

Virtual Environment Usability Assessment Methods Based on a Framework of Usability Characteristics

Kent O. Swartz

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in
partial fulfillment of the requirements for the degree of

Masters of Science
in
Computer Science

Dr. Deborah Hix, Chair
Dr. Manuel Pérez-Quñones
Dr. Ron Kriz

Aug 27, 2003
Blacksburg, VA

Keywords: Usability Engineering Methodology, Heuristic Evaluation, Virtual Environments,
Formative Evaluation

Copyright 2003, Kent O. Swartz

Virtual Environment Usability Assessment Methods Based on a Framework of Usability Characteristics

Kent Olen Swartz

(Abstract)

Developing economical yet effective methods of incorporating usability engineering as an integral part of software engineering is a primary focus of human-computer interaction (HCI) research. However, much HCI research has focused primarily on inspecting and evaluating applications supporting command-line or graphical user interface (GUI) interaction styles. With the dramatic increase in virtual environment (VE) research in recent years, the HCI community is beginning to place an added emphasis on creating methodologies to ensure usability in VE development. While the demand for VE-specific usability engineering methods and criteria is dire as the amount of money invested by military, government, commercial, and industrial organizations continues to grow, widely accepted methodologies for assessing VE usability are, at this point in time, minimal. There has been a recent increase in research discussing the need of VE-specific usability engineering methodologies, but few research projects have concentrated their efforts on providing such methodologies. Therefore, application developers attempting to apply a user-centered design approach in constructing VEs must often perform largely ad-hoc assessments or in-house evaluations using existing non-VE-specific usability engineering methodologies.

The primary focus of this research was to develop a method to guide usability engineering of VEs. The strategy used to develop this usability evaluation method was to modify existing usability evaluation methodologies to support VE development by leveraging the results of previous VE usability research performed at Virginia Tech and elsewhere. The result was a VE-specific usability evaluation methodology that encompasses two existing usability assessment techniques: usability inspections and formative evaluations. We applied this methodology to Crumbs, an immersive visualization VE developed at The National Center for Supercomputing Applications (NCSA).

A multi-dimensional framework of VE usability characteristics was a topic of research at Virginia Tech. This framework provided the backbone for VE-specific modifications to the existing usability evaluation techniques proposed in this research. Framework design guidelines permitted usability specialists to perform guidelines-based usability inspections of Crumbs. Results gathered from the guidelines-based usability inspections were used not only to redesign the Crumbs user interface but also drive creation of a formative evaluation plan. Application of the methodology not only uncovered usability issues with Crumbs, but also provided invaluable information regarding the effectiveness of the methodology itself. We conclude this thesis by describing a usability evaluation methodology, called the Modified Concept Book Usability Evaluation Methodology, proposed to improve the usability evaluation methodology performed on Crumbs and other VEs. Our methodology was adapted from an established methodology for performing content analysis on a large volume of qualitative data.

Because the realm of VEs is so vast and diverse in application domains and devices, we do not claim that our methodology supports an exhaustive usability evaluation of all VEs. However, the proposed technique is a first attempt at modifying existing usability

evaluation methods, and therefore can be used as a launching pad for methodologies to evaluate other aspects of specific VE applications.

Table Of Contents

1	Acknowledgements.....	1
2	Introduction	2
2.1	Motivation and Problem Statement	2
2.2	Objective of Research.....	3
2.3	Approach to Research.....	3
2.4	Summary of Contributions.....	5
2.5	Limitations of Traditional Usability Methods	6
2.6	Document Organization.....	9
3	Related Work.....	10
3.1	Usability Engineering Methodologies	10
3.1.1	Automated and Formalism-based Methods	11
3.1.2	Usability Inspections.....	12
3.1.2.1	Guidelines-Based Evaluation.....	13
3.1.2.2	Heuristic Evaluation	14
3.1.3	Empirical Usability Evaluations	16
3.1.3.1	Formative Evaluation	17
3.2	VE Framework of Usability Characteristics	18
3.3	Concept Book Approach to Content Analysis.....	22
3.4	Other Related Work.....	26
3.4.1	Testbed Evaluation	27
3.4.2	Researching Quantitative and Qualitative Measures for Presence.....	28
3.4.3	VE Design Guidelines Development	30
3.4.4	Traditional Empirical Evaluation.....	31
3.4.5	Altered Empirical Evaluation	32
4	Initial VE-Specific Usability Evaluation Method	33
4.1	Introduction to Initial VE-Specific Usability Evaluation Method.....	33
4.2	Assessing Weaknesses of Initial VE-Specific Usability Evaluation Method.....	36
4.3	Making Modifications to Traditional Usability Evaluation Methods	37
5	Applying the Usability Evaluation Method to Crumbs	40
5.1	Crumbs Introduction.....	40
5.2	Guidelines-Based Evaluation of Crumbs	41
5.2.1	Method.....	41
5.2.2	Evaluators.....	42
5.2.3	Results.....	43
5.3	Formative Evaluation of Crumbs	47
5.3.1	Method.....	47
5.3.2	Participants.....	47
5.3.3	Equipment.....	48
5.3.4	User Tasks.....	49
5.3.5	Procedures.....	50
5.3.5.1	Pilot Study Procedures	51
5.3.5.2	Main Evaluation Procedures	53
5.4	Results	55
5.4.1	Strengths	55
5.4.1.1	Wand as selection metaphor.....	55
5.4.1.2	Menu as command metaphor	56
5.4.1.3	Inclusion of Audio Feedback	57
5.4.1.4	Location awareness in data volume	58
5.4.2	Usability Issues.....	59
5.4.2.1	Issue #1: Awareness of middle wand button for scale/mark mode	59
5.4.2.1.1	Description	59
5.4.2.1.2	Guidelines.....	60

