech, 33
tracker, magnetic, 22, 24, 27, 31–34, 37,
 66, 107, 338, 362
tracker, magnetic field, 26
tracker, magnetic interference, 31, 66
tracker, magnetic pulse amplitude, 27
tracker, magnetometer, 24
tracker, mains interference, 31
tracker, mains synchronization, 31
tracker, measurement capability, 19
tracker, measurements, 19
tracker, measuring point, 21
tracker, mechanical, 22–24, 29, 66
tracker, MotionStar wireless suit, 29, 30
tracker, multiplexed, 33, 34
tracker, multiplexing effect, 21, 25
tracker, noise, 43, 67
tracker, object, 21
tracker, operating frequency, 31
tracker, operating range, 20, 25, 29, 32, 33
tracker, optical, 22, 35–38
tracker, orientation only, 270

tracker, orthogonal coils, 24
tracker, output, 20
tracker, outside-looking-in, 35
tracker, performance comparison, 41, 42
tracker, performance parameters, 19
tracker, Polhemus, 7, 162–164, 219, 220,
 228, 306, 309
tracker, Polhemus Fastrack, 25, 26, 31,
 32, 37, 43
tracker, Polhemus Isotrack, 25
tracker, Polhemus Long Ranger, 25–27
tracker, Polhemus stylus, 160, 162
tracker, Polhemus wooden tripod, 25, 26
tracker, poorly calibrated, 270
tracker, position, 175
tracker, position measurement error, 30
tracker, processing electronics, 29
tracker, range, 34, 37, 39, 41, 108
tracker, readings, 43
tracker, receiver, 18, 24–27, 29, 31–34, 41,
 42, 62, 74, 128, 160, 175, 176, 275
tracker, receiver coil, 25
tracker, receiver data, 29
tracker, receiver multiplexer, 27
tracker, receiver position, 24, 27
tracker, receiver voltage, 24
tracker, refresh rate, 37
tracker, repeatability, 20
tracker, resolution, 20, 32, 89
tracker, sampling rate, 25, 31
tracker, sensitivity, 35
tracker, sensitivity degradation, 35
tracker, signal, 34
tracker, sonic disk, 39
tracker, sonic disk IR code, 39
tracker, source, 128, 162, 175, 176
tracker, source-less operation, 39
tracker, streaming mode, 20
tracker, technology, 18, 22
tracker, third party, 67, 70
tracker, transmitter, 24–27, 29, 31–34
tracker, transmitter coil, 25, 27
tracker, transmitter field, 24
tracker, transmitter magnetic field, 26, 30
tracker, transmitter signal attenuation, 34
tracker, transmitter system of coordinates,
 29, 41
tracker, triangular frame, 33
tracker, triangulation, 33, 35, 38
tracker, ultrasonic, 35, 74
tracker, ultrasonic rangefinder module, 39
tracker, ultrasonic-inertial, 39

tracker, ultrasound, 32–34, 66, 74
tracker, update rate, 21, 22, 29, 32, 33, 35,
 38, 39, 41, 66
tracker, vicinity, 20
tracker, virtual, 225
tracker, weight, 24, 78
tracker, work envelope, 20, 22, 24, 31, 35,
 74
tracking, 16
tracking algorithm, 40
tracking envelope, 361
tracking station, 39, 41
tracking suit, 23, 24
tracking suit, Gypsy, 24
tracking suit, Gypsy 2.5, 24
tracking suit, mechanical, 23
tracking suit, wireless, 357
tracking unit, 9
tracking volume, 362
tracking, lateral shift, 275
tracking, vision-based, 360, 362
tractor cabin, 356, 357
trainee, 290, 292–295, 297–299, 301, 302,
 328, 329, 331, 332, 342
trainee, actions, 294, 299, 331
trainee, dexterity, 295
trainee, head motion, 337
trainee, hit/miss ratio, 329
trainee, learning, 295
trainee, maintenance worker, 360
trainee, motion, 329
trainee, officer, 334
trainee, performance, 295, 298
trainee, pilot, 342
trainee, progress, 338
trainee, skill level, 295
trainee, view, 336
training, 197, 290, 291, 360
training applications, 286
training cost, 332
training effectiveness evaluation study, 336
training manual, 331
training scenario, 336, 338
training simulation, 186, 239, 276
training system, 360
training, emmergency response, 290
training, on animals, 294
transformation matrix, 176
translating rod, 43
translation, 17, 19, 117
transmission delay, 117
trap door, 326

