Praise for *3D User Interfaces, Second Edition*

"An essential guide for anyone developing interfaces for Virtual and Augmented Reality gaming experiences."

—Richard Marks, Director of Magic Lab, Sony PlayStation

"An incredible resource for 3D interaction researchers and practitioners, made all the more timely and valuable with today's renewed interest in Virtual and Augmented reality platforms. Everyone in VR and AR can benefit from the decades of research thoughtfully organized and presented in this updated edition."

—Andy Wilson, Microsoft Research

"This is an essential book for researchers and developers creating 3D user interfaces. If you're developing Virtual Reality or Augmented Reality experiences, or even mobile and desktop 3D applications, you need to buy this book."

—Mark Billinghurst, University of South Australia

3D User Interfaces
Second Edition

Addison-Wesley Usability and HCI Series

Addison-Wesley

Visit **informit.com/series/usability** for a complete list of available publications.

Essential Guides for Human-Computer Interaction and User Interface Designers

Books in the HCI and Usability series provide practicing programmers with unique, high-quality references and tutorials on interaction and interface design, a critical component of success for any mobile app or website. The books in this series bring the full range of methods and options available to meet the challenge of designing for a natural and intuitive global user experience.

Make sure to connect with us!
informit.com/socialconnect

Pearson | informIT.com the trusted technology learning source | Addison-Wesley | O'REILLY Safari

3D User Interfaces

Theory and Practice
Second Edition

Joseph J. LaViola Jr.

Ernst Kruijff

Ryan P. McMahan

Doug A. Bowman

Ivan Poupyrev

Addison-Wesley

Boston • Columbus • Indianapolis • New York • San Francisco
Amsterdam • Cape Town • Dubai • London • Madrid • Milan
Munich • Paris • Montreal • Toronto • Delhi • Mexico City
São Paulo • Sidney • Hong Kong • Seoul • Singapore • Taipei • Tokyo

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearsoned.com.

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2016961395

ISBN-13: 978-0-13-403432-4

ISBN-10: 0-13-403432-5

1 17

Editor-in-Chief
Mark Taub

Senior Acquisitions Editor
Laura Lewin

Development Editor
Susan Zahn

Managing Editor
Sandra Schroeder

Project Editor
Lori Lyons

Production Manager
Dhayanidhi

Copy Editor
Warren Hapke

Indexer
Erika Millen

Proofreader
Sam Sunder Singh

Technical Reviewer
Andy Wilson

Publishing Coordinator
Olivia Basegio

Cover Designer
Chuti Prasersith

Compositor
codeMantra

To my family, with love—they are my life.

—Joe

To my wife and kids, for their love and support.

—Ernst

To my parents, who always encouraged me.

—Ryan

To Dawn and all the kids under the Bowman bigtop,
with all my love.

—Doug

To my parents, for bringing me up, and to
my wife forputting up with me.

—Ivan

Contents at a Glance

Contents

FOREWORD TO THE SECOND EDITION

It seems like yesterday that the first edition of this book was published. However, yesterday was over a dozen years ago—just in time for me to adopt it for the 3D user interface course I have been teaching ever since. The first edition was the ambitious and successful first text devoted to this subject. A timely and welcome gift to our field, it was the product of a team of authors actively engaged in leading-edge research, who codified and successively refined their work through the development of a series of tutorials out of which their book grew.

As Jim Foley wrote in his foreword to the first edition, "Three-dimensional user interfaces are finally receiving their due!" But, exhilarating though the times were then, the intervening years have brought us even more exciting advances. As I write this, virtual reality is becoming consumer reality, with tethered head-worn displays connected to desktop computers and mobile head-worn displays powered by phones—all superior in many ways to the orders-of-magnitude more expensive systems that were found in research labs back when the first edition appeared. Meanwhile, hand-held augmented reality games have become international sensations, downloaded hundreds of millions of times, and early commercial head-worn augmented reality displays are already in the hands and on the heads of eager developers.

Simply put, the processing and display technology needed for interactive 3D is now commonplace. The graphics power of current smartphones is comparable to that of full-size game consoles that debuted after the first edition was published. Hardware and software for accurately tracking the 3D position and orientation of a smartphone or a user's head and for capturing full hand and body pose are now integrated into an increasing number of those phones. Even smartwatches can run 3D applications.

But amidst this thrilling democratization of 3D, there is a problem. Powerful processors, high-quality displays, responsive interaction devices, and efficient software by themselves are not enough to make 3D useful. These technologies need to be combined together in the right way to create effective 3D user interfaces. However, the vast majority of user interface designers and developers are only knowledgeable about and comfortable with 2D user interfaces. What can be done about this?

That's where this book comes in. Joe LaViola, Ernst Kruijff, Doug Bowman, and Ivan Poupyrev—the authors of the first edition—have joined with Ryan McMahan. Together, they have given us this extensively updated second edition, which provides the thorough background needed to understand current 3D user interfaces and the software and hardware from which they are built. Like its predecessor, the second edition teaches the reader how to design, implement, and evaluate 3D user interfaces, summarizing and categorizing decades of ongoing research and practice. Two extended application case studies are threaded throughout the book, offering substantial examples of how to make appropriate choices from among the possibilities described, whether high-level design, displays, input devices, interaction techniques, or evaluation approaches.

Many of the technologies underlying 3D user interfaces are different from those used in 2D user interfaces, have not undergone the same level of standardization, and are more actively evolving. Yet there is one constant: people. Recognizing this, the authors dedicate several substantial chapters to the perceptual, cognitive, and physical bases, as well as to the principles of human–computer interaction for how we interact in and with the 3D world. By grounding the book in a principled understanding of users, they help guide us toward the design of effective 3D user interfaces that honor our abilities and respect our limitations.

I'm looking forward to using the second edition in my course at Columbia. And whether you're a seasoned researcher or practitioner, or a fresh convert to the power of 3D interaction, you're also in for a treat!

Steve Feiner
Department of Computer Science
Columbia University
January 2017

FOREWORD TO THE FIRST EDITION

Three-dimensional user interfaces are finally receiving their due! Research in 3D interaction and 3D display began in the 1960s, pioneered by researchers like Ivan Sutherland, Bob Sproull, Fred Brooks, Andrew Ortony, and Richard Feldman. Although many commercially successful 3D applications exist—computer-aided design and simulation, radiation therapy, drug discovery, surgical simulation, scientific and information visualization, entertainment—no author or group of authors has written a comprehensive and authoritative text on the subject, despite a continuing and rich set of research findings, prototype systems, and products.

Why is that? Why is it that this book by Doug Bowman, Ernst Kruijff, Joe LaViola, and Ivan Poupyrev is the first thorough treatment of 3D UIs?

Perhaps it was our digression during the last 20 years to the WIMP GUI. After all, the Windows, Icons, Menus, and Pointers GUI is used very widely by millions of users. Mac OS and Microsoft Windows users know it well, as do many UNIX users. Indeed, every user of the Web works with a GUI, and this year there are many hundreds of millions of them. Two-dimensional GUIs will be with us for a long time. After all, a lot of the workaday world with which we deal is flat—not just our Web pages but our documents, presentations, and spreadsheets too. Yes, some of these can be extended to 3D, but most of the time, 2D is just fine, thank you very much. Furthermore, pointing and selecting and typing are relatively fast and relatively error-free—they work, and they work well.

Perhaps it is that not as many people use 3D GUIs as use the 2D WIMP GUI, and so they are not thought to be as important. But the above list of 3D applications involves multibillion-dollar manufacturing industries, such as aerospace and automotive, and equally large and even more important activities in the life-saving and life-giving pharmaceutical and health care industries.

Perhaps it was that we needed the particular set of backgrounds that Doug, Joe, Ivan, and Ernst bring to the table. Doug comes out of the GVU Center at Georgia Tech, where he worked on 3D UIs with Larry Hodges and others and learned the value of careful user studies and experimentation, and he is now a member of an influential HCI group at Virginia Tech; Joe works at Brown with Andy van Dam, a long-time proponent of rich 3D interaction; Ivan comes from the HIT Lab at the University of Washington, where he worked with Tom Furness and Suzanne Weghorst, and now works with Jun Rekimoto at Sony CSL; and Ernst works with Martin Goebel in the VE Group at Fraunhofer IMK in Germany.

Whatever the case, I am excited and pleased that this team has given us the benefit of their research and experience. As I reviewed the draft manuscript for this book, I jotted down some of the thoughts that came to my mind: comprehensive, encyclopedic, authoritative, taxonomic; grounded in the psychological, HCI, human factors, and computer graphics literature; grounded in the personal research experiences of the authors, their teachers, and their students.

I myself have long preached the importance of integrating the study of the computer with the study of the human. Indeed, this is the key premise on which I built the GVU Center at Georgia Tech. This book certainly follows that admonition. There are numerous discussions of human issues as they relate to 3D navigation and interaction, drawing on references in psychology and human factors.

This is indeed a book for both practitioners and researchers. The extensive literature reviews, examples, and guidelines help us understand what to do now. Combined with the research agenda in Chapter 13, The Future of 3D User Interfaces (now Chapter 12), the material also helps us have a sense of what it is that we do not yet know.

I particularly commend to readers the Chapter 11 discussion of evaluating 3D UIs. We in the computer graphics community have tended to design devices and techniques and then "throw them over the wall" to the user community. This is not the route to success. Careful study of user needs coupled with evaluation as part of the ongoing design cycle is much more likely to lead to effective techniques. The authors, all of whom have grappled with the difficult task of designing 3D interfaces, know from first-hand experience how crucial this is. Their section 11.4, on the distinctive characteristics of the 3D interface evaluation process, is a wonderful codification of that first-hand knowledge.

Thanks to Doug and Ernst and Joe and Ivan!

Jim Foley
GVU Center
College of Computing
Georgia Tech
March 2004

PREFACE TO THE SECOND EDITION

It has been more than 10 years since the first edition of *3D User Interfaces: Theory and Practice*, and the field has certainly changed over that time. The original edition was highly regarded, but perhaps ahead of its time since 3D user interfaces were confined to university research labs and some industrial research labs. However, over the last 10 years with technological breakthroughs in both hardware and software, we see that the commercial sector has brought virtual and augmented reality displays, mobile devices, gaming consoles, and even robotic platforms to market that require 3D user interface technology to provide useful, powerful, and engaging user experiences in a variety of different application domains. Meanwhile, this commodity hardware and software has revitalized university research labs doing work with 3D user interfaces and is now making it more affordable to set up virtual and augmented reality labs in the classroom, making the technology much more accessible to students at the graduate, undergraduate, and high school levels. These developments make the content of this book more relevant than ever and more useful to a wider audience of researchers, developers, hobbyists, and students. Thus, we believe releasing a second edition now is timelier than ever.

Given the changes described above, we needed to significantly revamp the book to not only reflect current needs of our readers (both old and new), but to update the material given the plethora of research into 3D user interface hardware and software that has been performed over the last decade. In addition, we decided to make the book application agnostic, since 3D user interfaces can be applied almost anywhere, given appropriate sensors to determine someone or something's position, orientation, and/or motion in space. We have made several changes in this edition that we feel make the book more inclusive and timely, while still maintaining its strength in discussing 3D user interface design from hardware to software to evaluation of 3D user interface techniques and applications.

This edition is organized into six distinct parts. Three chapters from the first edition—on wayfinding, symbolic input, and augmented reality—no longer stand on their own but have been fused into other parts of the book. We added two completely new chapters on human factors and general human-computer interaction to provide the reader with a more solid foundational background that can be used as base material for the 3D user interface chapters. We have also significantly revised each chapter with new material based on the latest research developments and findings. In addition, we felt it was important to have better cohesion between chapters from an application development perspective, so we decided to introduce two running case studies that describe a mobile augmented reality application and a first-person virtual reality gaming application. The content of these case studies will appear at the end of each chapter in Parts III, IV, and V of the book so the reader can see how the material can be utilized in these specific applications and how 3D user interface design is employed in each part of the 3D application.

In this edition, Part I has two chapters. The first is an introduction to the concepts of 3D user interfaces, while Chapter 2 provides a historical background into the field and lays out a roadmap for 3D user interfaces and how they relate to other fields. This is also where we introduce the two case studies that will be discussed throughout the book. Part II provides background material on the human factors aspects of interfaces in general and 3D user interfaces in particular, with emphasis on the human sensory system and human cognition (Chapter 3) and a general introduction to the field of human-computer interaction (Chapter 4) that can be used as a basis for understanding the various 3D user interface concepts and ideas we present in later parts of the book. Part III delves into 3D user interface hardware, including output devices for the visual, auditory, and haptic/tactile systems (Chapter 5), and input devices used in 3D user interfaces, with specific emphasis on obtaining 3D position and orientation and motion information about the user in physical space (Chapter 6). We consider Part IV the core of the book because its focus is on the fundamental 3D interaction tasks used in the majority of 3D user interfaces. In this part, Chapter 7 describes techniques for 3D object selection and manipulation while Chapter 8 delves into navigation and wayfinding techniques for moving through both virtual and physical spaces. Finally, Chapter 9 focuses on different system control techniques that can be used to change application state, issue commands and provide overall input to a 3D application. Part V describes strategies for 3D user interface design and evaluation, with Chapter 10 examining different design strategies for choosing and developing 3D user interfaces, and Chapter 11 covering the all-important aspects of 3D user interface evaluation, a critical component of the development of a 3D user interface technique or application. Finally, Part VI contains Chapter 12, which looks into the future of 3D user interfaces by providing a discussion of the open research problems that we need to solve to move the field forward.