5.4.2.1.3	Redesign Suggestion	60
5.4.2.2	Issue #2: Inconsistent use of direct manipulation in removing objects.....	61
5.4.2.2.1	Description	61
5.4.2.2.2	Guidelines.....	62
5.4.2.2.3	Redesign Suggestion	63
5.4.2.3	Issue #3: Inconsistent method of selection.....	63
5.4.2.3.1	Description	63
5.4.2.3.2	Guidelines.....	64
5.4.2.3.3	Redesign Suggestion	64
5.4.2.4	Issue #4: Awareness of current position in crumb placement task.....	65
5.4.2.4.1	Description	65
5.4.2.4.2	Guidelines.....	65
5.4.2.4.3	Redesign Suggestion	65
5.4.2.5	Issue #5: Occlusion of 3D widgets	65
5.4.2.5.1	Description	65
5.4.2.5.2	Guidelines.....	66
5.4.2.5.3	Redesign Suggestion	66
5.4.2.6	Issue #6: Inappropriate use or lack of audio annotations	67
5.4.2.6.1	Description	67
5.4.2.6.2	Guidelines.....	67
5.4.2.6.3	Redesign Suggestion	67
5.4.2.7	Issue #7: Arm movement facilitating cascading menus	68
5.4.2.7.1	Description	68
5.4.2.7.2	Guidelines.....	68
5.4.2.7.3	Redesign Suggestion	68
5.4.2.8	Issue #8: Menu truncation due to sword location.....	69
5.4.2.8.1	Description	69
5.4.2.8.2	Guidelines.....	69
5.4.2.8.3	Redesign Suggestion	69
5.4.2.9	Issue #9: Different crumb modifications use similar interactions	69
5.4.2.9.1	Description	69
5.4.2.9.2	Guidelines.....	70
5.4.2.9.3	Redesign Suggestion	71
5.4.2.10	Issue #10: Awareness of colormap object box value	71
5.4.2.10.1	Description	71
5.4.2.10.2	Guidelines.....	72
5.4.2.10.3	Redesign Suggestion	72
5.4.2.11	Issue #11: Use of color box metaphor in colormap object.....	73
5.4.2.11.1	Description	73
5.4.2.11.2	Guidelines.....	73
5.4.2.11.3	Redesign Suggestion	73
5.4.2.12	Issue #12: Two stepped load colormap values interaction.....	74
5.4.2.12.1	Description	74
5.4.2.12.2	Guidelines.....	75
5.4.2.12.3	Redesign Suggestion	75
5.4.2.13	Issue #13: Opacity object intuitiveness	75
5.4.2.13.1	Description	75
5.4.2.13.2	Guidelines.....	76
5.4.2.13.3	Redesign Suggestion	76
5.4.2.14	Issue #14: Opacity object loading two stepped process	77
5.4.2.14.1	Description	77
5.4.2.14.2	Guidelines.....	77
5.4.2.14.3	Redesign Suggestion	77
5.4.2.15	Issue #15: Usefulness of Sonification to Tasks	77

5.4.2.15.1	Description	77
5.4.2.15.2	Guidelines.....	78
5.4.2.15.3	Redesign Suggestion	78
5.4.2.16	Issue #16: Precision of crumb placement.....	78
5.4.2.16.1	Description	78
5.4.2.16.2	Guidelines.....	79
5.4.2.16.3	Redesign Suggestion	79
5.4.3	Conclusions.....	80
6	Assessment of Evaluation Process	81
6.1	Unique Usability Issues	81
6.2	Usability Issues Requiring Further Investigation	85
6.3	Assessing the Impact of Usability Issues Specified in the Inspection	86
7	Concept Book Usability Engineering Methodology	88
7.1	Weaknesses of the Initial Proposed Usability Methodology	88
7.1.1	Guideline-based Inspection	88
7.1.2	Formative Evaluation.....	90
7.1.3	Guideline-based Inspection and Formative Evaluation Coordination.....	91
7.2	Modifications to Initial Proposed Usability Methods.....	92
7.2.1	Guideline-based Inspection	92
7.2.1.1	Inclusion of Usability Specialists' Experience and Traditional HCI Guidelines.....	92
7.2.1.2	Usage Scenarios	93
7.2.1.3	Object Interaction Guide.....	93
7.2.1.4	Guidelines Reduction	94
7.2.2	Formative Evaluation.....	95
7.2.2.1	Qualitative Data	95
7.2.2.1.1	Strengthening Situational Awareness Measurement	96
7.2.2.1.2	Strengthening Presence Measurement.....	96
7.2.3	Modified Concept Book.....	97
7.2.3.1	Preparation for the Formative Evaluation.....	98
7.2.3.1.1	Briefing	99
7.2.3.1.2	Sampling.....	99
7.2.3.1.3	Associating	100
7.2.3.1.4	Hypothesis Development.....	100
7.2.3.2	Conducting the Formative Evaluation	102
7.2.3.2.1	Hypothesis Testing.....	102
7.2.3.2.2	Immersion (Recollection)	103
7.2.3.3	Interpreting and Recording Results.....	103
7.2.3.3.1	Categorizing.....	103
7.2.3.3.2	Incubation.....	104
7.2.3.3.3	Synthesis.....	104
7.2.3.3.4	Culling.....	104
7.2.3.3.5	Interpretation	105
7.2.3.3.6	Write	105
7.2.3.3.7	Rethink	105
8	Future Work	107
9	Appendices	109
9.1	Appendix A: Participant Permission Form	109
9.2	Appendix B: Crumbs CAVE Evaluation Pre-Test Survey.....	111
9.3	Appendix C: Participant Instructions.....	113
9.4	Appendix D: Task Description and Reasoning	115
	Appendix E: Auxiliary Time Recording Form.....	120
9.5	Appendix F: Feedback Questionnaire.....	121
9.6	Appendix G: Usability Specification Table	123
9.7	Appendix H: Data Collection Form – Time and Error Count	127
9.8	Appendix I: Data Collection Form – Participant Comments and Evaluator's Observations.....	128
	Appendix J: Task to category mapping.....	129