traumatic brain injury, 304, 313
tree, 253
tree graph, 177, 178, 204
tree graph, branch, 177
tree graph, hand, 178
tree graph, hierarchical structure, 177, 178
tree graph, node, 177, 178
Treffitz, 26
Tremblay, 99
tremor, 20
triangle, 198–200, 222, 288
triangle, array, 222
triangle, Gouraud-shaded, 126, 132
triangle, mesh, 125, 158, 159, 163, 199,
 233, 319
triangle, smooth-shaded, 146
triangle, textured, 137
triangular sensing element, 36
triangulation, 162
triangulation-based scanning, 162
Trimension Systems Ltd., 78–81
truck, 251
true color, 120
TruSurround, 92
turbulence, 338
Turkey, 321, 322
Turner, 108
turnkey VR system, 9
turntable, 381
TV, 1, 7, 35, 60, 62, 370
TV transmitter, 62
tweezer, 299
two-hand manipulation, 303
two-tap filter, 31

U.S. Air Force Modeling and Simulation
 Office, 340
U.S. Army, 335
U.S. Food and Drug Administration, 268
U.S. Occupational Safety and Health
 Administration, 268
ubiquitous computing, 17
UDP, 153, 221
“ultimate display”, 5
Ultra Port Architecture (UPA), 136
ultrasonic microphone, 74
ultrasonic range data, 40
ultrasonic range tracking, 41
ultrasonic sensor, 9
ultrasonic signal, 32
ultrasonic speaker, 33, 39
ultrasound, 4, 24, 66, 376

ultrasound source, 34
UltraSpark III processor, 136
UMA, 133
UNESCO, 321
unicast communication, 151, 153
unicast packet, 151, 153
Uniformed Services University of the
 Health Sciences, 288
unit cube, 180
unit step function, 187
United Kingdom, 3, 153, 301, 337, 349,
 359
United States, 8, 10, 308, 332, 349, 359
universal serial bus (USB), 98, 102
universal VR task, 253, 270
universe, 202, 210, 212, 215, 218, 219,
 223, 224
universe, viewpoint, 220
University of Birmingham, 260
University of California at Berkeley, 200
University of California at San Diego, 288
University of Central Florida, 276
University of Colorado, 287
University of Geneva, 321
University of Houston, 316
University of Illinois at Chicago, 78
University of Leeds, 355
University of Medicine and Dentistry of
 New Jersey, 292, 309
University of North Carolina at Chapel
 Hill, 5, 38, 183, 202, 204
University of Pennsylvania, 357
University of Southern California, 313, 331
University of Strathclyde, 355
University of Washington, 289, 353
University of Waterloo, 211
Unix operating system, 136, 139, 214
unknown environment, 365
update rate, 17
URM, 39
urology resident, 293
Ursino, 297
usability engineering, 250
usability evaluation, methodology, 251
usability problem, 251
usability study, 244, 250
usability study, expert guidelines-based
 evaluation, 250, 251
usability study, formative usability
 evaluation, 250, 252
usability study, summative evaluation, 250,
 252, 253