As with the first edition, we offer numerous **guidelines**—practical and proven advice for the designer and developer. Guidelines are indicated in the text like this:

> **Tip**
>
> Follow the guidelines in this book to help you design usable 3D UIs.

The second edition of *3D User Interfaces: Theory and Practice* can be used in several different ways, depending on the reader's goals and intentions. For the student with no experience in human-computer interaction, the entire book can be used in an introductory 3D user interface design course at the graduate or undergraduate level. Those students who have had some background in human-computer interaction could essentially skip Part II of the book without loss of generality to support a more detailed course on 3D user interfaces.

Developers and 3D application designers can use the book for inspiration and guidance in the design, implementation, and evaluation of applications with 3D UIs. In the design process, developers can choose appropriate hardware from Part III, choose specific interaction techniques from Part IV, and learn how to evaluate their techniques and applications from Part V. Developers can also get inspiration from Chapter 10 on how best to go about designing their 3D application. It is our hope that developers, especially in the virtual and augmented reality communities, will use this material so they don't have to "reinvent the wheel" when developing their applications.

Finally, researchers can use the book as a comprehensive collection of related and prior work to help them understand what has been done in the field, ensure their research ideas are novel, get inspiration to tackle new problems in 3D user interfaces, and act as a one-stop shop for all things 3D UI. Chapter 12 would be especially useful for researchers who are looking for significantly challenging problems to explore.

Register your copy of *3D User Interfaces: Theory and Practice* at informit.com for convenient access to downloads, updates, and corrections as they become available. To start the registration process, go to informit.com/register and log in or create an account. Enter the product ISBN 9780134034324 and click Submit. Once the process is complete, you will find any available content under "Registered Products."

PREFACE TO THE FIRST EDITION

An architect sits in her home office, putting the final touches on the design of the new entrance to the city park. A three-dimensional virtual model of the park appears in front of her on the desk's surface. She nudges a pathway slightly to the right to avoid a low-lying area, and then makes the model life-size so she can walk along the path to view the effect. "Those dark colors on the sign at the entrance are too foreboding," she thinks, so she quickly changes the color palette to brighter primary colors. She looks up and notices that the clients are arriving for the final design review meeting. They are located in other offices around the city, but they can all view the 3D model and make suggested changes, as well as communicate with one another. "What's the construction plan?" asks one of the clients. The architect starts an animation showing the progress of the project from start to finish. "That first step may not work," says the client. "The excavation is much too close to the existing playground. Let me show you." He looks out his window, which has a view of the park, and overlays the virtual construction plan on it. "You're right," says the architect, "let's plan to move the playground slightly—that will be much cheaper than changing the construction site." After viewing the effects of the change, all agree that this plan will work, and the meeting adjourns.

This scenario and others like it illustrate the enormous potential of 3D environments and applications. The technology to realize such a vision is available now, although it will certainly be improved. But the scenario also leaves out a great deal of information—information that is crucial to making this dream a reality. How did the architect load the park model, and how does she manipulate her view of it? What technique is used to change the pathway? How can multiple clients all manipulate the model at the same time? How do the participants appear to each other in the virtual space? How is the speed and playback of the animation controlled? How did the client instruct the system to merge the real and virtual scenes?

These questions all relate to the design of the user interface (UI) and interaction techniques for this 3D application, an area that is usually given only a cursory treatment in futuristic films and books. The scenarios usually either assume that all interaction between the user and the system will be "natural"—based on techniques like intuitive gestures and speech—or "automatic"—the system will be so intelligent that it will deduce the user's intentions. But is this type of interaction realistic, or even desirable?

This book addresses the critical area of **3D UI design**—a field that seeks to answer detailed questions, like those above, that make the difference between a 3D system that is usable and efficient and one that causes user frustration, errors, and even physical discomfort. We present practical information for developers, the latest research results, easy-to-follow guidelines for the UI designer, and relevant application examples. Although there are quite a few books devoted to UIs in general and to 2D UI design in particular, 3D UIs have received significantly less attention. The results of work in the field are scattered throughout numerous conference proceedings, journal articles, single book chapters, and Web sites. This field deserves a reference and educational text that integrates the best practices and state-of-the-art research, and that's why this book was created.

How This Book Came to Be

The story of this book begins in April 1998, when Ivan Poupyrev and Doug Bowman were doctoral students at Hiroshima University and Georgia Tech, respectively, working on 3D interaction techniques for object manipulation in virtual environments (VEs). We started a lively email discussion about the design and usability of these techniques and about 3D UIs in general. Ivan, who was at the time a visiting research student at the University of Washington, suggested that the discussion would be even more profitable if other researchers in this new area could join in as well, and so the 3D UI mailing list was born. Since that time, over 100 researchers from around the globe have joined the list and participated in the discussion (to see an archive of all the list traffic or to join the list, check out http://www.3dui.org). Joe LaViola and Ernst Kruijff were two of the first people to join the list.

In August of that same year, Doug forwarded to the list a call for tutorials for the upcoming IEEE Virtual Reality Conference. After some discussion, Joe, Ivan, and Ernst agreed to join Doug to organize a tutorial on "The Art and Science of 3D Interaction." The tutorial was a big hit at the conference in Houston, and the four of us continued to present courses on the topic at ACM Virtual Reality Software and Technology 1999, IEEE VR 2000, and ACM SIGGRAPH 2000 and 2001.

After developing a huge amount of content for the notes supplements of these courses, we decided it would be silly not to compile and expand all of this information in book form. Furthermore, there was no way to include all the information available on 3D UIs in a one-day course. And that's why you're holding this book in your hands today—a book containing information on 3D UIs that can't be found in any other single source.

ACKNOWLEDGMENTS

This book would not have been possible without the hard work, support, and intelligence of a large group of people.

First, we offer our gratitude to the reviewers who gave their time and energy in improving the quality of the book. Their comments and suggestions have made the book more complete, more readable, and more useful. Thanks to Ben Shneiderman, Harry Hersh, D. Jay Newman, Jeff Pierce, Dieter Schmalstieg, and Bob Zeleznik for providing this invaluable service. Special thanks go to Jim Foley for his encouragement and support.

Next, we would like to thank our original editor at Addison-Wesley, Peter Gordon, for his invaluable advice and encouragement. The rest of the staff who worked on the first edition, including Bernie Gaffney, Amy Fleischer, Julie Nahil, Heather Mullane, and Curt Johnson have also been extremely helpful. Thanks also to Simone Payment and Carol Lallier for their competent and professional work during the production phase.

We thank Laura Lewin for her work on the second edition, in addition to Olivia Basegio and Susan Zahn. Getting this significantly updated new edition to print was almost as large of a task as a brand new book!

All of us would like to personally thank our colleagues in the 3D UI community for their fruitful discussions and collaborations. They include Mark Mine, Robert Lindeman, Matthew Conway, Ken Hinckley, Shumin Zhai, Kiyoshi Kiyokawa, Chris Shaw, Mark Billinghurst, Rudy Darken, Pablo Figueroa, Bernd Fröhlich, Steve Feiner, Wolfgang Stürzlinger, Martin Hachet, Yoshifumi Kitamura, Eric Ragan, Tobias Höllerer, Ravin Balakrishnan, and many others.

Portions of this material are based upon work supported by the National Science Foundation. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF). Portions of this material are based on work supported by the Office of Naval Research. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research (ONR).

Joe LaViola: During the writing of the first edition of this book, I was just a PhD student, and over the last decade or so, having become a professor, my work in 3D user interfaces has been more about my students' efforts here at the University of Central Florida than anything I could have done myself. Thus, I want to thank the students who have worked in the Interactive Systems and User Experience Lab including but not limited to Paul Varcholik, Emiko Charbonneau, Jared Bott, Salman Cheema, Jeff Cashion, Brian Williamson, Tad Litwiller, Andrew Miller, Arun Kulshreshth, Sara Buchanan, Travis Cossairt, Eugene Taranta, Corey Pittman, Kevin Pfeil, Andres Vargas, Seng Lee Koh, Conner Brooks, Michael Hoffman, Jim Crowley, and Juliet Norton. I also want to thank Chad Wingrave who was a postdoc with me for several years and was instrumental in moving 3D user interface research forward in my lab.

I want thank my PhD thesis advisor, colleague, and friend, Andries van Dam, for mentoring me over the years and always giving me goals to shoot for. Thanks also go to my family for their love and support, including my wife, Michelle, and my two beautiful daughters, Juliana, and Josephine, my Mom and Dad, my brother, Jamie, and the rest of my extended family. Without them, I could not have taken on such a large project.

I need to also thank my current and former friends and colleagues at Brown University, including John Hughes, David Laidlaw, Robert Zeleznik, Daniel Keefe, Daniel Acevedo Feliz, Andrew Forsberg, Loring Holden, Tim Miller, Steve Dollins, Lee Markosian, David Karelitz, Tim Rowley, Christine Waggoner, and all the members of the Brown University Computer Graphics Group. They were instrumental in helping with the first edition of this book.

Finally, I want to thank my coauthors, Doug, Ernst, Ivan, and Ryan, for their friendship and collaboration on the second edition of *3D User Interfaces: Theory and Practice*. It took us over a decade to decide to do this revision, and it felt almost as long to finish, but the results, in my opinion, are quite thorough and valuable to the community. Their contributions to this effort make the book what it is.

Ernst Kruijff: First of all, my thanks go to Joe, Ryan, Doug, and Ivan for their great cooperation, help, and extensive discussions. Thanks also go to my wife and kids, my parents, my brother, and my sister-in-law for their support, each in their own way. Furthermore, my thanks go to my colleagues at igroup / Bauhaus University Weimar, the Virtual Environments group at Fraunhofer IMK, the Augmented Reality group at TU Graz, CURE, and the Institute of Visual Computing at BRSU. Special thanks (in chronological order) for extensive collaboration, support and inspiration to Holger Regenbrecht, Jakob Beetz, Hartmut Seichter, Martin Göbel, Bernd Fröhlich, Gerold Wesche, Gernot Heisenberg , Aeldrik Pander, Steffi Beckhaus, Dieter Schmalstieg, Eduardo Veas, Alexander Bornik, Erick Mendez, Manuela Waldner, Daniel Wagner, André Hinkenjann, Jens Maiero, Christina Trepkowski, Alexander Marquardt, Saugata Biswas, as well as external collaborators Bernhard Riecke, Wolfgang Stuerzlinger, Rob Lindeman, Ed Swan, Steve Feiner, Jason Orlosky, and Kiyoshi Kiyokawa.

Ryan McMahan: First, I would like to thank Doug, Joe, Ernst, and Ivan for including me on this little adventure. It was a great honor to work with these aficionados. I especially need to thank

Amanda, my fiancée, for enduring the weekends of writing and my battles with deadlines. Her unwavering love kept me motivated. Thanks go to my mother, JoAnn McMahan, my family, and my friends for their continued support and encouragement. I would also like to recognize my colleagues and students at UT Dallas, especially the members of the Future Immersive Virtual Environments (FIVE) Lab. Finally, I am extremely grateful to my father, the late Ralph McMahan.

Doug Bowman: Much gratitude is due to Joe, Ivan, Ernst, and Ryan for seeing this project through and for all their years of friendship and collaboration. I especially appreciate Ryan stepping in and making such a big contribution to this second edition when my bandwidth was so limited. I greatly appreciate my closest collaborators: Tobias Höllerer and Chris North; thanks to both of you for pushing me beyond where I thought I could go, challenging my thinking, providing opportunities, picking up my slack, and (most importantly) being my friends. Thanks also go to my colleagues and students at Virginia Tech, including Jake Socha, David Hicks, Todd Ogle, David Cline, Ben Knapp, Ron Kriz, Mehdi Setareh, Walid Thabet, Thomas Ollendick, David Cox, Debby Hix, John Kelso, Joe Gabbard, Chad Wingrave, Jian Chen, Nicholas Polys, Wendy Schafer, Marcio Pinho, Eric Ragan, Felipe Bacim, Regis Kopper, Tao Ni, Yi Wang, Bireswar Laha, Andrew Ray, Bert Scerbo, Cheryl Stinson, Mahdi Nabiyouni, Panagiotis Apostolellis, Wallace Lages, and Run Yu. Thanks to the whole Center for HCI community for their support, in particular to Andrea Kavanaugh. Past colleagues at Georgia Tech also deserve thanks. They include Larry Hodges, Drew Kessler, David Koller, Donald Johnson, Donald Allison, Brian Wills, Jean Wineman, Jay Bolter, Elizabeth Davis, Albert Badre, and Ben Watson. There are many other colleagues and friends who have shaped my career and helped me to where I am now—thanks for your many contributions.

Finally, I would like to thank my wife, Dawn, for her unfailing love and support. My kiddos—Drew, Caroline, Lucy, Laken, and David—have made my life full during this project. Thanks also to my extended family and friends, especially those at Grace Covenant Presbyterian Church. My work and life are all due to the grace of God. *Soli Deo gloria*.