9.9 Appendix K: Raw Analytical Data Error Counts 131

9.10 Appendix L: Raw Analytical Data Completion Times 132

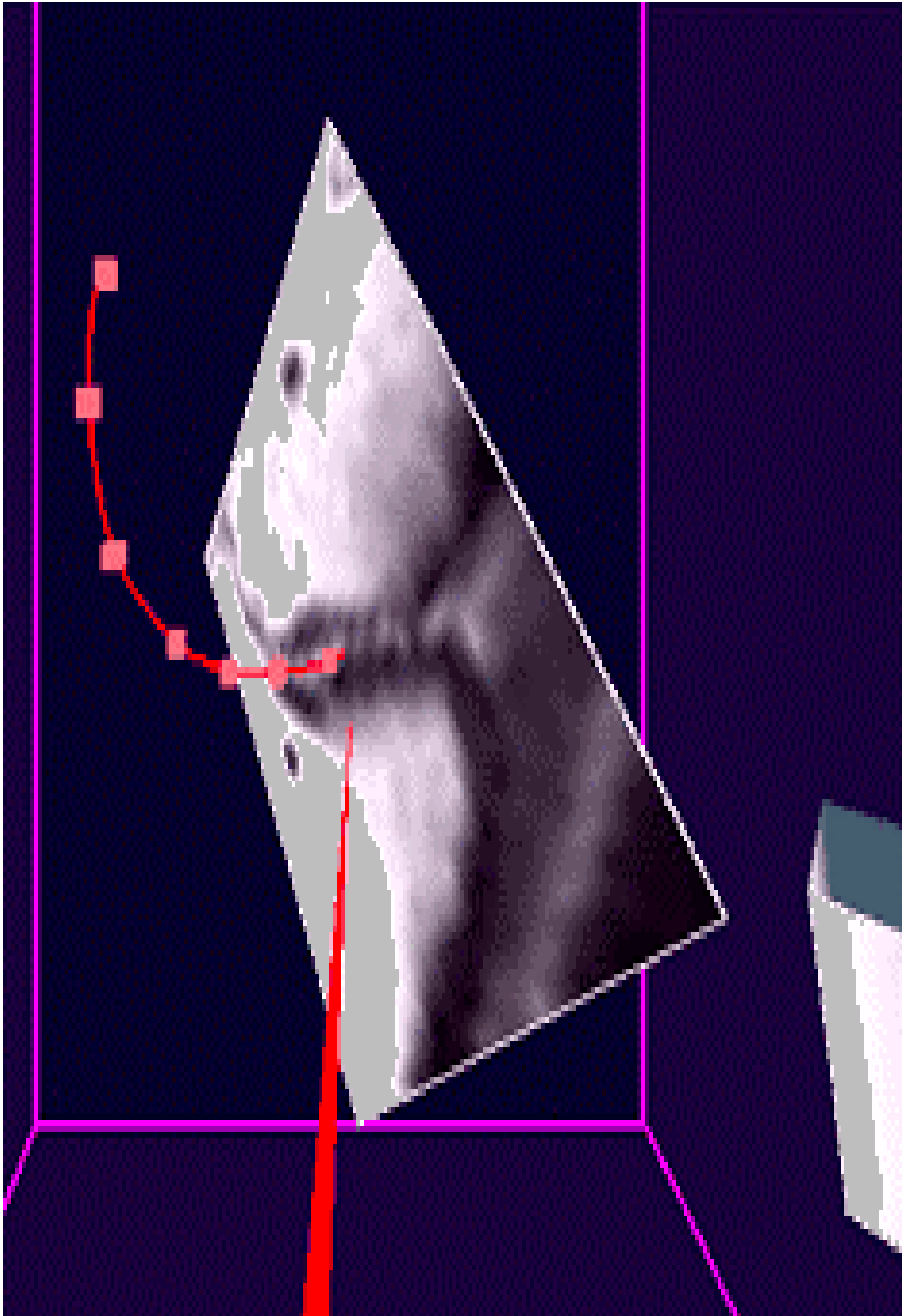
10 References 133

List of Figures

1: Methodology Modification Cycle.....	5
2: Formative Evaluation Cycle.....	18
3: VE Framework of Usability Characteristics Overview.....	20
4: In-depth Representation of “Virtual Model” Section of Framework.....	21
5: Initial Usability Engineering Methodology.....	35
6: Modified Evaluation Method.....	39
7: Crumbs Fiber Tracing.....	40
8: Vertebrae Dataset.....	49
9: Sperm Tail Dataset.....	50
10: Pilot Study Physical Setup.....	52
11: Main Evaluation Physical Setup.....	54
12: Colormap Object.....	73
13: Color Map Sub-menu.....	74

List of Tables

1: Usability Method Categories.....	11
2: Guidelines-Based Usability Issues and Redesign Suggestions.....	43
3: Participant Experience Comparison.....	48
4: Menu Benchmark Tasks.....	56
5: Audio Feedback Benchmark Tasks.....	58
6: Local Awareness Benchmark Tasks.....	58
7: Removing Objects Benchmark Tasks.....	61
8: Crumb Modification Benchmark Task.....	70
9: Colormap Awareness Benchmark Task.....	71
10: Load Colormap Benchmark Task.....	74
11: Opacity Object Benchmark Task.....	75
12: Usability Issues Unique to Guidelines-Based Inspection.....	81
13: Usability Issues Unique to the Formative Evaluation.....	84
14: Usability Issues Marked for Further Evaluation.....	86



1 Acknowledgements

We would like to thank our colleagues at Virginia Tech and NCSA for their continued support of ongoing HCI and VE research. In particular members of the VT-CAVE office: Dr. Ronald Kriz, John Kelso, Valdis Kletnieks, Kevin Curry, and Joseph Vandyke for software and hardware support, NCSA CAVE office: Stuart Levy, Cami Tiska, George Estes, and Jeff Carpenter, and Rachael Brady for allowing us to evaluate Crumbs. We would also like to thank Joseph Gabbard for his continuing support and insight regarding this research.

2 Introduction

2.1 Motivation and Problem Statement

HCI researchers have preached long and hard about the need to integrate user-centered methods in developing and evaluating interactive computer applications. Software engineers have witnessed the benefits of applying an iterative approach to software development with usability evaluations occurring early and often in the process. Therefore, software developers demand efficient usability engineering methods that are (1) applicable to a user-centered design approach and (2) appropriate for the specific application. However, evolution of usability engineering methods has been slow in comparison to industry needs. When industries transition to a new “hot” topic, HCI researchers are often still focusing their attention on older technology. This cycle leads to sustained periods of time when industry must attempt to produce usable applications without the assistance of appropriate user-centered design methods. Although, it is unrealistic to require HCI research into new technologies to precede industry demand, the HCI community needs a sense of urgency to provide software developers with appropriate methods to construct usable systems in emerging technological advancements in a timely fashion.

The recent increase in virtual environment (VE) research and development demonstrates the trend of computer industry need arriving much sooner than accessible and cost-effective usability engineering methodologies. Recently the amount of resources dedicated to VE research and development has been increasing rapidly. Government agencies, universities, commercial organizations, and the entertainment industry have all shown an increased interest in VE technology. Each organization is eager to leverage the enthusiasm currently surrounding VEs. Once the “hype” surrounding new interactive VEs fades, users are usually faced with the frustration of interacting with a complex system that appears to have little regard for usability issues. Most of the blame for development of these systems can logically be placed on software engineers eager to push the “wow” factor of the technology without examining usability tradeoffs. However, examining current state-of-the-art usability engineering methods exposes another possible contributor to the production of largely unusable VEs, namely

the lack of appropriate usability engineering methodologies. There currently exists no standardized - or for that matter even popular - methodology for conducting usability engineering that effectively addresses VEs. There is a growing awareness among both VE and HCI researchers of the shortcomings of traditional usability engineering methods in addressing the unique criteria of VEs. Although research efforts have been conducted in an attempt to rectify the situation, the breadth of aspects unique to VEs causes this research to be seemingly slow and incremental in nature. The research conducted as a portion of this thesis was among the first to identify VE-specific limitations on traditional usability engineering methods and to propose more effective techniques specific to VEs.

2.2 Objective of Research

The objective of this research was not only to increase awareness of the limitations of traditional usability engineering methodologies for designing and evaluating VEs, but also to initiate the process of modifying these methodologies to facilitate VE-specific usability engineering. It was not our objective to create a methodology that provided exhaustive coverage of VE-specific issues inherent in all types of VEs, but rather to propose a usability engineering methodology that begins to incorporate VE-specific issues into its strategy.

2.3 Approach to Research

In our initial attempt to create VE-specific usability engineering methods, we modified existing usability engineering methods to support usability characteristics that are unique to VEs. By modifying existing methods, we hoped not only to provide usability assessment methods that build upon previous HCI research, but also possibly to discover if appropriate modification is not feasible. If so, we will need to alter our approach to creating new (rather than modifying existing) usability engineering methods to accommodate design and evaluation of VEs. This research was a single portion of an on-going, multi-year project that will ultimately result in powerful and efficient VE-specific usability engineering methods. Our approach to assessing the limitations of existing usability engineering methods and making appropriate VE-specific modifications was three-stage:

- Choose existing usability engineering methodologies.

- Tailor the chosen usability engineering methodologies to facilitate VE usage.
- Conduct an evaluation using the modified engineering methodologies.

First, we chose an existing usability engineering methodology or combination of methodologies. We determined some VE-specific limitations that currently exist within the traditional method, including appropriate consideration of unique VE-specific characteristics and instability of VE interaction styles. This was accomplished by validating each method using the *Framework of Usability Characteristics of VEs* (Gabbard and Hix 1998) to ensure that each methodology provides appropriate coverage of VE characteristics. We also incorporated our own experience in usability engineering and VE knowledge to uncover possible VE-specific limitations. The result of the first stage was a list of characteristics unique to VEs that current methods are not capable of assessing. This phase is discussed in detail in Sections 4.1 and 4.2.