usability study, user task analysis, 250
USB port, 128, 133
user, 3, 5, 8, 11, 16, 18, 20, 22, 24, 29, 35,
43–45, 47, 48, 50, 52, 57–60, 62, 63,
65–70, 72, 74–76, 78, 81, 84, 86–91,
93, 94, 97, 99, 102, 107, 108, 110,
117, 118, 128, 129, 140, 149–151,
153, 160, 167, 172, 175, 178, 184,
186, 190, 192–195, 210, 213, 215,
220, 226, 233, 234, 238, 243, 251,
260, 268, 269, 271, 274, 276, 279,
289, 316, 323, 353, 357, 367, 373,
376, 378, 381
user’s 3D input space, 381
user’s action, 19, 22, 89, 97, 102, 116, 149,
151, 167, 168, 187, 194, 195, 225,
250, 251, 276
user’s arm reach, 43
user’s body, 29
user’s characteristics, 157, 226, 243, 253
user’s command, 16
user’s contact motion, 93
user’s cuff, 52
user’s eye, 60, 65
user’s face, 88
user’s feeling of immersion, 18, 229, 234
user’s finger, 44, 46, 49, 51
user’s fingertips, 66
user’s forearm, 102
user’s gesture, 196
user’s hand, 17, 42, 44, 46, 93, 100, 105,
381
user’s hand configuration, 100
user’s hand gesture, 107
user’s hand movement, 252
user’s head, 3, 17, 18, 21, 33, 34, 42, 59,
64, 68, 70, 78, 84, 87, 89, 91, 270
user’s head motion, 7, 18, 66, 74, 84, 270
user’s head position, 66, 70
user’s head posture, 18
user’s HMD, 20
user’s immersion, 57, 84, 250
user’s information flow, 250
user’s input, 2, 13, 54, 57, 116, 117, 194,
213, 218, 228, 229, 256, 290, 370,
373
user’s instantaneous position, 70
user’s interactivity, 84
user’s learning, 51
user’s limb, 17, 268
user’s motion, 102
user’s needs, 243

user's perception, 84
user's performance, 243, 244, 260, 276
user's position, 70
user's response, 244
user's surroundings, 84, 88
user's view, 4
user's viewing angle, 74
user's viewing direction, 18, 65, 74
user's voice input, 89
user's work envelope, 50, 75
user's wrist, 50, 105, 109
user's wrist motion, 46
user's *X*, *Y*, *Z* position, 70
user, absence state, 268
user, adapted, 274, 276
user, age, 95
user, arm pain, 68
user, auditory system, 267
user, behavior, 202
user, body, 29, 33, 35
user, cognitive load, 256
user, collaborative work, 70
user, comfort, 13, 93
user, compensatory response, 274
user, disoriented, 44, 269
user, expert, 293, 295, 308
user, fatigue, 24, 45, 61, 63, 67, 95, 102,
 108, 247, 250
user, feeling of immersion, 59
user, field of view (FOV), 75, 84
user, forehead, 63
user, freedom of motion, 24, 62, 66, 68, 72,
 75, 100, 102, 108, 109
user, gender, 95, 97
user, gesture library, 47
user, hand size, 53
user, head motion, 61, 66
user, head orientation, 39
user, hearing impaired, 262
user, horizontal field of view angle, 74
user, HRTF, 88, 89
user, I/O, 136
user, immersive sensation, 74
user, in sweet spot zone, 91
user, instructor, 290
user, interface, 3
user, interface technique, 381
user, migraine prone, 268
user, motion volume, 34
user, musculo-skeletal system, 267
user, natural interaction, 72, 102
user, nauseated, 271

user, navigation, 167, 203, 219, 224, 226, 251, 268, 316
user, novice, 293
user, pain, 95
user, pointing gesture, 176
user, position change, 74
user, primary, 78
user, remote, 149, 153, 154
user, safety, 13, 93, 244, 267, 276
user, secondary, 78
user, sense of immersion, 24
user, several simultaneous, 149
user, shadow, 78
user, sitting, 68
user, skill, 95
user, skin, 267
user, surrounded by virtual world, 74
user, thermal comfort zone, 100, 101
user, tired, 93
user, torso geometry, 88
user, tracked, 29, 226
user, visually impaired, 262
user-computer interface, 3
user-to-screen distance, 74
Utah Teapot, 166