Ivan Poupyrev: This book is result of many years of collaboration and friendship with my coauthors, and needless to say this book would be never possible without them. This collaboration started and grew while I was doing my PhD research that was conducted jointly between Hiroshima University in Japan and Human Interface Technology Laboratory (HIT Lab) at the University of Washington in Seattle. Thus, I am deeply indebted to everyone who supported me during my Hiroshima and Seattle years, including but not limited to Professor Tadao Ichikawa, who supervised my PhD thesis at Hiroshima University, Masahito Hirakawa, Bryn Holmes, Suzanne Weghorst, and Professor Tom Furness III who accepted me to the HIT Lab family, as well as Mark Billinghurst, Mark Phillips, Jennifer Feyma, and Chris Airola for being collaborators and friends who gave me life outside the lab. I am grateful to the Japanese government for providing me with the Monbusho Scholarship for conducting graduate studies in Japan.

Parts of the augmented reality material were developed as a part of my postdoctoral research conducted at ATR Media Integration and Communication Research Labs in Kyoto. I am grateful to the ATR community and leadership of Ryohei Nakatsu, Jun Ohya, Nobuji Tetsutani, as well as collaborators and friends including Jun Kurumisawa, Tatsumi Sakaguchi, Keiko Nakao, Lew Baldwin, Desney Tan, Sidney Fels, Michael Lyons, Tal Shalif, and many others.

The first edition of the book itself was written while I was a Researcher at the Sony Computer Science Laboratories in Tokyo, and I would like to extend my gratitude to Jun Rekimoto, Mario Tokoro, Hiroaki Kitano, and Yuimko Kawashima for their guidance and patience.

For the second edition of this book, my deepest gratitude goes to my coauthors, who carried the absolute majority of efforts in preparing this substantial revision of the book. The project started when I moved to Google ATAP with a mission to build and grow a new research initiative that subsequently spanned into product efforts which consumed all my time, severely limiting my ability to contribute to the second edition efforts. I am humbled and indebted to my coauthors for their understanding and support during my rollercoaster years at Google ATAP.

ABOUT THE AUTHORS

JOSEPH J. LAVIOLA, JR., Associate Professor of Computer Science, directs the Interactive Systems and User Experience Research Cluster of Excellence at the University of Central Florida.

ERNST KRUIJFF, Interim Professor for Computer Graphics and Interactive Systems at the Institute of Visual Computing, Bonn-Rhein-Sieg University of Applied Sciences, leads the 3DMi group's design of multisensory 3D user interfaces.

RYAN P. McMAHAN, Assistant Professor of Computer Science and of Arts, Technology, and Emerging Communication at UT Dallas, directs its Future Immersive Virtual Environments (FIVE) Lab.

DOUG A. BOWMAN, Professor of Computer Science at Virginia Tech, directs its 3D Interaction Research Group and Center for Human-Computer Interaction. He is an ACM Distinguished Scientist.

IVAN POUPYREV is Technical Program Lead working on advanced interaction research at Google's Advanced Technology and Products (ATAP) division.

CHAPTER 9

SYSTEM CONTROL

On a desktop computer, we input text and commands using 2D UI elements such as pull-down menus, pop-up menus, or toolbars on an everyday basis. These elements are examples of system control techniques—they enable us to send commands to an application, change a mode, or modify a parameter. While we often take the design of such techniques in 2D UIs for granted, system control and symbolic input are not trivial in 3D applications. Simply adapting 2D desktop-based widgets is not the always the best solution. In this chapter, we discuss and compare various system control and symbolic input solutions for 3D UIs.

9.1 Introduction

The issuing of commands is a critical way to access any computer system's functionality. For example, with traditional desktop computers we may want to save a document or change from a brush tool to an eraser tool in a painting application. In order to perform such tasks, we use system control techniques like menus or function keys on a keyboard. Designers of desktop and touch-based system control interfaces have developed a plethora of widely used and well-understood graphical user interface (GUI) techniques, such as those used in the WIMP (Windows, Icons, Menus, Point and Click) metaphor (Preece et al. 2002). While quite a few techniques exist, designing a 3D UI to perform system control can be challenging. In this chapter, we will provide an overview of system control methods, and review their advantages and disadvantages.

Although much of the real work in a 3D application consists of interaction tasks like selection and manipulation, system control is critical because it is the glue that lets the user control the interaction flow between the other key tasks in an application. In many tasks, system control is intertwined with symbolic input, the input of characters and numbers. For example, users may need to enter a filename to save their work or specify a numeric parameter for a scaling command. In this chapter, we will focus on system control and symbolic input concurrently instead of handling these tasks separately.

To be sure, 2D and 3D applications differ with respect to symbolic input. For example, in writing this book with a word processor, the core activity is symbolic input, accomplished by typing on a keyboard. This activity is interspersed with many small system control tasks—saving the current document by clicking on a button, inserting a picture by choosing an item from a menu, or underlining a piece of text by using a keyboard shortcut, just to name a few. Yet, within most 3D applications the focus is the opposite: users only input text and numbers occasionally, and the text and numbers entered usually consist of short strings. While this may change in the future with more effective techniques, the current state of affairs is centered on limited symbolic input to support system control tasks.

We can define system control as the user task in which commands are issued to

1. request the system to perform a particular function,
2. change the mode of interaction, or
3. change the system state.

The key word in this definition is **command**. In selection, manipulation, and travel tasks, the user typically specifies not only *what* should be done, but also *how* it should be done, more or less directly controlling the action. In system control tasks, the user typically specifies only *what* should be done and leaves it up to the system to determine the details.

In this chapter we consider system control to be an explicit instead of an implicit action. Interfaces from other domains have used methods to observe user behavior to automatically adjust the mode of a system (e.g., Celentano and Pittarello 2004; Li and Hsu 2004), but we will not focus on this breed of interfaces.

In 2D interfaces, system control is supported by the use of a specific **interaction style**, such as pull-down menus, text-based command lines, or tool palettes (Preece et al. 2002). Many of these interaction styles have also been adapted to 3D UIs to provide for a range of system control elements (see section 9.4), which may be highly suitable for desktop-based 3D UIs. 2D methods may also be appropriate in handheld AR applications, where the application often also relies on screen-based (touch) input. But for immersive applications in particular, WIMP-style interaction may not always be effective. We cannot assume that simply transferring conventional interaction styles will lead to high usability.

In immersive VR, users have to deal with 6-DOF input as opposed to 2-DOF on the desktop. These differences create new problems but also new possibilities for system control. In 3D UIs it may be more appropriate to use nonconventional system control techniques (Bullinger et al. 1997). These system control methods may be combined with traditional 2D methods to form hybrid interaction techniques. We will talk about the potential and implications of merging 2D and 3D techniques at several stages throughout this chapter.

9.1.1 Chapter Roadmap

Before describing specific techniques, we will consider two categories of factors that influence the effectiveness of all techniques: human factors and system factors. We then present a classification of system control techniques for 3D UIs (section 9.3). Next, we describe each of the major categories in this classification (sections 9.4–9.8). In each of these sections, we describe representative techniques, discuss the relevant design and implementation issues, discuss specific symbolic input issues, and provide guidance on the practical application of the techniques. In section 9.9, we cover multimodal system control techniques, which combine multiple methods of input to improve usability and performance. We conclude the chapter, as usual, with general design guidelines and system control considerations in our two case studies.

9.2 System Control Issues

In this section, we discuss higher-level characteristics of the human and of the system that influence the design and user experience of system control interfaces.

9.2.1 Human Factors

3D UI designers have to deal with a number of **perception** issues that can limit the user experience of a system control interface. Issues include visibility, focus switching, and the choice of feedback modalities.

Visibility is probably the most prevalent issue in 3D applications. In both VR and AR, system control elements (such as menus or buttons) can occlude the content of the environment. Scaling down system control elements could be an option if supported by the screen resolution, but this may result in (further) reduction of legibility. Another approach is to use semitransparent system control elements. While this may make content more visible, it may come at the cost of reduced visibility and readability of the system control element itself. Visibility issues particularly effect graphical widgets (see section 9.5).

Focus switching occurs when a system control element is visually decoupled from the area where the main interaction is performed. For example, if you are using a tablet to display menus while modeling in an immersive environment, you will need to switch focus between the tablet screen and the object you are working on. The visual system may literally have to adjust its focus (accommodation) when using the separate displays. In addition, if system control actions occur frequently, this may interrupt the flow of action. And especially when multiple displays are used, the sequence of actions in a task may influence performance (Kruijff et al. 2003; McMahan 2007). Collocated menus (see section 9.5.1) are one way to avoid this sort of switching.

In a 2D desktop GUI, mode change feedback is often a combination of haptic, auditory, and visual events: the user presses the mouse or touchpad and may feel a click (haptic), hear a click generated by the input device or operating system (auditory), and observe how the appearance of, for example, a button may change (visual). While it may seem straightforward to port the same feedback to a 3D application, this is often not the case. Visual button press feedback may easily remain unnoticed, while auditory feedback may be drowned out by ambient noise. Multimodal feedback is often a good choice to deal with these issues; we will take a closer look at this type of interface in section 9.9.

The main **cognitive** issue in system control is the functional breadth and depth of the system. Depending on the complexity of an application, a varying number of options (functions) may be available. These options are likely structured in different categories (breadth). Each category may have several options that may be further structured in subcategories (depth). Complex functional structures may cause users to be cognitively challenged to understand which options are available and where they can be found, and designers should be careful to create understandable classifications of functions and to hide rarely used functionality.

Ergonomic issues in system control interfaces include control placement, and the pose, grip, and motion types a particular device is used with. Shape, size, and location of controls can highly affect system control. For example, the button to trigger a system control action may not be easily reachable while a device is held with a particular grip, and thus pressing the button may require regrasping the device in a grip that is not optimal for other tasks. This is often the case in handheld AR setups: holding the device for accessing system controls with the thumbs, for example, is not compatible with the grasp used for viewing the augmented scene.

Designers should investigate the relationship between different grips and the accessibility of buttons. For more information, refer to Veas and Kruijff (2008, 2010), as well as standard industrial reference guides (Cuffaro 2013).

The motion required to perform a specific system control action is another ergonomic issue. For example, for a 1-DOF menu (see section 9.4.1), we need to consider how a user can rotate his wrist, as well as the placement and size of the menu items that will lead to comfortable and efficient selection.

Finally, when multiple devices are used for system control and other actions, designers should consider how switching between devices is accomplished, where devices may get stored ("pick up and put away"), and how the tasks are matched to specific devices. For example, while using a tablet with a pen might improve the control of detailed menus within an immersive environment, the flow of action can be disrupted when users need to switch frequently between the tablet and, for example, a 3D mouse for other tasks.

9.2.2 System Factors

As we noted above, system control is often the glue that holds together the application. High-level characteristics of the system are important to consider when designing system control interfaces. System characteristics can even dictate specific choices for system control. The main issues are the visual display devices, the input devices, and ambient factors.

Visual displays will impose specific perceptual boundaries, such as resolution, size, and luminance, that will affect what system control methods may be used and how well they will perform. For example, more complex applications can force the use of a secondary screen (such as a tablet) for system control, as system control would otherwise clutter and occlude the main screen.

The choice of input devices defines the possibilities for control mappings. General-purpose devices such as a stylus or joystick may work with a number of system control techniques. In other cases, you may need to introduce a secondary device or physically extend the primary input device to provide controls that are well suited for system control methods. Extending an existing device or designing a completely new device is not trivial, but it is certainly possible because of the wide availability of DIY approaches (see Chapter 6, "3D User Interface Input Hardware," section 6.6).

Ambient system factors such as noise, device constraints (e.g., no tethered devices possible, nowhere to place additional devices), or the motion range of a user may also affect the design of system control techniques and system control task performance. For example, ambient noise may cause recognition errors in speech interfaces, and such errors can reduce user performance and perceived usability.

9.3 Classification

Although there is a broad diversity of system control techniques for 3D UIs, many draw upon a small number of basic types or their combination. Figure 9.1 presents a classification of techniques organized around the main interaction styles. This classification was influenced by the description of nonconventional control techniques in McMillan et al. (1997). There is a device-driven logic underneath most styles: often, available devices drive the selection of a specific system control technique. We use the classification as the organizing structure for the next five sections.

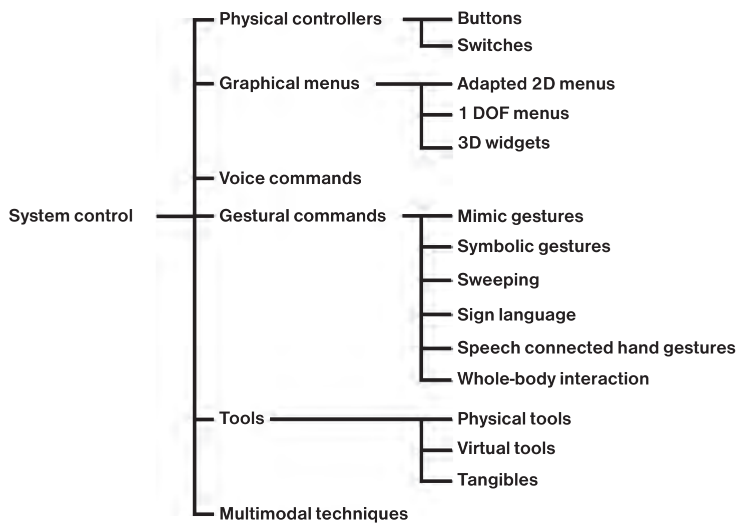

Figure 9.1 Classification of system control techniques.