Next we modified the usability engineering methods according to the limitation list created in the previous step. This step required experience and research in both HCI and VE. Appropriate accommodations were found in HCI research, VE research, or developed from scratch to address each of the limitations. The result of this stage was a proposed VE-specific usability engineering method. This phase is discussed in detail in Section 4.3.

Finally we conducted a usability evaluation using the proposed usability engineering method developed in the previous step to assess whether the modifications appropriately addressed and resolved the limitations reported in step one. Specifically, we conducted a usability evaluation on Crumbs, an immersive medical visualization application developed at NCSA. The evaluation methodology was two-fold, first with a usability inspection occurring at Virginia Tech and then a user-inclusive empirical evaluation performed at the University of Illinois at Champaign-Urbana. The evaluation resulted not only in a usability issue report for Crumbs that facilitated usability improvements, but also provided valuable experience in conducting VE usability assessments that ultimately led to the discovery of new VE-specific limitations that were not originally hypothesized. This experience and new knowledge was then used to

determine the existing limitations that were still present in the proposed usability engineering methodology. This phase is discussed in detail in Chapter 5.

Performing this three-phase process was an iterative process, where each cycle built on the research of previous cycles. This iterative approach is similar to the current trend in software and usability engineering. Each cycle either results in a modified VE-specific usability engineering method or concludes that appropriate modifications do not exist to allow the method to assess completely the usability of interaction characteristics unique to VEs.

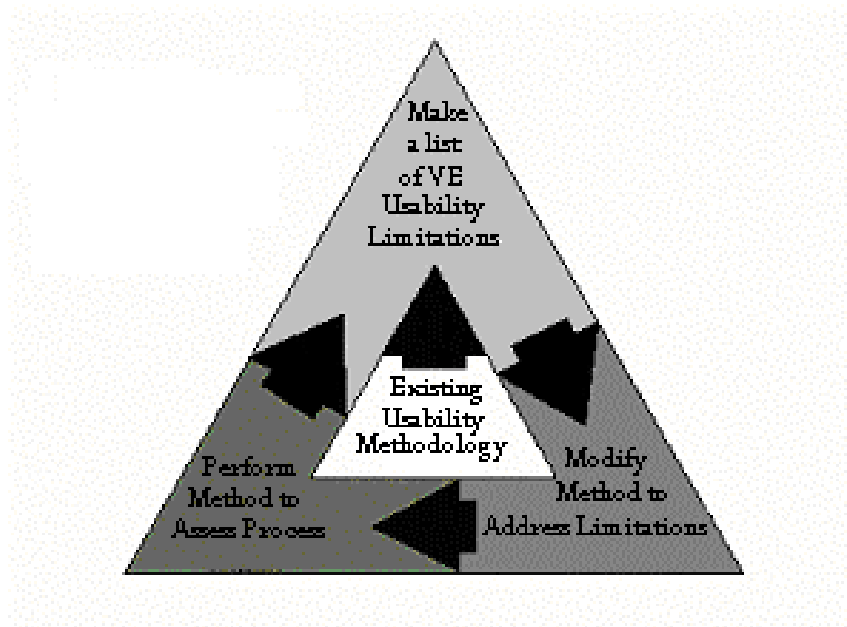


Figure 1: Methodology Modification Cycle

2.4 Summary of Contributions

This thesis makes several contributions to a multi-year project already underway at Virginia Tech. The first is to use the *Framework of Usability Characteristics of VEs* (Gabbard and Hix 1998) to assess several traditional usability engineering methods and determine, according to that framework, what modifications are necessary. Another contribution is development of a usability engineering methodology composed of traditional usability engineering methods altered to include VE usability characteristics present in the framework, but currently absent from the methods. We do not claim that the new methodology is a finished product, but rather a work-in-progress or first step

toward discovering usability engineering methodologies that are appropriate and efficient for use in the emerging VE field.

2.5 Limitations of Traditional Usability Methods

Numerous methods are set forth in HCI literature facilitating usability engineering principles in the design and evaluation of interactive systems. However, these methods possess well-documented shortcomings and limitations in the interactive arena they were created to design and evaluate. These pre-existing limitations only increase when attempting to use these traditional methods to create VEs. Most usability engineering methods were developed to design and evaluate either command-line or traditional graphical user interface (GUI) systems. Therefore, they were developed to discover efficiently and effectively those usability issues that are inherent in these environments. In the past, no research was conducted to facilitate the use of advanced and/or innovative interaction techniques, and therefore these methods fail to address issues unique to such interaction techniques. This section details some of the VE-specific limitations facing most traditional usability engineering methods including:

- Lack of a standardized interaction style.
- Users' lack of experience with VE interaction metaphors, techniques, and devices.
- Failure to capture metrics on VE-specific features.

A key limitation of traditional usability methods was the stability of the technology at the time of the methods' creations. Most usability methods were created after the industry had already settled on a standardized interaction style. Hix and Hartson (1993) define interaction styles as a "collection of interface objects and associated techniques from which an interaction designer can choose when designing the user interaction component of an interface." One well-known interaction style is the GUI window, icon, menu, pointer (WIMP) paradigm. VEs suffer from a lack of such a dominant and well-known interaction style. The VE community is currently in a constant state of flux as evolving applications often use heretofore unique interaction techniques. Although this situation parallels early GUI interaction development, prior to acceptance of the WIMP paradigm, the VE community is still nowhere near conforming to a single interaction style, and probably will not be in the near future. Due to the innovative and

complex nature of VEs, it is not presently clear that limiting VE development to standardized interaction styles is even desirable. In fact, the current trend in the VE community is to focus research on VE interaction styles to categorize specific styles for use in specific tasks (Bowman 1998; Gabbard 1998; Hix and Gabbard 1999). Traditional usability engineering methods were built to incorporate and assess standardized interaction styles and do not provide the flexibility to handle non-routine interfaces such as the ones that are commonplace in most VEs. Furthermore, traditional usability engineering methods often assume that participants in the usability assessment are familiar with the interaction style. There is a “common knowledge” of interaction techniques within the standardized interaction style. Therefore, the methods can focus on how well the application facilitates usable interactive strategies, given the constraints associated with familiar input devices and the WIMP paradigm.

To incorporate end users into usability evaluations of VEs, more time often must be spent on introducing users to the unique interaction metaphors used within a specific VE. Most people know how to interact with a GUI application using a keyboard and/or mouse, but devices such as a “wand” or “spaceball” are foreign to most users, and how developers have coupled these devices with an interaction metaphor is even more foreign. Therefore, the idea of a system supporting complex interaction usable by most people without training is relatively unaccepted in the VE community. It is a well-documented fact that user-inclusive usability assessments are expensive, but nonetheless necessary to ensure usability of interactive applications. Large amounts of resources are required to collect data from sets of sample users and perform task-based evaluations. VE usability specialists ordinarily do not have the luxury of assuming that all participants will have prior VE experience or experience using any of the information presentation metaphors or interaction metaphors leveraged by the VE. Therefore additional effort must be focused in the development of user-inclusive evaluation techniques to ensure that the evaluation of the domain task-related aspects of a VE is not overshadowed by an inappropriate interaction or information presentation metaphor.