V-Dome, 81
V-Dome, projection area, 83
v. d. Weyden, 129, 130
vacuum, 317
valve handle, 367
van der Linde, 106
variable diffuser, 69
variable-distance focusing, 66
VE engine, PC-based, 127
VE testbed, 246
vector, 122, 123, 126, 143, 171, 172, 174, 262
vector, interpolation, 165, 166
vehicle, 236–238, 270
vein, 297, 298
vein wall, 297
Velcro, 107
Venus de Milo, 161, 162
verbal command, 2
vertex, 148, 161, 170, 174, 183, 184
vertex, color, 167, 168
vertex, connectivity, 160, 164, 184
vertex, coordinates, 138, 160, 164, 172, 175, 180
vertex, deformation propagation law, 184
vertex, illumination, 165

vertex, information, 118
vertex, lighting, 172
vertex, manipulation, 184
vertex, normal, 123, 165, 166, 190
vertex, position, 174
vertex, shader, 133
vertex, twin, 184, 185
vertical–polar coordinate system, 85, 86
vertices, 160
vertices, shared, 158
vertices, wireframe, 161
vestibular sensor, 269, 270
vestibular stimulation, 270
vestibular system, 270
vestibular–ocular reflex, 271
VESUB simulator, 334, 336
VGA, 11, 254
Viatronix Inc., 295
vibration, 126, 193, 311
vibration, amplitude, 99, 194
vibration, frequency, 99, 194
vibrotactile actuator, 99, 100
vibrotactile feedback, 99
video arcade, 3
video chip, 120
video feedback, 3, 75, 128, 256, 367, 368
video game, 3, 60, 278, 304, 309, 310, 322, 324
video game player, 92
video image, 353, 362, 367, 370
video projector, 329
video sequence, 35
video signal, 18
video timing, 120
video tracking system, 364
video-game graphics, 133
Videologic, 92
VIEW, 8, 10
view control, 17
view direction, 367.
view plane, 378
view trajectory, 296
view transformation, 178
view, occluded, 316
viewer, 81, 83, 122, 199
viewing angle, 180, 199, 379
viewing board, 351
viewing cone, 234
viewing direction, 18
viewing plane, 180
viewing vector, 66

viewpoint, 200, 202–204, 219, 246, 252, 254, 272, 326, 355
Viewpoint Inc., 121, 164, 170, 177, 195, 198
Viirre, 267, 268
violence, 324
violin, 84
Virgin Mary, 319
virtual abdomen, 299
virtual airplane, 268
virtual arm, 308
virtual assembly, 354
Virtual Assembly Design Environment (VADE), 353, 354
virtual ball, 84, 175, 176, 184, 191, 192, 213, 256–258, 260, 263–265, 306
virtual battlefield, 332, 333
Virtual Binoculars SX, 65, 66
virtual biopsy, 296
virtual body, 18
virtual building, 78, 151, 197
virtual camera, 42, 44, 172, 178, 179, 196, 198–201, 204, 205, 224, 234, 235, 238, 254, 296
virtual camera, view frustum, 214
virtual casino, 324
virtual casualty, 291
virtual cathedral, 322
virtual classroom, 313
virtual cockpit, 339, 340
virtual colonoscopy, 378
virtual DigiKey, 306
virtual dinosaur, 121
virtual dissection room, 288
virtual environment, 24, 46, 47, 153, 195, 197, 263, 269, 367, 371
Virtual Environment Configurable Training Aids (VECTA), 340
virtual environment, authoring, 210
virtual environment, degree of autonomy, 194
virtual environment, distributed, 149, 150, 246
virtual environment, multi-user, 152
virtual environment, networked, 149, 153, 154
virtual environment, shared, 149, 150, 151, 154, 315
virtual environment, view, 41
virtual fixture, 370
virtual football player, 177, 178, 195
virtual garden, 317, 318