As is the case in the other chapters on 3D interaction techniques, our classification is only one of many possibilities, and slightly different alternatives have been proposed (Lindeman et al. 1999; Dachselt and Hübner 2007).

9.4 Physical Controllers

Physical controllers such as buttons and switches offer a lightweight solution for performing system control, analogous to function keys in desktop systems.

9.4.1 Techniques

Buttons and switches are a direct way of changing a mode in an application. In contrast to using a pointing device to select, for example, an item from a menu, the physical controller allows the user to directly switch the mode between different states. Examples of well-known techniques are the function keys on a keyboard or buttons on a gaming device to which functions can be assigned. Gamers, for instance, often toggle between weapons in first-person shooters by pressing specific buttons.

9.4.2 Design and implementation issues

Buttons and switches can provide a useful and straightforward system control method; however, there are a number of issues that should be noted.

Placement and Form

When built-in controllers are used, you should carefully validate their placement and the potential need for regrasping a device to access the button, as discussed in section 9.2.1. While some devices are designed carefully from this perspective (e.g., Figure 9.2), other devices may have controllers placed at locations that are less accessible. Furthermore, devices such as tablets or phones often have very flat buttons that may be difficult to reach and control, making system control in handheld AR applications tricky to perform. Thus, it is not only the placement but also the physical form and quality (robustness) of buttons and switches that should be considered carefully.

Figure 9.2 A Thrustmaster flight joystick deploying numerous switches and buttons. (© Guillemot Corporation S.A. All rights reserved. Thrustmaster® is a registered trademark of Guillemot Corporation S.A.)

These controllers are often used eyes-off: finding the right controller can be achieved using proprioceptive feedback but also through the feel of a button when different controllers are located close to each other. When designing new devices or extending existing devices, it is important to carefully evaluate different variants.

Representation and Structure

Buttons and switches are not connected to any menu-like structure. Rather, their structure is based on the placement of buttons and their interrelationship. Button locations on many devices are more often defined by accessibility (ergonomic placement) than by functional structure. This means that mode feedback changes should be clearly communicated to the user, possibly through multiple sensory modalities. It may also make sense to place a small label or pictogram on the button itself to indicate its usage, allowing the user to visually explore the functionality of the buttons before operation.

9.4.3 Practical Application

Buttons and switches are highly useful in a number of situations, in particular when users need to switch frequently between functions. These function keys can be lightweight, quick, and straightforward if users know where to find each button and what it does. However, they can be a burden too when mode change is not clearly communicated, buttons are badly positioned, or there is an unknown functional mapping. In applications that are used for short durations by inexperienced users, function keys may be very useful, but only with a small functional space. For example, for public systems in theme parks, a very limited number of buttons can be easily understood and matched to simple tasks in the system. If users have the time and motivation to learn more complicated sets of functions, this may come with a great increase in performance. Game interfaces are a great example: gamers with prolonged experience with specific button layouts can achieve incredible speeds in performing system control actions. Finally, physical controllers can also be used for symbolic input, as buttons can be directly assigned to certain letters or numbers.

9.5 Graphical Menus

Graphical menus for 3D UIs are the 3D equivalent of the 2D menus that have proven to be a successful system control technique in desktop UIs. Because of their success and familiarity to users, many developers have chosen to experiment with graphical menus for 3D UIs. However, the design of graphical menus for 3D UIs comes with some unique challenges.

9.5.1 Techniques

Graphical menus used in 3D UIs can be subdivided into three categories:

- adapted 2D menus
- 1-DOF menus
- 3D widgets

Adapted 2D Menus

Menus that are simple adaptations of their 2D counterparts have, for obvious reasons, been the most popular group of 3D system control techniques. Adapted 2D menus basically function in the same way as they do on the desktop. Some examples of adapted 2D menus are pull-down menus, pop-up menus, floating menus, and toolbars. These menus are a common choice for more complex sets of functions. Menus are well suited for providing good structure for larger numbers of functions, and most users are familiar with the underlying principles (interaction style) of controlling a menu. On the other hand, these menus can occlude the environment, and users may have trouble finding the menu or selecting items using a 3D selection technique.

Figure 9.3 shows an example of an adapted 2D menu used in a Virtual Museum application in a surround-screen display. It allows a user to plan an exhibition by finding and selecting images of artwork. The menu is semitransparent to reduce occlusion of the 3D environment. Another example can be seen in Figure 9.4, showing a pie menu in a virtual environment. Pie menus can often be combined with marking-menu techniques (see section 9.7; Gebhardt et al. 2010)

Figure 9.3 A floating menu in the Virtual Museum application. (Photograph courtesy of Gerhard Eckel, Fraunhofer IMK)

Figure 9.4 Pie menu in immersive virtual environment. (Photograph courtesy of Thorsten Kuhlen, Virtual Reality & Immersive Visualization Group, RWTH Aachen)

There are numerous ways of adapting 2D menus by tuning aspects such as placement or input technique. For example, one adaptation of 2D menus that has been successful in 3D UIs is to attach the menus to the user's head. This way, the menu is always accessible, no matter where the user is looking. On the other hand, head-coupled menus can occlude the environment and potentially reduce the sense of presence.

Another method is attaching a menu to the user's hand in a 3D UI, assigning menu items to different fingers. For example, Pinch Gloves (see Chapter 6, Figure 6.25), can be used to interpret a pinch between a finger and the thumb on the same hand as a menu selection. An example of a finger-driven menu in AR is depicted in Figure 9.5 (Piekarski and Thomas 2003). Using Pinch Gloves, a typical approach is to use the nondominant hand to select a menu and the dominant hand to select an item within the menu. However, in many applications there will be more options than simple finger mapping can handle. The TULIP (Three-Up, Labels In Palm) technique (Bowman and Wingrave 2001) was designed to address this problem by letting users access three menu items at a time and using the fourth finger to switch to a new set of three items.

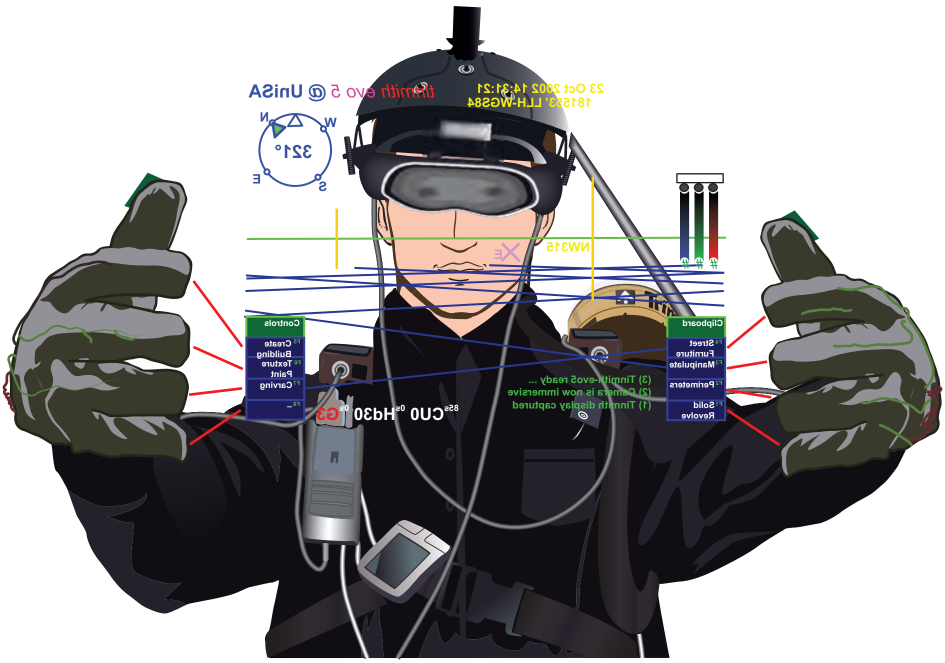

Figure 9.5 TINMITH menu using Pinch Gloves. (Adapted from Piekarski and Thomas 2003)

Another powerful technique is to attach the menu to a physical surface, which could be not just a phone or a tablet but also any other kind of surface. Figure 9.6 shows an example. These devices are often tracked. Finding the menu is then as easy as bringing the physical tablet into view. The physical surface of the tablet also helps the user to select the menu items, and the menu can easily be put away as well. However, the structure and flow of action may change considerably if the tablet is used for menus while a different input device is used for primary tasks.

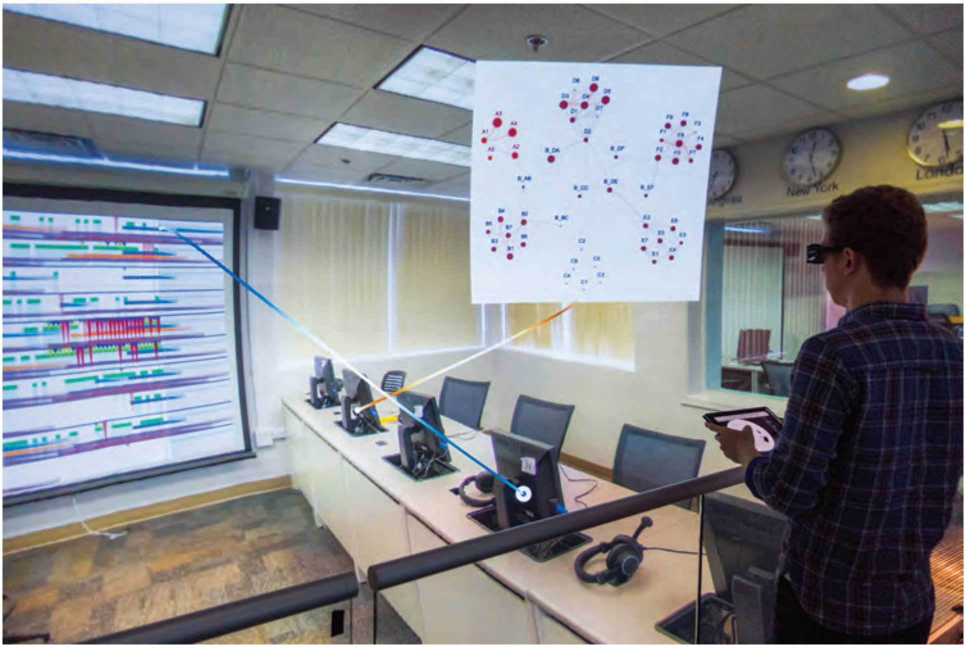

Figure 9.6 Tablet control of interactive visualizations layered on top of surround-view imagery in the UCSB Allosphere. Photograph shows left-eye view of stereo content viewed and controlled by the user. (Image courtesy of Donghao Ren and Tobias Höllerer)

1-DOF Menus

Selection of an item from a menu is essentially a one-dimensional operation. This observation led to the development of **1-DOF menus.** A 1-DOF menu is often attached to the user's hand, with the menu items arranged in a circular pattern around it; this design led to the name **ring menu** (Liang and Green 1994; Shaw and Green 1994). With a ring menu, the user rotates his hand until the desired item falls within a "selection basket." Of course, the hand rotation or movement can also be mapped onto a linear menu, but a circular menu matches well with the mental expectation of rotation. The performance of a ring menu depends on the physical movement of the hand and wrist, and the primary axis of rotation should be carefully chosen.

Hand rotation is not the only possible way to select an item in a 1-DOF ring menu. The user could also rotate the desired item into position with the use of a button or buttons on the input device: a dial on a joystick is one example of how this could be achieved. Another method is using tangible tiles, such as those used in the tangible skin cube (Figure 9.7, Lee and Woo 2010). 1-DOF menus can also be used eyes-off by coupling the rotational motion of the wrist to an audio-based menu. These kinds of techniques have also been used in wearable devices (Kajastila and Lokki 2009; Brewster et al. 2003).

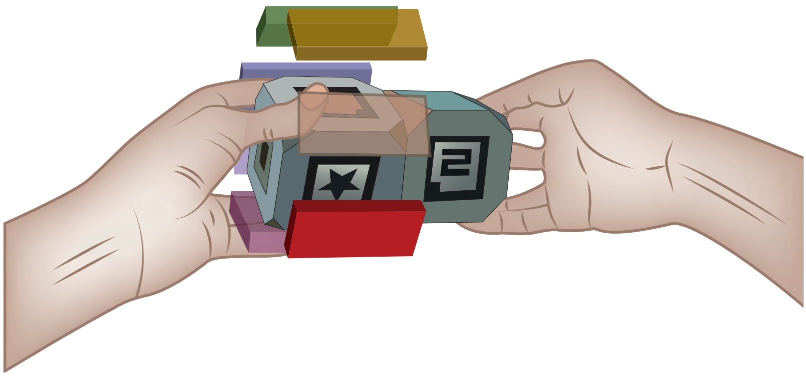

Figure 9.7 Ring menu implemented with tangible skin cubes. (Adapted from Lee and Woo 2010)

Handheld widgets are another type of 1-DOF menu that, instead of using rotation, use relative hand position (Mine et al. 1997). By moving the hands closer together or further apart, different items in the menu can be selected.