Traditional usability engineering methods are primarily designed to focus on collecting traditional metrics on a single-user application performing low level tasks. This design is beneficial to the conventional “passive” GUI/WIMP paradigm which

assumes that the primary driver of action in the environment is unambiguous with well-defined user actions (e.g., move mouse to specific location and click mouse). However, VEs tend to be more dynamic and active in nature. Entities such as agents, objects, or other collaborators often have the ability to operate and alter the environment independent of any user input. Users are oftentimes required to perform more complex tasks such as sharing and multi-threading. Existing usability engineering methods do not attempt to assess multi-user interactive systems that support collaborative tasks.

“Conventional” evaluation techniques fail to capture some of the critical features that characterize interaction with the new generation of user interfaces created by VEs. The techniques also fail to address application context-dependent characteristics that are important to the usability of VEs, such as situational awareness and perceived presence. In particular, these conventional techniques do not possess any strategy for collecting and analyzing quantitative or qualitative data on such characteristics.

Finally, VE users often are unable to give detailed subjective (e.g., verbal protocol) data concerning new interaction techniques common in VEs. This may be due, in part, to the increase in complexity associated with interacting in VEs. Often, users are not sure if their inability to complete a task is due to their misunderstanding of an interface metaphor or whether the application has usability issues associated with the domain aspects. Confused by the source of the problem, users often resort to a simple communication of frustration without elaboration. Again the familiarity with the WIMP metaphor for GUI applications helps to eliminate some of this confusion and allows the users to elaborate on what they believe is the cause of the problem. This inability to verbalize their opinions accurately concerning possible usability issues in VE applications has profound implications for user-centered design. Iterative redesign is greatly impaired if the participants are unable to express what they believe to be the strengths and weaknesses of the interaction. Traditional methods give minimal advice or strategy for dealing with participants that find it difficult to talk about and communicate the problems they are having with a system.

Because of these limitations, use of existing usability engineering methods for user-centered design and evaluation of VE user interfaces requires a thorough assessment of each method to uncover what VE-specific aspects it lacks and direct modification of

the method to compensate for those shortcomings. The comprehensive framework of usability characteristics specifically provides a structure upon which new research into method development and modification can be based.

2.6 Document Organization

This thesis is organized as follows. Chapter 3 describes related research on which the thesis is built. It describes some existing usability engineering methods, a set of guidelines for VE development, and a method of conducting qualitative research such as we have performed. This chapter does not exhaustively list every usability assessment or qualitative research method, but it does present research that is relevant to the thesis. Chapter 4 describes modifications we made to existing usability assessment methods using the methodology modification cycle and demonstrates a complete cycle in the methodology modification cycle. It focuses on two specific existing usability assessment methods. First, the two methods are examined to determine possible existing limitations to VE usability assessment. Next, Chapter 4 describes a usability assessment method designed to remove some of the limitations discovered. Chapter 5 describes application of the revised method on a specific VE called Crumbs. Chapters 6 and 7 discuss the second iteration of the methodology modification cycle which is a continuation of the cycle started in Chapter 4. Chapter 8 draws conclusions and discusses the contributions this thesis can have on future VE-specific usability assessment methodology research.

3 Related Work

This chapter introduces several existing techniques for conducting usability assessments, a comprehensive framework of VE-specific usability characteristics, a technique for collecting and analyzing qualitative data, and a review of some current research in the VE usability engineering field. Research introduced in this section includes, (1) proven usability engineering methodologies in the HCI field, (2) new VR-specific usability research, and (3) proven interviewing techniques used in behavioral and social sciences. Merging this existing research provided the foundation for our VE-specific usability assessment methodology. Therefore a solid understanding of the research introduced in this chapter is imperative to the remainder of our research.

3.1 Usability Engineering Methodologies

Iterative performance of usability engineering methods is crucial to development of any interactive application. Application development void of a usability strategy often results in function-rich software riddled with usability issues that construct a barrier between users and application functionality. However, simply supplementing software engineering by appending a usability assessment at the conclusion of development is also insufficient. If usability assessment does not occur early and frequently in the development lifecycle, unnecessary resources have already been spent before the usability issues are discovered. This conflict places developers in a precarious and no-win position of deciding whether to allow known usability issues to remain in the application or redesign large sections of the already completed application interaction. Therefore, user-centered interaction design iteratively applies usability engineering methods that have been researched and proven successful in the HCI community. Since user-centered design gained acceptance in the computer science community, there have been a number of methodologies proposed and researched to assist in assessing the usability of interactive applications. Deciding which usability methodology to use is based on objectives, resources, and time constraints of the evaluation, but as Nielsen (Nielsen and Mack 1994) notes, the ultimate trade-off is possibly between doing some usability assessment and none at all.

Although there are at least four basic categories of usability engineering methods -- automatic, empirical, formal, and inspection -- only two of these categories have gained widespread use. [Table 1](#) lists four of the most common categories of usability engineering methods and a brief description of each.

Usability Method Category	Description
Automated	User interaction is described using an interface specification technique. The interface specification is then validated using a software application designed to assess usability of the interface described by the specification technique.
Formalism-based	User interaction is represented using a set of formal interaction models and then a set of usability functions is applied to the models.
Inspection	Evaluations are based on a set of user interaction design guidelines and/or heuristics and relies on judgment and experience of evaluators.
Empirical	Evaluations are completed using a representative sample of application users and a pre-determined set of representative tasks.

Table 1: Usability Method Categories

3.1.1 Automated and Formalism-based Methods

An automatic usability method involves validating an interface specification using evaluation software. There are two problems with this type of evaluation method. The first is that automatic methods require creation of evaluation software specifically to assess a certain interaction style and using a specific interface specification. Currently, there is no standard interface specifications, therefore requiring modifications to interface specifications to facilitate the evaluation of new interaction styles is time-consuming. Secondly, according to (Nielsen and Mack 1994), “with the current state of the art, automatic methods do not work.” Nearly a decade later, this is still largely the case. Calculating usability measures using formal usability engineering methods require the use of interaction models and usability formulas. These methods require a substantial amount of model and formula knowledge for the evaluator in order to conduct an evaluation properly. Furthermore, validated models and formulas are not available for

most interaction styles. Therefore formalism-based evaluations are extremely time-consuming and comparably hard to perform in the evaluation of highly interactive systems.

Two currently more popular methods used in usability assessments are usability inspections and empirical evaluations. The two differ in methodology, but both strive for the common goal of increasing application usability.