virtual hand, 49, 157, 175, 176, 178, 184, 186, 191, 195, 213, 215–217, 222, 224, 227, 228, 253, 255, 257, 275, 306, 310, 316, 371, 381
virtual hand, hierarchical structure, 178, 179
virtual hand, tremor, 25
virtual handshake, 195
Virtual Heritage Network, 321
Virtual Holographic Workstation, 136
virtual human, 194
Virtual Interface Environment Workstation, 8
Virtual Jungle Cruise, 326
virtual laboratory, 316, 317
virtual laparoscopic interface, 301, 302
virtual lecture hall, 315
“virtual” man, 279
virtual map, 251
virtual maze, 272
virtual mine, 329–331
virtual mistress, 279
virtual model, 43, 355, 369
virtual museum, 319
Virtual Notre Dame, 321
virtual object, 5, 18, 20, 43, 45, 58, 117, 123, 151, 152, 158, 160, 164, 165, 178, 183, 194, 205, 210, 218, 220, 224, 247, 248, 256, 260, 265, 365
virtual object, deformable, 262
virtual object, fragile, 248, 249
virtual object, geometry, 214
virtual object, shape, 158
virtual object, tactile identification, 92
virtual obstacle, 364
virtual office, 194
Virtual Physics Laboratory, 316, 317
Virtual Presence Ltd., 301
virtual prism, 173, 174
virtual programming interface, 7
virtual projectile, 326
virtual prototyping, 197, 350, 351, 356, 358
virtual prototyping, assembly verification, 351
virtual prototyping, CAD, 351
virtual prototyping, ergonomic study, 351
virtual prototyping, free form design, 351
virtual rabbit, 198, 199
virtual rabbit, surface detail, 199
virtual ray, 42
virtual reality, 1–4, 7, 8, 10, 16
virtual reality application, 17

virtual reality application developer, 13
virtual reality industry, 12
virtual reality simulation, 1, 84
virtual reality system, 7
virtual reality transfer protocol (VRTP),
 227
virtual rehabilitation, 287
virtual rehabilitation room, 306
Virtual Research Systems, 67, 68
virtual restoration, 319
virtual room, 84
virtual scene, 9, 18, 20, 46, 74, 81, 122,
 160, 175, 182, 211, 316, 326, 329,
 331, 335, 336, 357, 360, 367, 370,
 371
virtual sex, 279
virtual shopping tour, 362
virtual showroom, 362, 363
virtual sound, 16
virtual spring, 188, 265–267
Virtual Stinger Trainer, 329, 330
virtual surface, 93
virtual switch, 340
virtual teacher, 313
Virtual Technologies, 107–109, 231
virtual tour, 321
virtual tour guide, 322, 323
Virtual Visual Environment Display
 (VIVED), 7, 8
virtual work envelope, 363
virtual world, 2, 5, 22, 46, 58, 70, 74, 84,
 116, 117, 120, 134, 151–153, 157,
 160, 172–177, 179, 180, 197, 202,
 210, 212, 214, 215, 223, 224, 226,
 227, 243, 250, 253, 256, 279, 299,
 316
virtual world, cluttered, 182
virtual world, distributed, 149
virtual world, networked, 153
virtual world, shared, 149
virtual world, state, 149, 213
virtual world, state coherence, 151
virtual world, state update, 238
VirtualANTHROPOS, 356, 357
Virtuoso Developer, 319
viscosity, 193
Visible Human, 287, 288
Visible Human, elbow, 288
Visible Human, online gallery, 288
Visible Human, skin, 288
Visible Man, 288, 289, 294
vision, 58, 265