In general, 1-DOF menus are quite easy to use. Menu items can be selected quickly, as long as the number of items is relatively small and ergonomic constraints are considered. Because of the strong placement cue, 1-DOF menus also afford rapid access and use. The user does not have to find the menu if it is attached to his hand and does not have to switch his focus away from the area in which he is performing actions.

3D Widgets

The most exotic group of graphical menu techniques for system control is 3D widgets. They take advantage of the extra DOF available in a 3D environment to enable more complex menu structures or better visual affordances for menu entries. We distinguish between two kinds of 3D widgets: collocated (context-sensitive) and non-context-sensitive widgets.

With collocated widgets, the functionality of a menu is moved onto an object in the 3D environment, and geometry and functionality are strongly coupled. Conner and colleagues (1992) refer to widgets as "the combination of geometry and behavior." For example, suppose a user wishes to manipulate a simple geometric object like a box. We could design an interface in which the user first chooses a manipulation mode (e.g., translation, scaling, or rotation) from a menu and then manipulates the box directly. With collocated 3D widgets, however, we can place the menu items directly on the box—menu functionality is directly connected to the object (Figure 9.8). To scale the box, the user simply selects and moves the scaling widget, thus combining the mode selection and the manipulation into a single step. The widgets are context-sensitive; only those widgets that apply to an object appear when the object is selected. As in the example, collocated widgets are typically used for changing geometric parameters and are also often found in desktop modeling applications.

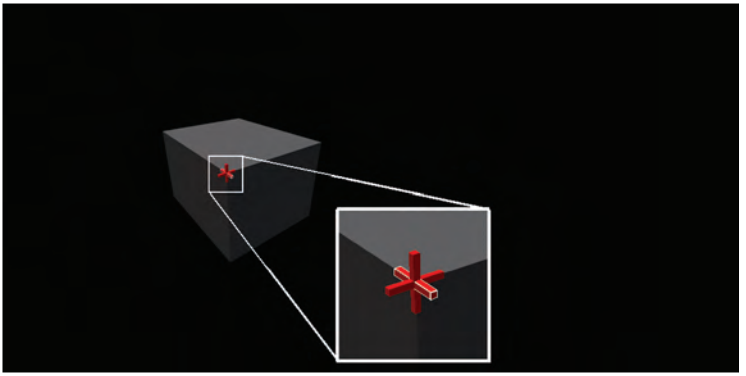

Figure 9.8 A 3D collocated widget for scaling an object. (Image courtesy of Andrew Forsberg, Brown University Computer Graphics Group)

The command and control cube, or C^3 (Grosjean et al. 2002), is a more general-purpose type of 3D widget (non-context-sensitive). The C^3 (Figure 9.9) is a $3 \times 3 \times 3$ cubic grid, where each of the 26 grid cubes is a menu item, while the center cube is the starting point. The user brings up the menu by pressing a button or making a pinch on a Pinch Glove; the menu appears, centered on the user's hand. Then the user moves his hand in the direction of the desired menu item cube relative to the center position and releases the button or the pinch. This is similar in concept to the marking menus (Kurtenbach and Buxton 1991) used in software such as Maya from Autodesk.

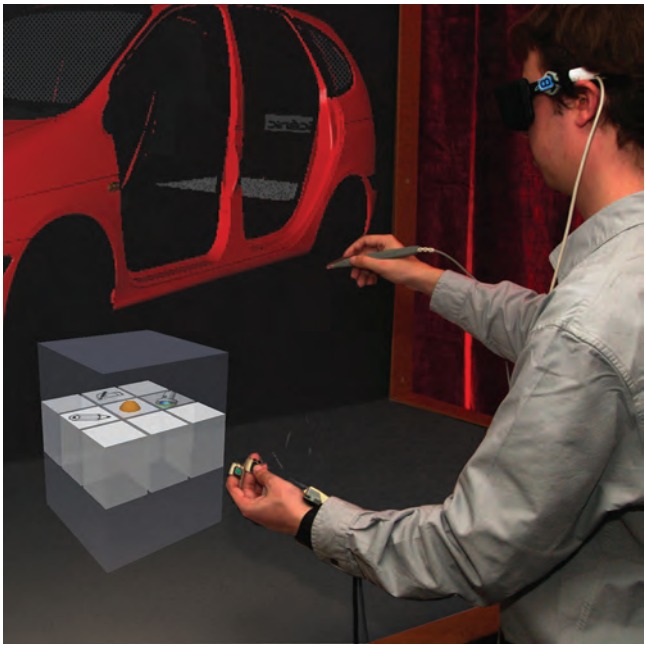

Figure 9.9 The command and control cube. (i3D-INRIA. Data © Renault. Photograph courtesy of Jerome Grosjean)

9.5.2 Design and Implementation Issues

There are many considerations when designing or implementing graphical menus as system control techniques in a 3D UI. In this section we will discuss the main issues that relate to placement, selection, representation, and structure.

Placement

The placement of the menu influences the user's ability to access the menu (good placement provides a spatial reference) and the amount of occlusion of the environment. We can consider menus that are **world-referenced**, **object-referenced**, **head-referenced**, **body-referenced**, or **device-referenced** (adapted from the classification in Feiner et al. 1993).

World-referenced menus are placed at a fixed location in the virtual world, while object-referenced menus are attached to an object in the 3D scene. Although not useful for most general-purpose menus, these may be useful as collocated 3D widgets. Head-referenced or body-referenced menus provide a strong spatial reference frame: the user can easily find the menu. Mine et al. (1997) explored body-referenced menus and found that the user's proprioceptive sense (sense of the relative locations of the parts of the body in space) can significantly enhance menu retrieval and usage. Body-referenced menus may even enable eyes-off usage, allowing users to perform system control tasks without having to look at the menu. Head- and body-referenced menus are mostly used in VR—while they may be applied in AR too, for example in an HWD setup, this is done infrequently. The last reference frame is the group of device-referenced menus. For instance, on a workbench, display menus may be placed on the border of the display device. The display screen provides a physical surface for menu selection as well as a strong spatial reference. Handheld AR applications often make use of device-referenced designs: while the user moves the "window on the world" around, the menus stay fixed on the display plane.

Handheld AR systems often use hybrid interfaces, in which 2D and 3D interaction methods are used in concert to interact with spatial content. Both due to familiarity and the fact that 2D techniques (such as menus) are frequently optimized for smaller screens, these methods are often an interesting option. Just because an application contains 3D content does not mean that all UI elements should be 3D. Nonetheless, hybrid interaction may introduce some limitations, such as device or context switching.

Non-collocated menus also result in focus switching, since menus are often displayed at a different location than the main user task. This problem can be exacerbated when menus are deliberately moved aside to avoid occlusion and clutter in an environment. Occlusion and clutter are serious issues; when a menu is activated, the content in the main part of the interaction space may become invisible. It is often hard to balance the issues of placement, occlusion, and focus switching. An evaluation may be needed to make these design decisions for specific systems.

Menu usage in AR may be even more challenging—with HWDs, the screen real estate can be limited by a narrow FOV. An even bigger problem is occlusion: even when handheld AR displays offer a wide FOV, placement of menus can still clutter the space and occlude the environment. Thus, AR system controls often need to be hidden after usage to free up the visual space.

Selection

Traditionally, desktop menus make use of a 2D selection method (mouse-based). In a 3D UI, we encounter the problem of using a 3D selection method with these 2D (or even 1D) menus. This can make system control particularly difficult. In order to address this problem, several alternative selection methods have been developed that constrain the DOF of the system control interface, considerably improving performance. For example, when an adapted 2D menu is shown, one can discard all tracker data except the 2D projection of the tracker on the plane of the menu. 2-DOF selection techniques such as image-plane selection also address this issue (see Chapter 7, "Selection and Manipulation"). Still, selection will never be as easy as with an inherently 2D input method. Alternatively, the menu can be placed on a physical 2D surface to reduce the DOF of the selection task, or a phone or a tablet can be used for menus.

Representation and Structure

Another important issue in developing a graphical menu is its representation: how are the items represented visually, and if there are many items, how are they structured?

The size of and space between items is very important. Do not make items and inter-item distances too small, or the user might have problems selecting the items, especially since tracking errors may exacerbate selection problems.

Application complexity is often directly related to the number of functions. Make sure to structure the interface by using either functional grouping (items with similar function are clustered) or sequential grouping (using the natural sequence of operations to structure items). Alternatively, one could consider using context-sensitive menus to display only the applicable functions.

Control coding can give an extra cue about the relations between different items and therefore make the structure and the hierarchy of the items clearer (Bullinger et al. 1997). Methods include varying colors, shapes, surfaces, textures, dimensions, positions, text, and symbols to differentiate items.

Finally, AR applications used in outdoor environments will be limited by visibility issues (Kruijff et al. 2010): both the bright conditions and limits in screen brightness affect visibility and legibility of menus. Color and size should thus be chosen carefully. Often it might help to use more saturated colors and larger sizes to increase visibility and legibility (Gabbard et al. 2007; see Figure 9.10).

Figure 9.10 Legibility with different backgrounds in augmented reality. This mockup image shows several frequently occurring issues: the leader line of label 1 (right) is difficult to see due to the light background, whereas label 2 (left) can be overlooked due to the low color contrast with the blue sign behind and above it. Label 3 can barely be read, as the label does not have a background color. While this reduces occlusion, in this case legibility is low due to the low contrast and pattern interferences of the background. (Image courtesy of Ernst Kruijff)

9.5.3 Practical Application

Graphical menu techniques can be very powerful in 3D UIs when their limitations can be overcome. Selection of menu items should be easy, and the menu should not overlap too much with the user's workspace in which the user is working.

Especially with applications that have a large number of functions, a menu is probably the best choice of all the system control techniques for 3D UIs. A good example of an application area that requires a large set of functions is engineering (Cao et al. 2006; Mueller et al. 2003). Medical applications represent another domain that has large functional sets (Bornik et al. 2006). Both of these can benefit from the hybrid approach: using 2D menus on devices such as tablets. Still, the approach of putting graphical menus on a remote device works only when users can see the physical world. For example, it will be useless in an immersive HWD-based system, except when the tablet is tracked and the menu is duplicated in the HWD. Finally, menus have often been used for symbolic input by showing, for example, a virtual keyboard in combination with a tablet interface to input text and numbers.

9.6 Voice Commands

The issuing of voice commands can be performed via simple speech recognition or by means of spoken dialogue techniques. Speech recognition techniques are typically used for issuing single commands to the system, while a spoken dialogue technique is focused on promoting discourse between the user and the system.

9.6.1 Techniques

A spoken dialogue system provides an interface between a user and a computer-based application that permits spoken interaction with the application in a relatively natural manner (McTear 2002; Jurafsky and Martin 2008).

The most critical component of a spoken dialogue system (and of simple speech recognition techniques) is the **speech recognition engine**. A wide range of factors may influence the speech recognition rate, such as variability among speakers and background noise. The recognition engine can be speaker-dependent, requiring initial training of the system, but most are speaker-independent, which normally do not require training. Systems also differ in the size of their vocabulary. The response generated as output to the user can confirm that an action has been performed or inform the user that more input is needed to complete a control command. In a spoken dialogue system, the response should be adapted to the flow of discourse (requiring a dialogue control mechanism) and generated as artificial speech.

Today's voice recognition systems are advanced and widespread. In particular, phones use voice commands and spoken dialogue. Therefore, as such, phones could be leveraged to allow for voice control in immersive VR and head-worn and handheld AR systems.

Many 3D UIs that use speech recognition also include other complementary input methods (Billinghurst 1998). These techniques are labeled **multimodal** and are discussed in section 9.8.

9.6.2 Design and Implementation Issues

The development of a 3D UI using speech recognition or spoken dialogue systems involves many factors. One should start by defining which tasks need to be performed via voice interfaces. For an application with a limited number of functions, a normal speech recognition system will probably work well. The task will define the vocabulary size of the speech engine—the more complex the task and the domain in which it is performed, the more likely the vocabulary size will increase. Highly complex applications may need conversational UIs via a spoken dialogue system in order to ensure that the full functionality of voice input is accessible. In the case of a spoken dialogue system, the design process should also consider what vocal information the user needs to provide in order for the system to determine the user's intentions.

Developers should be aware that voice interfaces are *invisible* to the user. The user is normally not presented with an overview of the functions that can be performed via a speech interface. In order to grasp the actual intentions of the user, one of the key factors is verification. Either by error correction via semantic and syntactic filtering (prediction methods that use the semantics or syntax of a sentence to limit the possible interpretation) or by a formal discourse model (question-and-answer mechanism), the system must ensure that it understands what the user wants.

Unlike other system control techniques, speech-based techniques initialize, select, and issue a command all at once. Sometimes another input stream (like a button press) or a specific voice command should be used to initialize the speech system. This disambiguates the start of a voice input and is called a push-to-talk system (see also Chapter 6, "3D User Interface Input Hardware," section 6.4.1). Error rates will increase when the application involves direct communication between multiple participants. For instance, a comment to a colleague can easily be misunderstood as a voice command to a system. Therefore, one may need to separate human communication and human–computer interaction when designing speech interfaces. Syntactic differences between personal communication and system interaction might be used to distinguish between voice streams (Shneiderman 2000).