3.1.2 Usability Inspections

“Usability inspection is the generic name for a set of methods based on having evaluators inspect or examine usability-related aspects of a user interface” (Nielsen and Mack 1994). Nielsen and Mack begin their executive summary of a book dedicated to usability inspection methods with this simple but effective definition. Inspections assess user interface designs based on judgmental opinions of inspectors, without including representative users. They are designed to uncover, for example, usability violations in generic methods used for interaction between users and applications. They are less effective at addressing domain-specific areas of usability and therefore cannot uncover all existing usability issues. One reason for this restriction is the shortage of usability specialists with expert domain knowledge. Another reason is that inspection techniques are usually based on a predefined set of guidelines or questions to evaluate a wide variety of applications. However, usability inspections are good at locating a large number of usability issues at a fraction of the cost of alternate usability assessment techniques, especially when used very early in the design process. Traditionally, the results of a usability inspection are a list of usability issues discovered in the application and recommendations for interaction redesign based on these issues.

Usability inspections are effective only within a certain time frame in the software/usability engineering cycle. Usability inspections are not useful during the original requirements and design phases of software engineering, when there are no interaction models or designs to evaluate. During these phases, usability engineering methods such as user analysis, task analysis, needs analysis and use case development are more appropriate development methods. Inspections start to play a role soon after

development of the initial prototype in a lifecycle that incorporates rapid prototyping. Inspections are also not effective late in the lifecycle when the software is either released or ready to be released. At this point so much time, effort, and money have been put into application development that interaction redesign issues tend to be too expensive to correct. Therefore, late in the development lifecycle, beta testing and client suggestions steer development of the next release.

As previously mentioned, usability inspection is a meta-level categorization of a specific type of usability assessment method. There are currently a number of usability evaluation methods that fall under this classification, each with its own unique assessment strategy and many of which have evolved from previous inspection methods. The following two sections give an overview of two different usability inspection techniques, each playing a significant role in our research.

3.1.2.1 Guidelines-Based Evaluation

Guidelines-based evaluations are a type of usability inspection where an application interaction design is validated against a comprehensive list of usability and user interaction design guidelines. The definition of a guideline is sketchy at best but can be described as a tested principle, ground rule, or rule of thumb for design of the interface (Cuomo 1992). For an example of a comprehensive (albeit archaic) set of guidelines largely for command-line interfaces, see “Guidelines for Designing User Interface Software” (Smith and Moser 1986). Because usability guidelines target a wide range of interactive systems, guidelines are general in content. Another type of inspection method that compares closely to guidelines-based evaluation is a standards evaluation. Standards evaluations incorporate the assistance of a usability specialist with expertise in a particular commercial style guide (e.g. Motif) to assess how well an interaction design complies with the guide.

It is not uncommon for guideline documents and style guides to contain around 1,000 different guidelines. Because these documents contain such a comprehensive coverage of usability issues, they require a lot of experience and effort to incorporate into a usability engineering lifecycle. These guidelines do not provide situations or contexts on when and where to use the guidelines and therefore inexperienced evaluators are left

attempting to evaluate an interface with each guideline, even if the guideline does not apply to the current application.

3.1.2.2 Heuristic Evaluation

Another popular method of usability inspection, heuristic evaluation, was introduced by (Nielsen and Molich 1990). Heuristic evaluation is the most informal technique of usability inspections and was ultimately motivated by a “discount usability” perspective to make usability engineering methods more cost-effective. Therefore a premium is placed on usability engineering methods that are both economical and effective. The heuristic evaluation technique attempts to provide a usability engineering methodology that is efficient enough to apply within reasonable time and resource constraints.

Heuristic evaluation is a usability assessment method in which one or a group of usability specialists evaluate a particular user interaction design in depth to determine if it conforms to an established set of usability design guidelines. Nielsen (1994) conducted a factor analysis on 249 reported usability issues and created the following set of ten guidelines referred to as “heuristics”:

- **Visibility of system status** - The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- **Match between system and the real world** - The system should speak the users’ language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- **User control and freedom** - Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
- **Consistency and standards** - Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- **Error prevention** - Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

- **Recognition rather than recall** - Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- **Flexibility and efficiency of use** - Accelerators—unseen by the novice user—may often speed up the interaction for the expert user to such an extent that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- **Aesthetic and minimalist design** - Dialogues should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
- **Help users recognize, diagnose, and recover from errors** - Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- **Help and documentation** - Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

This set of heuristics is the standard set often used by HCI evaluators to perform a heuristic evaluation.

One characteristic of heuristic evaluation is the free-form method evaluators use to assess the interaction design. Evaluators are given the list of ten heuristics and a brief description of the user interface. They are then instructed to explore any part of the interface. This open-ended strategy of evaluation is designed to facilitate the discovery of usability issues that might be overlooked if evaluators were confined to following a set of pre-defined representative user tasks. Nielsen (1994) does introduce the addition of usage scenarios in the event that evaluators do not have domain-specific knowledge to allow them to use the interface. Addition of usage scenarios introduces a two-pass heuristic evaluation. The first pass follows usage scenarios and introduces evaluators to representative tasks and interaction techniques. The second pass allows evaluators more

freedom to create their own tasks and evaluate other parts of the interaction not covered in the usage scenarios.

The usability specialists report any discovered violations and apply a user-centered perspective to rate the severity of discovered usability issues. Violation reports include individual heuristics that the interaction design violates. It is not enough simply to report a usability issue; evaluators must be descriptive enough to explain the situation that led to the problem, and record the specific heuristic violations that caused the problem. These reports are then discussed with the development team and redesign decisions are made based on severity of each usability issue and the estimate of resources required for redesign.

One of the common complaints HCI researches have concerning heuristic evaluation is the lack of guidance in method instructions. Effective performance of a heuristic evaluation appears to rely heavily on evaluator experience with the method. The main advantage of heuristic evaluation - its simplicity to understand and apply - ironically appears also to pose the possibility of being a disadvantage. The only specified guidance provided to evaluators is the set of heuristics and an “understanding” of the interaction. This lack of guidance can ultimately lead to over-simplified evaluation that does not focus on the interaction that will occur most frequently when completing common tasks. Also, many usability issues may be overlooked if they cannot be categorized as violating one of the specific heuristics. In contrast, blindly applying the heuristics to every situation can produce usability issue reports filled with problems having little impact on system usability.

3.1.3 Empirical Usability Evaluations

Empirical evaluations are one of the most popular methods of usability assessment. Empirical evaluations involve usability specialists observing as representative participants perform a predetermined set of typical tasks while interacting with an application. The evaluation process is usually recorded either by video and/or written account. This historical account of the evaluation provides valuable information regarding the participant’s confusions, errors, complaints, and other experiences. Two commonly measured types of data are time to task completion and errors that are made.

User-based evaluations, in comparison to usability inspections, do focus on domain-specific aspects of an application. Hix and Hartson (1994) state that representative participants and usability experts are required to examine the effects an interface has on user performance. In most cases, participants should possess domain knowledge and therefore be able to evaluate these specific aspects of the application. However, incorporating users into an evaluation introduces a number of possible problems. Empirical evaluations inherently are more expensive in comparison to usability inspections and recruiting a representative sample of users can be difficult.

As previously stated, empirical evaluation is a meta-level categorization of a specific type of usability assessment method. There are a number of empirical evaluation methods that fall under this classification, each with its own particular traits. The following section gives an overview of one particularly significant empirical technique.