visual acuity, 18
visual artifact, 140
visual capture, 271
visual channel, 117
visual cue, 265, 306
visual data, 367
visual distraction, 313
visual feedback, 16, 84, 92, 140, 172, 238,
 262, 265, 266, 270, 271, 274, 276,
 310, 326, 351, 353, 367, 371, 378
visual feedback, corrupted, 92
visual feedback, distorted, 275
visual flow, 268
visual inspection, 370
visual occlusion, 22, 367, 381
visual sensorial channel, 265, 271
visual shift, 274
visual stability, 271
visual stimulation, 270
visual system, 267, 270, 276
visual target, 275
visual–vestibular coupling, 270
visual-vestibular coupling, altered, 270
visual-vestibular coupling, distinct from
 real world, 270
visual-vestibular coupling, real-world
 equivalent, 270
visualization, 315, 357, 359, 360, 371, 373,
 374, 376
visualization application, 134
visualization loop, 373
Visualization of the Impact of Tolerance
 Allocation (VITAL), 355
visualization system, 373, 376
visualization technique, 376
visualization time, 373
Visualize fx, 119
visuovestibular sensorial conflict, 271
Vlaar, 190
vocal chords, 293
voice command, 331, 336, 364
voice input, 290, 361
voice recognition, 252, 290
volcano, 326
voltage, 24, 25, 27
voltage, analog, 27
voltage, control, 27
voltage, induced, 27
volume buffer, 376, 379
VolumePro 1000 Development Kit, 379
volumetric dataset, 287
volumetric display, 378–382

volumetric graphics, 376–379
volumetric hardware acceleration, 376
volumetric interaction, 381
volumetric refresh rate, 379
volumetric rendering, 296, 376
volumetric rendering board, 378
volumetric scene, 381
volumetric simulators, 378
volumetric visualization, 376
vomiting, 269
Voronoi volume, 183
voxel, 376–379
VPL Inc., 8–10, 61
VR, 17, 18, 35, 36, 39, 41, 54, 57, 61, 74,
 135, 136, 139, 244
VR application, 10, 21, 57, 74, 134, 178,
 182, 231, 238, 253, 285, 286
VR application, developer, 10
VR application, emerging, 349
VR application, multi-monitor, 140
VR application, sectors, 286
VR application, traditional, 349
VR arcade, 326
VR architecture, distributed, 141
VR companies, 10
VR debriefing, 333
VR defense application, 328
VR engine, 61, 110, 116, 117, 127, 128,
 140, 154, 157, 170, 182, 194, 197,
 231, 365, 374
VR engine, architecture, 117
VR engine, computation load, 197
VR engine, definition, 116
VR engine, distributed, 140
VR engine, PC based, 117, 128, 129
VR engine, system time, 194
VR engine, workstation based, 117
VR exposure, 246, 268, 269, 275, 276,
 280
VR exposure, aftereffects, 269, 275
VR exposure, duration, 268, 269
VR exposure, incremental, 274
VR exposure, intensity, 269
VR exposure, repeated, 274, 275
VR exposure, vestibular aftereffects, 276
VR exposure, visual aftereffects, 276
VR hardware, 4, 247
VR I/O interface, 11
VR industry, 12
VR instruction, 212
VR interaction, 17, 52, 61, 269, 280
VR interactivity, 117