9.6.3 Practical Application

Speech input can be a very powerful system control technique in a 3D UI; it is hands-free and natural. The user may first need to learn the voice commands—which is easy enough for a smaller functional set—before they can be issued. However, most of today's systems are powerful enough to understand complete sentences without learning. Moreover, most of today's phones already include a powerful speech recognition system that can be readily used. As such, for hybrid interfaces or handheld AR, voice recognition may be a good option. Speech is also well suited for symbolic input, as speech can be directly translated into text. As such, it is a lightweight method to dictate text or numbers. Also, as we will talk about in section 9.9, voice can be combined with other system control techniques to form a multimodal input stream to the computer.

In those domains where users need to rely on both hands to perform their main interaction tasks (e.g., a medical operating room), voice might be a useful system control asset. The doctor can keep using his hands, and since there is nothing to touch, the environment can be kept sterile. Still, continuous voice input is tiring and cannot be used in every environment.

Voice interface issues have been studied in many different contexts. For example, using speech commands for controlling a system via a telephone poses many of the same problems as using voice commands in a 3D UI. Please refer to Brewster (1998) for further discussion of issues involved in such communication streams.

9.7 Gestural Commands

Gestures were one of the first system control techniques for VEs and other 3D environments. Ever since early projects like Krueger's Videoplace (Krueger et al. 1985), developers have been fascinated by using the hands as direct input, almost as if one is not using an input device at all. Especially since the seemingly natural usage of gestures in the movie *Minority Report,* many developers have experimented with gestures. Gesture interfaces are often thought of as an integral part of *perceptual user interfaces* (Turk and Robertson 2000) or *natural user interfaces* (Wigdor and Wixon 2011). However, designing a truly well performing and easy-to-learn system is one of the most challenging tasks of 3D UI design. While excellent gesture-based interfaces exist for simple task sets that replicate real-world actions—gaming environments such as the Wii or XBox Kinect are good examples—more complex gestural interfaces for system control are hard to design (LaViola 2013).

Gestural commands can be classified as either **postures** or **gestures.** A posture is a static configuration of the hand (Figure 9.11), whereas a gesture is a movement of the hand (Figure 9.12), perhaps while it is in a certain posture. An example of a posture is holding the fingers in a V-like configuration (the "peace" sign), whereas waving and drawing are examples of gestures. The usability of gestures and postures for system control depends on the number and complexity of the gestural commands—more gestures imply more learning for the user.

Figure 9.11 Examples of postures using a Data Glove. (Photograph courtesy of Joseph J. LaViola Jr.)

Figure 9.12 Mimic gesture. (Schkolne et al. 2001; © 2001 ACM; reprinted by permission)

9.7.1 Techniques

One of the best examples to illustrate the diversity of gestural commands is Polyshop (later Multigen's Smart Scene; Mapes and Moshell 1995). In this early VE application, all interaction was specified by postures and gestures, from navigation to the use of menus. For example, the user could move forward by pinching an imaginary rope and pulling herself along it (the "grabbing the air" technique—see Chapter 8, "Travel," section 8.7.2). As this example shows, system control overlaps with manipulation and navigation in such a 3D UI, since the switch to navigation mode occurs automatically when the user pinches the air. This is lightweight and effective because no active change of mode is performed.

In everyday life, we use many different types of gestures, which may be combined to generate composite gestures. We identify the following gesture categories, extending the categorizations provided by Mulder (1996) and Kendon (1988):

- **Mimic gestures:** Gestures that are not connected to speech but are directly used to describe a concept. For example, Figure 9.12 shows a gesture in 3D space that defines a curved surface (Schkolne et al. 2001).
- **Symbolic gestures:** Gestures as used in daily life to express things like insults or praise (e.g., "thumbs up").
- **Sweeping**: Gestures coupled to the use of marking-menu techniques. Marking menus, originally developed for desktop systems and widely used in modeling applications, use pie-like menu structures that can be explored using different sweep-like trajectory motions (Ren and O'Neill 2013).
- **Sign language:** The use of a specified set of postures and gestures in communicating with hearing-impaired people (Fels 1994), or the usage of finger counting to select menu items (Kulshreshth et al. 2014).
- **Speech-connected hand gestures:** Spontaneous gesticulation performed unintentionally during speech or language-like gestures that are integrated in the speech performance. A specific type of language-like gesture is the **deictic gesture**, which is a gesture used to indicate a *referent* (e.g., object or direction) during speech. Deictic gestures have been studied intensely in HCI and applied to multimodal interfaces such as Bolt's "put that there" system (Bolt 1980).
- **Surface-based gestures:** Gestures made on multitouch surfaces. Although these are 2D, surface-based gestures (Rekimoto 2002) have been used together with 3D systems to create hybrid interfaces, an example being the Toucheo system displayed in Figure 9.13 (Hachet et al. 2011).

- **Whole-body interaction:** While whole-body gestures (motions) can be mimic or symbolic gestures, we list them here separately due to their specific nature. Instead of just the hand and possibly arm movement (England 2011), users may use other body parts like feet, or even the whole body (Beckhaus and Kruijff, 2004).

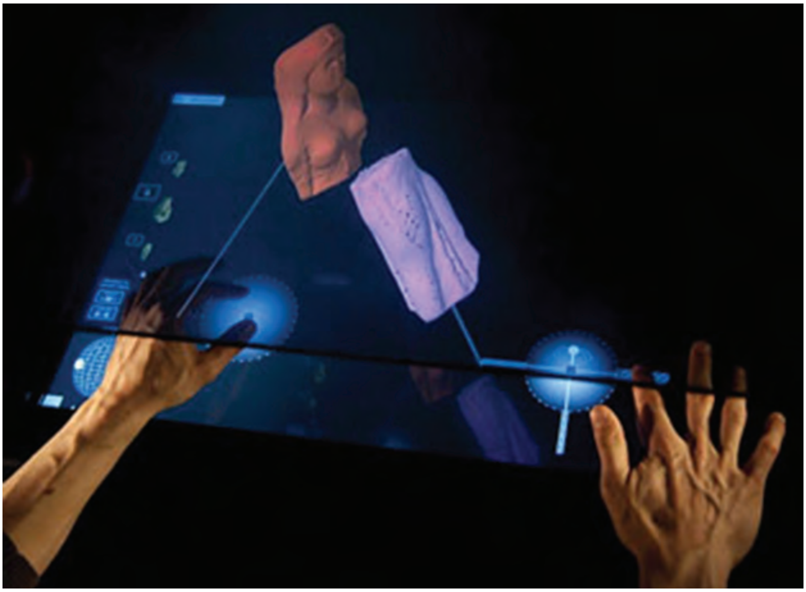

Figure 9.13 TOUCHEO—combining 2D and 3D interfaces. (© Inria / Photo H. Raguet).

9.7.2 Design and Implementation Issues

The implementation of gestural interaction depends heavily on the input device being used. At a low level, the system needs to be able to track the hand, fingers, and other body parts involved in gesturing (see Chapter 6, "3D User Interface Input Hardware," section 6.3.2. At a higher level, the postures and motions of the body must be recognized as gestures. Gesture recognition typically makes use of either machine learning or heuristics.

Gesture recognition is still not always reliable. Calibration may be needed but may not always be possible. When gestural interfaces are used in public installations, recognition should be robust without a calibration phase.

When a menu is accessed via a gestural interface, the lower accuracy of gestures may lead to the need for larger menu items. Furthermore, the layout of menu items (horizontal, vertical, or circular) may have an effect on the performance of menu-based gestural interfaces.

Gesture-based system control shares many of the characteristics of speech input discussed in the previous section. Like speech, a gestural command combines initialization, selection, and issuing of the command. Gestures should be designed to have clear *delimiters* that indicate the initialization and termination of the gesture. Otherwise, many normal human motions may be interpreted as gestures while not intended as such (Baudel and Beaudouin-Lafon, 1993). This is known as the gesture segmentation problem. As with push-to-talk in speech interfaces, the UI designer should ensure that the user really intends to issue a gestural command via some implicit or explicit mechanism (this sometimes is called a "push-to-gesture" technique). One option could be to disable gestures in certain areas, for example close to controllers or near a monitor (Feiner and Beshers, 1990).

The available gestures in the system are typically invisible to the user: users may need to discover the actual gesture or posture language (*discoverability*). Subsequently, the number and composition of gestures should be easy to learn. Depending on the frequency of usage of an application by a user, the total number of gestures may need to be limited to a handful, while the set of gestures for an expert user may be more elaborate. In any case, designers should make sure the cognitive load is reasonable. Finally, the system should also provide adequate feedback to the user when a gesture is recognized.

9.7.3 Practical Application

Gestural commands have significant appeal for system control in 3D UIs because of their important role in our day-to-day lives. Choose gestural commands if the application domain already has a set of well-defined, natural, easy-to-understand, and easy-to-recognize gestures. In addition, gestures may be more useful in combination with another type of input (see section 9.9). Keep in mind that gestural interaction can be very tiring, especially for elderly people (Bobeth et al. 2012). Hence, you may need to adjust your choice of system control methods based on the duration of tasks.

Figure 9.14 A user performing a 3D climbing gesture in a video game application. (Image courtesy of Joseph LaViola).

Entertainment and video games are just one example of an application domain where 3D gestural interfaces are becoming more common. This trend is evident from the fact that all major video game consoles and the PC support devices capture 3D motion from a user. In other cases, video games are being used as the research platform for exploring and improving 3D gesture recognition. Figure 9.14 shows an example of using a video game to explore what the best 3D gesture set would be for a first-person navigation game (Norton et al. 2010). A great deal of 3D gesture recognition research has focused on the entertainment and video game domain (Cheema et al. 2013; Bott et al. 2009; Kang et al. 2004; Payne et al. 2006; Starner et al. 2000).

Figure 9.15 A user controlling an unmanned aerial vehicle (UAV) with 3D gestures (Image courtesy of Joseph LaViola).

Medical applications used in operating rooms are another area where 3D gestures have been explored. Using passive sensing enables the surgeon or doctor to use gestures to gather information about a patient on a computer while still maintaining a sterile environment (Bigdelou et al. 2012; Schwarz et al. 2011). 3D gesture recognition has also been explored with robotic applications in the human-robot interaction field. For example, Pfeil et al. (2013) used 3D gestures to control unmanned aerial vehicles (UAVs; Figure 9.15). They developed and evaluated several 3D gestural metaphors for teleoperating the robot. Williamson et al. (2013) developed a full-body gestural interface for dismounted soldier training, while Riener (2012) explored how 3D gestures could be used to control various components of automobiles. Finally, 3D gesture recognition has recently been explored in consumer electronics, specifically for control of large-screen smart TVs (Lee et al. 2013; Takahashi et al. 2013).

Gesture interfaces have also been used for symbolic input. Figure 9.16 shows an example of such an interface, in which pen-based input is used. These techniques can operate at both the character-level and word-level. For example, in Cirrin, a stroke begins in the central area of a circle and moves through regions around a circle representing each character in a word (Figure 9.16).

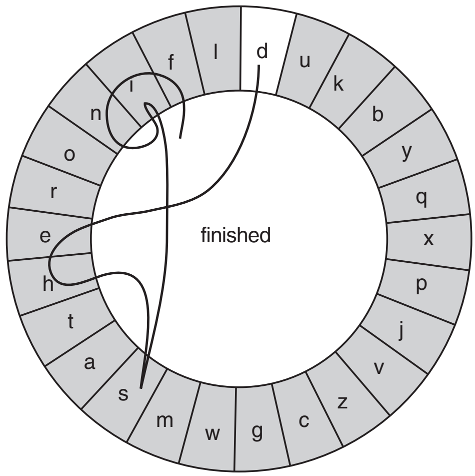

Figure 9.16 Layout of the Cirrin soft keyboard for pen-based input (Mankoff and Abowd 1998, © 1998 ACM; reprinted by permission

9.8 Tools

In many 3D applications, the use of real-world devices for 3D interaction can lead to increased usability. These devices, or their virtual representations, called **tools**, provide directness of interaction because of their real-world correspondence. Although individual tools may be used for selection, manipulation, travel, or other 3D interaction tasks, we consider a *set* of tools in a single application to be a system control technique. Like the tool palettes in many popular 2D drawing applications, tools in 3D UIs provide a simple and intuitive technique for changing the mode of interaction: simply select an appropriate tool.

Figure 9.17 Tool belt menu. Note that the tool belt appears larger than normal because the photograph was not taken from the user's perspective. (Photograph reprinted from Forsberg et al. [2000], © 2000 IEEE)

We distinguish between three kinds of tools: physical tools, tangibles, and virtual tools. Physical tools are a collection of real physical objects (with corresponding virtual representations) that are sometimes called **props.** A physical tool might be used to perform one function only, but it could also perform multiple functions over time. A user accesses a physical tool by simply picking it up and using it. Physical tools are a subset of the larger category of tangible user interfaces (TUI, Ullmer and Ishii 2001). Physical tools are tangibles that represent a real-world tool, while tangibles in general can also be abstract shapes to which functions are connected. In contrast, virtual tools have no physical instantiation. Here, the tool is a metaphor; users select digital representations of tools, for example by selecting a virtual tool on a tool belt (Figure 9.17).

9.8.1 Techniques

A wide range of purely virtual tool belts exist, but they are largely undocumented in the literature, with few exceptions (e.g., Pierce et al. 1999). Therefore, in this section we focus on the use of physical tools and TUIs as used for system control in 3D UIs.