3.1.3.1 Formative Evaluation

Formative evaluation is a form of empirical evaluation where usability assessment involving user observation happens early and often in the usability engineering lifecycle (Hix and Hartson 1993). The main purpose of formative evaluations is to improve usability continually through iterative user observational studies. The iterative nature of formative evaluation assists the application development team in uncovering usability issues early and determining a plan for redesign. Compared to alternate empirical evaluation techniques such as summative evaluation, formative evaluation requires fewer participants and less time. This frugal use of resources allows formative evaluation to be economical enough to include in a user-centered iterative design strategy. Formative evaluations are often viewed with the misconception of resulting in minimal data taken on few participants. However, experienced usability specialists can collect enormous amounts of numerical-based quantitative data, narrative-based qualitative data, directly observed objective data, and opinionated subjective data using this technique. Although the data are usually not analyzed through a process resulting in statistically significant results, they do provide quantitative results that show whether usability issues exist and qualitative results that indicate where and why usability issues will occur most likely. Figure 2 demonstrates the steps and iterative nature of formative evaluation.

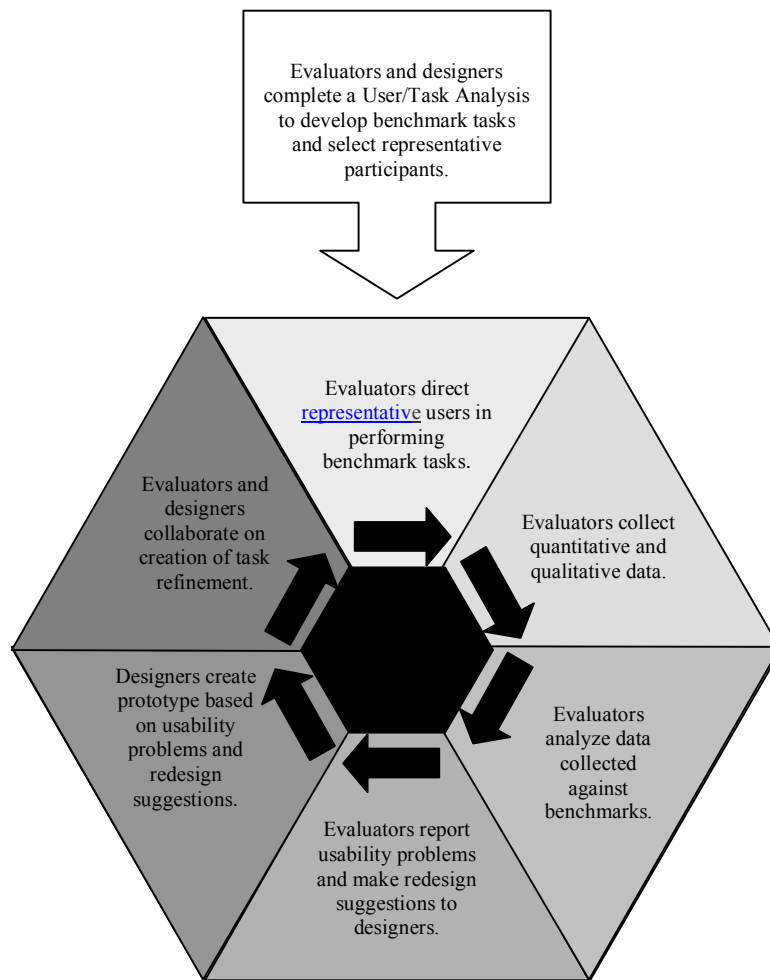


Figure 2: Formative Evaluation Cycle

3.2 VE Framework of Usability Characteristics

The framework of usability characteristics (Gabbard 1998) provides several VE-related usability resources including a comprehensive set of usability guidelines, detailed discussion of the guidelines, and assistance in locating auxiliary references. The guidelines are an organized presentation of multiple VE research efforts gathered from various sources, including but not limited to: investigative research visits to some top VE research facilities, VE-related journals and conferences, human-computer interaction related literature, and World Wide Web internet-based searches for related work. The

framework is organized into four main sections each associated with a specific component of VE interaction:

- *VE Users and User Tasks* – general user and task characteristics and types of tasks in VEs.
- *VE User Interface Input Mechanics* – usability characteristics of VE input devices.
- *Virtual Model* – usability characteristics of generic components typically found in VEs.
- *VE UI Presentation Components* – usability characteristics of VE output devices.

This organization was influenced by Norman's *theory of action* (Norman 1986). Norman's theory is an "approximate theory of action which distinguishes among different stages of activities, not necessarily always used nor applied in that order, but different kinds of activities that appear to capture the critical aspects of doing things" (Norman 1986). In particular, Norman concentrated his theory on the interaction between a person and a computer. Computer system users develop tasks conceptually. Computer systems provide physical devices and objects to control system state.

Transition between the conceptual and physical creates what Norman calls "gulfs" between the system and its users. Usability is determined by how well the "gulfs" are bridged. Norman introduces two "gulfs" in his theory: the *gulf of execution*, and the *gulf of evaluation*. The *gulf of execution* exists whenever users must translate their goals into the required physical actions to manipulate correctly the system's physical variables. Although Norman believes this gulf can be partially bridged with user training and experience, he places a large load of the responsibility on interaction designers saying, "the choice of input device can often make an important impact on the usability of the system" (Norman 1986). The *gulf of evaluation* is the ability of users to assess the state of the system and compare it to their goals. One of the important issues in bridging this gulf is the ability to interpret the system state. Norman believes this task can be greatly impacted by interaction designers in providing "appropriate output displays by the system itself" (Norman 1986). The *VE Users and User Tasks* and *Virtual Model* sections of the *Framework*, respectively, represent users and computers in Norman's theory. The *VE*

User Interface Input Mechanisms section represents the “interface mechanisms” section of the bridge Norman uses to reduce the *gulf of execution*. Norman also introduces an “interface display” section of the bridge used to reduce the *gulf of evaluation* that is represented in the *Framework* by the *VE UI Presentation Components* section.

Each main *Framework* section, represented by the four shaded boxes in Figure 3, contains VE design guidelines and in-depth discussion of topics relevant to the specific section. Each white box that is attached to a main section box in Figure 3 represents a sub-section. Refinement of main sections further assists VE developers and evaluators, providing scaffolding to guide their efforts.