VR interface, 16, 53, 250, 268
VR interface hardware, 17
VR manufacturer, 11
VR market, 10–12
VR market survey, 285, 349
VR model, 367
VR modeling, 157
VR modeling cycle., 158
VR modeling task, 133
VR object, 42, 44
VR output interface, 110
VR overdose, 276
VR pioneering companies, 10
VR product, 8, 10
VR programming, 214, 239, 243
VR research, 10
VR sensorial modality, 18
VR simulation, 13, 21, 25, 42, 46, 57, 76,
89, 100, 110, 116, 117, 126, 127, 147,
148, 157, 160, 166, 167, 169, 183,
228, 243, 248, 262, 269, 294, 298,
317, 324, 356–358, 360, 362, 364,
370, 371
VR simulation, manual control, 176
VR system, 35, 57, 58, 61, 116, 243, 246,
250, 268, 276, 277
VR system, components, 13
VR system, PC based, 127
VR system, turnkey, 78
VR task, 248, 256
VR Teaching Laboratory, 128, 129, 212
VR technology, 243, 247, 269, 278, 287
VR testbed, 253
VR therapy, 309, 311
VR trainer, 276
VR training, 295
VR world, 21
VR, accuracy requirements, 17
VR, applications, 13
VR, business potential, 349
VR, devices, 1
VR, effect on public life, 279
VR, effects on society, 243, 244
VR, end user, 286
VR, functionality, 2
VR, impact on private life, 278
VR, impact on professional life, 278,
280
VR, impact on public life, 278–280
VR, networking, 278
VR, societal impact, 244, 277
VR-assisted rehabilitation, 106

VR-based education, 316
VR-based training, 353, 360
VRML, scene graph, 227
VRT3, 10
Vuurweek, 139

walking, 196, 236, 304
walkthrough, 151, 167, 204
walkthrough simulation, 202
wall, 88, 89, 105, 106, 170, 171, 184,
 188–190, 203, 252, 256, 262, 295,
 306, 321
wall display, 374
wall, sound reverberation, 89
wall, stiffness, 188
wall-mounted display, 340
wall-size image, 11
wand, 42, 43, 252, 317, 318
Wann, 276
war on terrorism, 328
warehouse, 358
warm virtual world, 100
Watchman, 7
water effects, 326
water spray, 326
Watson, 256, 258–260
Wax, 295, 296
weapon, 329, 332
wearable PC, 362
weather effects, 333
Web application, 10
Web browser, 315
Web page, 3D-enabled, 227
weight, 9, 11, 265
weight, illusion, 265, 266
Welch, 35, 37, 274, 276
welding path, 364
well management, 374
Wenzel, 7, 86–88
Werkhoven, 275
Weyden, 133, 134
Wheatstone bridge, 53
white water rafting, 326
Whyte, 359
wide-area network (WAN), 150, 152, 153,
 246, 332, 340
Wiker, 95
Wildcat II, 135, 170
Wildcat II 5110, 144–146
Wildcat II 5110, parscale architecture,
 145
Williams, 149, 350

wind, 3, 326, 328, 336
window, 272, 313
window resolution, 125
window thermometer, 194
Windows, 214
Windows 2000, 140
Windows NT, 227
wire assembly board, 362
wire bundle, 362
wireframe rendering, 8
WireGL, 147–149
wireless communication, 50
wireless interface, 361
wireless modem, 29
wireless receiver, 50
wireless transmission, 362
wiring diagram, 362
Wolfe, 376
wood, 100
Wood, 249
work envelope, 357
workbench, 212, 252, 351
workbench, angle, 78
workbench, Baron, 78
workbench, diffuser screen, 78
workbench, dual-projector configuration,
 79
workbench, enclosure, 78
workbench, horizontal, 78
workbench, IR controller, 78
workbench, L-shaped, 78
workbench, mirror, 78
workbench, second projector, 78
workbench, single-projector configuration,
 79
workbench, sound system, 78
workbench, stereo collapse effect, 78
workbench, stereo viewing cone, 78
workbench, stereo viewing volume, 78
workbench, surface tilting, 78
workbench, tilted, 78
workbench, tilting mechanism, 78
workbench, V-Desk 6, 78
workbench, visual artifacts, 78
workbench, wooden enclosure, 78
worker, 361, 362
worker fatigue, 357
worker training, 360
workload, 334
workstation, 3, 9, 117, 121, 135, 136, 139,
 151, 290, 292, 316, 329, 332–334,
 336, 358, 367, 373, 379