Based on the idea of props, a whole range of TUIs has appeared. TUIs often make use of physical tools to perform actions in a VE (Ullmer and Ishii 2001; Fitzmaurice et al. 1995). A TUI uses physical elements that represent a specific kind of action in order to interact with an application. For example, the user could use a real eraser to delete virtual objects or a real pencil to draw in the virtual space. In AR, tools can also take a hybrid form between physical shape and virtual tool. A commonly used technique is to attach visual markers to generic physical objects to be able to render virtual content on top of the physical object. An example can be found in Figure 9.18, where a paddle is used to manipulate objects in a scene (Kawashima et al. 2000).

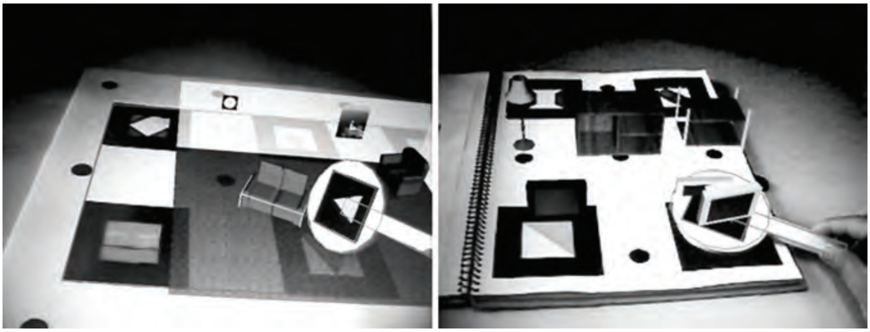

Figure 9.18 Using tools to manipulate objects in an AR scene. (Kato et al. 2000, © 2000 IEEE).

Figure 9.19 Visualization artifacts—physical tools for mediating interaction with 3D UIs. (Image courtesy of Brygg Ullmer and Stefan Zachow, Zuse Institute Berlin)

Figure 9.19 shows a TUI for 3D interaction. Here, Ethernet-linked interaction pads representing different operations are used together with radio frequency identification (RFID)-tagged physical cards, blocks, and wheels, which represent network-based data, parameters, tools, people, and applications. Designed for use in immersive 3D environments as well as on the desktop, these physical devices ease access to key information and operations. When used with immersive VEs, they allow one hand to continuously manipulate a tracking wand or stylus, while the second hand can be used in parallel to load and save data, steer parameters, activate teleconference links, and perform other operations.

A TUI takes the approach of combining representation and control. This implies the combination of both physical representations and digital representations, or the fusion of input and output in one mediator. TUIs have the following key characteristics (from Ullmer and Ishii 2001):

- Physical representations are computationally coupled to underlying digital information.
- Physical representations embody mechanisms for interactive control.
- Physical representations are perceptually coupled to actively mediated digital representations.

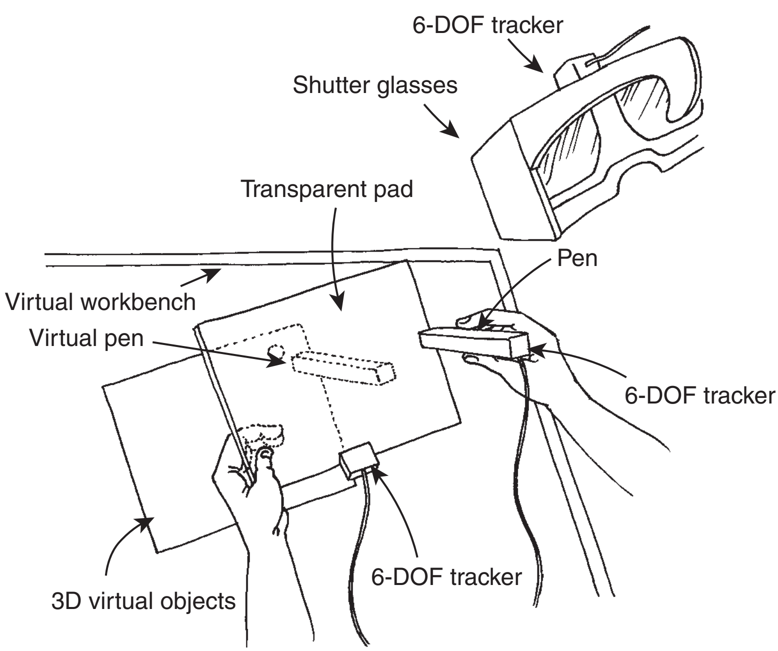

Figure 9.20 Personal Interaction Panel—combining virtual and augmented reality. (Adapted from Schmalstieg et al. 1999)

These ideas can also be applied to develop prop-based physical menus. In HWD-based VEs, for example, a tracked pen can be used to select from a virtual menu placed on a tracked tablet (Bowman and Wingrave 2001), which may also be transparent. An example of the latter approach is the Personal Interaction Panel (Schmalstieg et al. 1999), which combines a semitransparent Perspex plate to display a menu, combining VR and AR principles (Figure 9.20). Tablet computers can also be used; a tablet has the advantage of higher accuracy and increased resolution. The principal advantage of displaying a menu on a tablet is the direct haptic feedback to the user who interacts with the menu. This results in far fewer selection problems compared to a menu that simply floats in the VE space.

9.8.2 Design and Implementation Issues

The form of the tool communicates the function the user can perform with the tool, so carefully consider the form when developing props. A general approach is to imitate a traditional control design (Bullinger et al. 1997). Another approach is to duplicate everyday tools. The user makes use of either the real tool or something closely resembling the tool in order to manipulate objects in a spatial application.

Another important issue is the **compliance** between the real and virtual worlds; that is, the correspondence between real and virtual positions, shapes, motions, and cause-effect relationships (Hinckley et al. 1994). Some prop-based interfaces, like the Cubic Mouse (Fröhlich and Plate 2000), have demonstrated a need for a clutching mechanism. See Chapter 7 for more information on compliance and clutching in manipulation techniques.

The use of props naturally affords eyes-off operation (the user can operate the device by touch), which may have significant advantages, especially when the user needs to focus visual attention on another task. On the other hand, it also means that the prop must be designed to allow tactile interaction. A simple tracked tablet, for example, does not indicate the locations of menu items with haptic cues; it only indicates the general location of the menu.

A specific issue for physical menus is that the user may want to place the menu out of sight when it is not in use. The designer may choose to put a clip on the tablet so that the user can attach it to his clothing, may reserve a special place in the display environment for it, or may simply provide a handle on the tablet so it can be held comfortably at the user's side.

9.8.3 Practical Application

Physical tools are very specific devices. In many cases, they perform only one function. In applications with a great deal of functionality, tools can still be useful, but they may not apply to all the user tasks. There is a trade-off between the specificity of the tool (a good affordance for its function) and the amount of tool switching the user will have to do.

Public installations of VEs (e.g., in museums or theme parks) can greatly benefit from the use of tools. Users of public installations by definition must be able to use the interface immediately. Tools tend to allow exactly this. A well-designed tool has a readily apparent set of affordances, and users may draw from personal experience with a similar device in real life. Many theme park installations make use of props to allow the user to begin playing right away. For example, the Pirates of the Caribbean installation at Disney Quest uses a physical steering wheel and cannons. This application has almost no learning curve—including the vocal introduction, users can start interacting with the environment in less than a minute (Mine 2003).

9.9 Multimodal Techniques

While discussing the various system control techniques, we already mentioned that some techniques could be combined with others. In this section, we go deeper into the underlying principles and effects of combining techniques using different input modalities—**multimodal techniques**. Such techniques connect multiple input streams: users switch between different techniques while interacting with the system (LaViola et al. 2014). In certain situations, the use of multimodal system control techniques can significantly increase the effectiveness of system control tasks. However, it may also have adverse effects when basic principles are not considered. Here, we will shed light on different aspects of multimodal techniques that will help the developer make appropriate design choices for multimodal 3D UIs.

9.9.1 Potential Advantages

Researchers have identified several advantages of using multimodal system control techniques (mostly in the domain of 2D GUIs) that can also apply to 3D UIs:

- **Decoupling:** Using an input channel that differs from the main input channel used for interaction with the environment can decrease user cognitive load. If users do not have to switch between manipulation and system control actions, they can keep their attention focused on their main activity.
- **Error reduction and correction:** The use of multiple input channels can be very effective when the input is ambiguous or noisy, especially with recognition-based input like speech or gestures. The combination of input from several channels can significantly increase recognition rates (Oviatt 1999; Oviatt and Cohen 2000) and disambiguation in 3D UIs (Kaiser et al. 2003).
- **Flexibility and complementary behavior:** Control is more flexible when users can use multiple input channels to perform the same task. In addition, different modalities can be used in a complementary way based on the perceptual structure of the task (Grasso et al. 1998; Jacob and Sibert 1992).
- **Control of mental resources:** Multimodal interaction can be used to reduce cognitive load (Rosenfeld et al. 2001); on the other hand, it may also lead to less effective interaction

> because multiple mental resources need to be accessed simultaneously. For example, as Shneiderman (2000) observes, the part of the human brain used for speaking and listening is also the part used for problem solving—speaking consumes precious cognitive resources.

Probably the best-known multimodal technique is the famous "put-that-there" (Bolt 1980). Using this technique, users can perform manipulation actions by combining pointing with speech. Many others have used the same combination of gesture and speech (e.g., Figure 9.21), where speech is used to specify the command and gestures are used to specify spatial parameters of the command, all in one fluid action. In some cases, speech can be used to disambiguate a gesture, and vice versa.

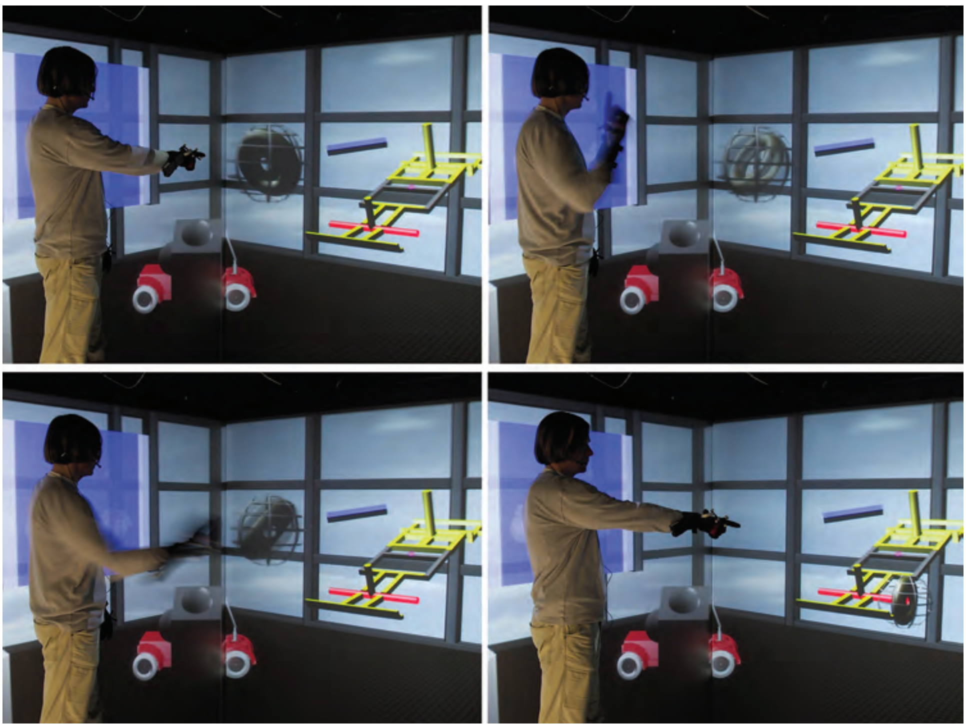

Figure 9.21 A car wheel is selected, rotated, and moved to its correct position using voice and gestures. (Photographs courtesy of Marc Eric Latoschik, AI & VR Lab, University of Bielefeld; Latoschik 2001)

Another possible technique is to combine gesture-based techniques with traditional menus, as in the "marking menus" technique. This means that novice users can select a command from a visual menu, while more experienced users can access commands directly via gestural input. This redundancy is similar to the use of keyboard shortcuts in desktop interfaces.

9.9.2 Design Principles

Designing multimodal system control techniques can be a complex undertaking. On a single technique level, the design guidelines from the various techniques discussed in the previous sections will apply. However, by combining techniques, several new issues come into play.

First, the combination of modalities will depend on the task structure: How can you match a specific task to a specific modality, and how does the user switch between modalities? Switching may affect the flow of action in an application—disturbances in the flow of action may lead to bad performance and lower user acceptance. A good way to verify flow of action is to perform a detailed logging that identifies how much time is spend on specific subparts in the task chain and compares this to single-technique (non-multimodal) performance. When combining two techniques, it can also make sense to do multimapping of tasks to modalities, that is, to allow users to perform a specific task using multiple methods.

Second, while multimodal techniques may free cognitive resources, this is not necessarily the case for all implementations. Cognitive load should thus be evaluated, either through self-assessment by a user (which only provides general indications) or through additional correlation with physiological measures that can assess stress or even brain activity. See Chapter 3, "Human Factors Fundamentals," section 3.4.3, for more information. In direct relation to cognitive load, attention is also an issue to consider: does the user need to pay much attention to using the combined technique (or accompanying visual or non-visual elements), or can the user remain focused on the main task?