workstation, multi-pipeline, 146
world exploration, 314
world state, 116, 117
world system of coordinates, 39, 173–175,
 178, 182
world system of coordinates, origin, 174
wrist, 109, 265
wrist motion, 42
wrist orientation, 49
wrist position, 9
WTK, action function, 218, 219
WTK, classes, 214
WTK, content node, 215
WTK, display windows, 215
WTK, event scheduling, 218, 219
WTK, external sensor, 219, 220
WTK, file importing, 215
WTK, functions, 214, 218, 220
WTK, geometry primitive, 214
WTK, macro, 219
WTK, material table, 215
WTK, message passing, 221
WTK, movable node, 215–217
WTK, networking, 220
WTK, object instances, 232
WTK, object-oriented naming convention,
 214
WTK, performance, 228
WTK, rendering speed, 228
WTK, run-time loop, 220
WTK, scene graph, 215, 216
WTK, sensor, 218–220
WTK, separator node, 215
WTK, server manager, 220
WTK, simulation loop, 218, 219
WTK, simulation manager, 219
WTK, simulation server, 220, 221
WTK, system latency, 228, 229
WTK, texture importing, 215
WTK, transform node, 215
WTK, Universe class, 214
WTK, World2World, 220, 221

xBox, 129, 131, 132
xBox, audio processor, 133
xbox, block diagram, 132
xBox, CPU, 132
xBox, fill rate, 132
xBox, front-side bus, 132
xBox, GPU, 131, 132
xBox, hard disk, 133
xBox, network interface, 133

xBox, unified memory architecture, 133

XGA resolution, 11

Xia, 199, 200

Yanagihara, 364, 365

yaw, 17, 38, 39, 53

Yim, 99

Younse, 77

Z-buffer, 10, 119, 180

Z-buffering, 135

Z-buffering process, 180

Zachmann, 361

Zerkus, 100, 101

Zimmerman, 7

zooming in–out, 43, 337

Zyda, 151, 152

A groundbreaking Virtual Reality textbook is now even better

Virtual reality is a very powerful and compelling computer application by which humans interact with computer-generated environments in a way that mimics real life and engages various senses. Although its most widely known application is in the entertainment industry, the real promise of virtual reality lies in such fields as medicine, engineering, oil exploration, and the military, to name just a few. Through virtual reality, scientists can triple the rate of oil discovery, pilots can dogfight numerically superior "bandits," and surgeons can improve their skills on virtual (rather than real) patients.

This Second Edition of the first comprehensive technical book on virtual reality provides updated and expanded coverage of the technology such as:

- Input and output interfaces including touch and force feedback
- Computing architecture (with emphasis on the rendering pipeline and task distribution)
- Object modeling (including physical and behavioral aspects)
- Programming for virtual reality (WorldToolKit, Java 3D, GHOST, and PeopleShop)
- An in-depth look at human factors issues, user performance, and sensorial conflict aspects of VR
- Traditional and emerging VR applications

The new edition of *Virtual Reality Technology* is specifically designed for use as a textbook. Thus, it includes definitions, review questions, and a CD-ROM with video clips that reinforce the topics covered. The CD-ROM also contains a *Laboratory Manual* with homework and programming assignments in VRML and Java 3D, as follows:

- Introduction to VRML and Java 3D
- Sensor and Event Processing
- VRML and JavaScript
- Scene Hierarchy, Geometry, and Texture
- VRML PROTO and Glove Devices
- Viewpoint Control, Sound, and Haptic Effects

The Second Edition will serve as a state-of-the-art resource for both undergraduate and graduate students in engineering, computer science, and other disciplines.

GRIGORE C. BURDEA is a professor at Rutgers, the State University of New Jersey, and author of the book *Force and Touch Feedback for Virtual Reality*, also published by Wiley. **PHILIPPE COIFFET** is a Director of Research at CNRS (French National Scientific Research Center) and Member of the National Academy of Technologies of France. He has authored twenty books on robotics and virtual reality, which have been translated into several languages.

Cover Design and Illustration: Michael Rukowski



ISBN 0-471-36089-9



9 780471 360896