9.9.3 Practical application

Using multimodal techniques can be useful in many situations. Complex applications can benefit from the complementary nature of multimodal techniques, allowing for more flexible input and potentially reducing errors. The reduction of errors is especially important for applications with limited or no time for user learning. For example, consider a public space installation: by supporting multiple modes of input, discovering the underlying functionality may become easier for users.

Also, some modalities may be easier to perform by certain classes of users: an elderly user may have difficulties with precise motor input but may be able to control an application by voice instead. This points to the general complementary behavior of multimodal techniques: when one input channel is blocked, either due to external factors or user abilities, another channel can be used. For instance, consider bright daylight limiting text legibility in an AR application or environmental noise limiting voice recognition. Being able to perform the task using another input channel can drastically increase performance.

Finally, multimodal techniques are applicable to scenarios that mimic natural behavior. In both realistic games and in applications that use a natural interaction approach, combinations of input modalities that mirror the ways we interact with other humans can improve the user experience.

9.10 Design Guidelines

Throughout this chapter, we have presented many design guidelines for specific 3D system control techniques. In this section, we summarize some general guidelines. Because there still have not been many empirical evaluations of system control techniques for 3D UIs, however, most of the guidelines should be regarded as rules of thumb.

> **Tip**
>
> Avoid disturbing the flow of action of an interaction task.

System control is often integrated with another 3D interaction task. Such a task structure forms a chain of actions. Because of this integration, system control techniques should be designed to avoid disturbing the flow of action of an interaction task. Lightweight mode switching, physical tools, and multimodal techniques can all be used to maintain the flow of action.

> **Tip**
>
> Prevent unnecessary focus switching and context switching.

One of the major interruptions to a flow of action is a change of focus. This may occur when users have to cognitively and/or physically switch between the actual working area and a system control technique, or even when they must look away to switch devices.

> **Tip**
>
> Design for discoverability.

Especially with "invisible" system control techniques like voice and gesture input, users will need to discover what is possible with the application. Make sure to aid this process by providing cues within the application or (alternatively) introduce a training phase.

> **Tip**
>
> Avoid mode errors.

Always provide clear feedback to the user so that she knows which interaction mode is currently active.

Tip

Use an appropriate spatial reference frame.

Placing your system control technique in the right position can make a big difference in its usability. Users often get distracted when searching for a way to change the mode of interaction or issue a command. If the controls are not visible at all, placed far away from the actual work area, or not oriented toward the user, the result will be wasted time. On the other hand, system controls attached to the user's hand, body, or a device are always available.

Tip

Structure the functions in an application and guide the user.

There are several good techniques for structuring the functionality of an application, including hierarchical menu structures and context-sensitive system control. In cases where the number of functions is so large that these techniques are not effective, it can make sense to place some of the system control functionality on another device, such as a tablet, where resolution and selection accuracy are less of an issue.

Tip

Consider using multimodal input.

Using multimodal input can provide more fluid and efficient system control but can also have its drawbacks.

Tip

3D is not always the best solution—consider hybrid interfaces.

Just because the applications are inherently 3D, 3D system control techniques are not necessarily the best solution. Often a 2D technique is more straightforward, especially if a tablet or phone can be used for controlling the menus. However, take care when designing interfaces that require a great deal of switching between modalities.

9.11 Case Studies

System control issues are critical to both of our case studies. If you have not yet read the introduction to the case studies, take a look at section 2.4 before reading this section.

9.11.1 VR Gaming Case Study

Given the description so far of our VR action-adventure game, you might think that the game is purely about direct interaction with the world and that there are no system control tasks to consider. Like most real applications, however, there are a variety of small commands and settings that the player needs to be able to control, such as saving a game to finish later, loading a saved game, pausing the game, choosing a sound effects volume, etc. Since these actions will happen only rarely and are not part of the game world itself, a simple 3D graphical point-and-click menu such as those described in section 9.5 will be sufficient. Pointing can be done with the dominant hand's controller, acting as a virtual laser pointer.

There are, however, two more prominent system control tasks that will occur often, and are more central to the gameplay. The first of these tasks is opening the inventory, so a collected item can be placed in it or so an item can be chosen out of it. An inventory (set of items that are available to the player) is a concept with a straightforward real-world metaphor: a bag or backpack. We can fit this concept into our story by having the hero (the player) come into the story already holding the flashlight and a shopping bag. The shopping bag can be shown hanging from the nondominant hand (the same hand holding the flashlight).

To open the bag (inventory), the player moves the dominant hand close to the bag's handle (as if he's going to pull one side of the bag away from the other). If the player is already holding an object in the dominant hand, he can press a button to drop it into the bag, which can then automatically close. This is essentially a "virtual tool" approach to system control (section 9.8), which is appropriate since we want our system control interface to integrate directly into the world of the game.

If the player wants to grab an item out of the inventory, it might be tricky to see all the items in the bag from outside (especially since, like many games of this sort, the inventory will magically be able to hold many items, some of which might be larger than the bag itself). To address this, we again draw inspiration from the game *Fantastic Contraption*, which provides a menu that's accessed when the player grabs a helmet and puts it over his head, placing the player in a completely different menu world. Similarly, we allow the player to put his head inside the bag, which then grows very large, so that the player is now standing inside the bag, with all the items arrayed around him (a form of graphical menu), making it easy to select an object using the selection techniques described in section 7.12.1. After an item is selected, the bag shrinks back to normal size and the player finds himself standing in the virtual room again.

The second primary in-game system control task is choosing a tool to be used with the tool handle on the player's dominant hand. As we discussed in the selection and manipulation chapter, the user starts with a remote selection tool (the "frog tongue"), but in order to solve the puzzles and defeat the monsters throughout the game, we envision that many different tools might be acquired. So the question is how to select the right tool mode at the right time. Again, we could use a very physical metaphor here, like the virtual tool belt discussed in section 9.8, where the player reaches to a location on the belt to swap one tool for another one. But this action is going to occur quite often in the game and will sometimes need to be done very quickly (e.g., when a monster is approaching and the player needs to get the disintegration ray out NOW). So instead we use the metaphor of a Swiss army knife. All the tools are always attached to the tool handle, but only one is out and active at a time. To switch to the next tool, the player just flicks the controller quickly up and down. Of course, this means that the player might have to toggle through several tool choices to get to the desired tool, but if there aren't too many (perhaps a maximum of four or five), this shouldn't be a big problem. Players will quickly memorize the order of the tools, and we can also use sound effects to indicate which tool is being chosen, so switching can be done eyes-free.

Key Concepts

- Think differently about the design of system control that's part of gameplay and system control that's peripheral to gameplay.
- When there are few options to choose from, a toggle that simply rotates through the choices is acceptable (and maybe even faster), rather than a direct selection.
- System control doesn't have to be boring, but be careful not to make it too heavyweight.

9.11.2 Mobile AR Case Study

While many AR applications tend to have lower functional complexity, the HYDROSYS application provided access to a wider range of functions. The process of designing adequate system control methods revealed issues that were particular for AR applications but also showed similarities to difficulties in VR systems.

As with all 3D systems, system control is highly dependent on the display type and input method. In our case, we had a 5.6" screen on the handheld device, with a 1024 x 600 resolution, and users provided input to the screen using finger-touch or a pen. The interface provided access to four different task categories: data search, general tools, navigation, and collaboration tools. As each category included numerous functions, showing all at once was not a viable approach—either the menu system would overlap most of the augmentations, or the menus would be illegible. Thus, we had to find a screen-space-effective method that would not occlude the environment.

We created a simple but suitable solution by separating the four groups of tasks. We assigned each task group an access button in one of the corners. When not activated, a transparent button showing the name of the task group was shown. We chose a transparent icon to limit occlusion of the augmented content. The icon did have legibility issues with certain backgrounds (see the bottom of Figure 9.22), a problem typical for AR interfaces (Kruijff et al. 2010). This was not a major problem, since users could easily memorize the content of the four icons.

Once a menu button was selected, a menu would appear at one of the four borders of the screen, leaving the center of the screen visible. In this way, users could have a menu open while viewing the augmented content. We used straightforward 2D menus with icons, designed to be highly legible under different lighting conditions that occur in outdoor environments. Menu icons combined visual representations and text, as some functions could not be easily represented by visual representation alone. In some cases, we could not avoid pop-up lists of choices—only a limited number of icons could be shown on a horizontal bar—but most of the menus were space-optimized. One particular trick we used was the filtering of menu options. In the data search menu bar (see the top of Figure 9.22), we used the principle of *guided exploration*. For example, certain types of visualizations were only available for certain types of sensor data: when a user selected a sensor data type, in the next step, only the appropriate visualization methods were shown, saving space and improving performance along the way.

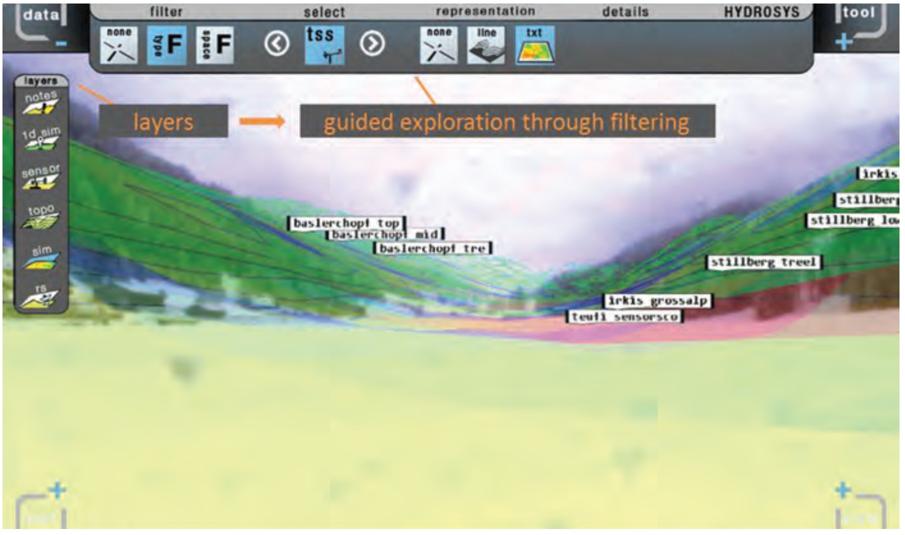

Figure 9.22 Example of a menu bar, in this case the data search menu, using the guided exploration approach. (Image courtesy by Ernst Kruijff and Eduardo Veas).

Key Concepts

- Perceptual issues: Visibility and legibility affect the design methods of AR system control in a way similar to those of general 2D menu design. However, their effects are often stronger, since AR interfaces are highly affected by display quality and outdoor conditions.
- Screen space: As screen space is often limited, careful design is needed to optimize the layout of system control methods to avoid occlusion of the augmentations.

9.12 Conclusion

Though system control methods for 3D UIs have been developed and used extensively, many issues are still open for further research. We have discussed a wide range of techniques, but the design space for such techniques is virtually limitless. We expect to see many new and interesting 3D system control techniques, not only within the categories described here but also in new categories that have not yet been invented. There is also a lack of good empirical evidence for the user experience of various system control techniques at the present time—further UX evaluations are needed in order to validate current design guidelines and develop new ones. Nevertheless, this chapter has served to demonstrate the importance and complexity of system control interfaces and has presented a wide range of existing system control techniques for 3D UIs. This concludes our tour through 3D interaction techniques for the universal tasks. We now move on to general design and evaluation of 3D UIs in part V.

Recommended Reading

For more details on human factors see:

Salvendy, G. (ed.) (2012). *Handbook of Human Factors and Ergonomics*. Hoboken, NJ: John Wiley & Sons.

Kruijff, E., Swan II, E., and Feiner, S. (2010). "Perceptual issues in Augmented Reality Revisited." *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality,* 3–12.

An introduction to the origins of nonconventional control can be found in this book chapter:

Bullinger, R, H., Kern, P., and M. Braun (1997). "Controls." In G. Salvendy (ed.), *Handbook of Human Factors and Ergonomics,* 697–728. New York: John Wiley & Sons.

More details on graphics widgets can be found in:

Schmalstieg D., and Höllerer, T. (2016). *Augmented Reality: Principles and Practice.* Addison-Wesley.

More details on using voice as an input modality can be found in the following text:

Pieraccini, R. (2012). *The Voice in the Machine: Building Computers That Understand Speech*. Cambridge, MA: MIT Press.

Jurafsky, D., and J. Martin (2008) *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall.

More detail on gestures can be found int he following text:

Wigdor, D., and Wixon, D. (2011*). Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*. Burlington, MA: Morgan Kaufmann.

The following two papers provide more information on using tools as part of a 3D interface:

Ishii, H., and B. Ullmer (1997). Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms. *Proceedings of the ACM Conference on Human Factors in Computing Systems,* ACM Press, 234–241.

Hinckley, K., R. Pausch, J. Goble, and N. Kassell (1994). Passive Real-World Interfaces Props for Neurosurgical Visualization. *Proceedings of the 1994 ACM Conference on Human Factors in Computing Systems (CHI '94),* ACM Press, 452–458.

INDEX

Numbers

A

D

E

F

G

H

I

J-K

L

M

N

O

P

Q-R

S

T

U

W-X-Y-Z