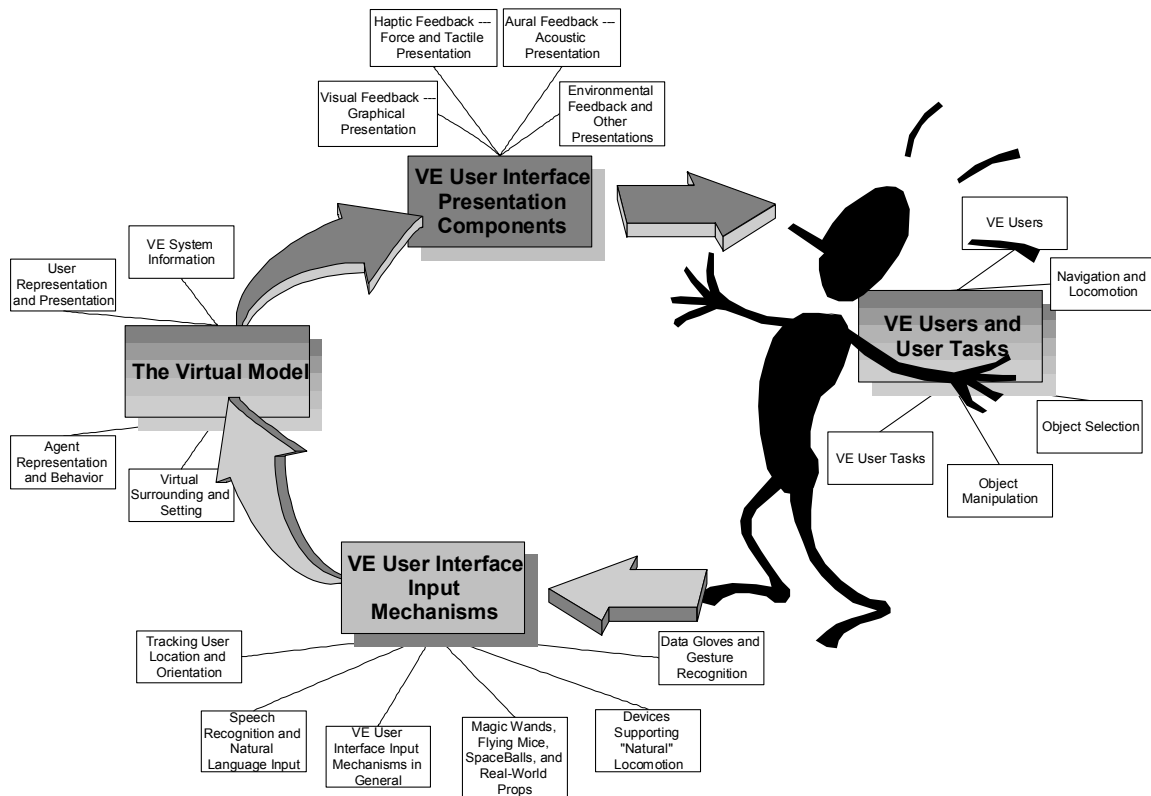


Figure 3: VE Framework of Usability Characteristics Overview (used with permission from Gabbard, 1998)

Figure 3 is a representation of extensive coverage of VE-specific issues provided by the *Framework*. Figure 4 gives a detailed look at the *Virtual Model* main section complete with sub-sections and guidelines. The *Framework* provides detailed discussion

of each of the guidelines presented, as well as pointers to reference material for further discussion. There are 195 total guidelines across all sections.

The Virtual Model	
VE System Information 1. Use progressive disclosure for information-rich interfaces 2. Pay close attention to the visual, aural, and haptic organization of presentation 3. Strive to maintain interface consistency across applications 4. Language and labeling for commands should clearly and concisely reflect meaning 5. System messages should be worded in a clear, constructive manner so as to encourage user engagement 6. For large environments, include a navigational grid and/or a navigational map 7. When implementing maps, adhere to map design principles 8. Present domain-specific data in a clear, unobtrusive manner such that the information is tightly coupled to the environment and vice-versa 9. Strive for unique, powerful presentation of application-specific data, providing insight not possible through other presentation means	User Representation and Presentation 1. For collaborative VEs, design avatars to convey user viewpoint and activity 2. Ensure that users' avatars provide a familiar, accurate, and relevant frame of reference 3. Provide egocentric point of view(s) when users need to experience a strong sense of self-presence 4. Provide exocentric view(s) when relative positioning and motion between user and objects are important 5. User embodiments should be as efficient as possible 6. Allow users to control presentation of both themselves and others 7. Allow users to alter point of view, or viewpoint
Agent Representation and Behavior 1. Include agents that are relevant to user tasks and goals 2. Real-world, high-fidelity physical and behavioral agent representation may be useful for training and simulation VEs 3. Allow agent behavior to adapt dynamically, depending upon context, user activity, etc. 4. Represent interactions among agents and users (rules of engagement) in a semantically consistent, easily visualized manner 5. Organize multiple agents according to user tasks and goals	Virtual Surrounding and Setting 1. Use setting to increase presence 2. Exploit real-world experience, by mapping desired functionality to everyday items 3. Use relevant settings that suggest user activity and tasks 4. Employ rendering techniques that support detailed presentation of setting without introducing lag

Figure 4: In-depth Representation of “Virtual Model” Section of Framework

The strength of the *Framework* is not in categorizing VE-specific issues according to Norman's theory, but in its extensive coverage of VE-specific usability issues. Therefore one obvious use of the *Framework* is as an instructional tutorial and design guide to VE designers concerning VE-specific usability issues. The *Framework* also can be used by HCI researchers in gathering material related to a certain VE-specific issue for further research. Another potential usage for the *Framework* and the reason for its relevance to this thesis is that it provides a foundation for modification of existing usability assessment methods to include VE-specific issues and also for creation of new VE-specific usability assessment methods.

3.3 Concept Book Approach to Content Analysis

Content analysis is simply the act of restructuring fractured communication for the purpose of making sense out of it or using our own experience and judgment to develop conclusions concerning unstructured communication. This is a familiar task that we perform every time we communicate socially with others or extract news items from the media. Berelson (1971) compares content analysis to “close reading plus judgment.” Due to the ambiguity of words, if we do not use content analysis and choose to draw conclusions prior to analyzing communication based entirely on grammatical “face value”, then we may miss a substantial portion of communication such as non-verbal cues.

Content analysis is particularly important to researchers attempting to collect and analyze qualitative data such as usability specialists drawing conclusions from user questionnaires and interviews. Content analysis is a measurement device used by researchers who must analyze large amounts of qualitative data. According to Mostyn (1985), qualitative researchers are more interested in “why” and “how” a person experiences a certain phenomenon rather than “what” was experienced. Mostyn (1985) builds on this idea when he states that the major conceptual difference separating quantitative and qualitative data is that the former is collected when the researcher is interested in “what, where, when and how many” and the latter is collected when the researcher wants to know “why”. Therefore it is insufficient for the qualitative researcher to collect concrete quantitative data and draw conclusions based on data analysis. There must be a method of collecting qualitative data matched with a method of analyzing such data. There are a number of proven methods available for collecting qualitative data, such as concurrent verbal protocol, retrospective verbal protocol, critical incidents, and structured interviews (Hix and Hartson 1993).

Qualitative researchers are not able to read minds and must deal merely with what respondents can communicate; therefore raw qualitative data consists mostly of words and gestures. Analysts attempt to find relationships between the semantics and syntax that exist within the data. However, because content analysis is concerned with

discovering both the conspicuous and hidden meaning of communication, qualitative researchers are not afforded the luxury of simply locating meaningful relationships in the data (Mostyn 1985). Therefore application of content analysis of qualitative data requires researchers either to have previous experience with respondents' behavior, motivations, and roles or to gather this information via user and task analysis.

According to Hosti (1969), all content analysis approaches must adhere to three basic requirements:

- *Objectivity*: The evaluator must not influence the data due to a bias.
- *Systematic*: The design of the analysis must be based on collecting data relevant to the hypothesis.
- *Generality*: The sample used to collect data must be representative enough of the intended population to promote a reasonable generalization of the results.

These requirements also allude to basic necessities that must exist prior to conducting a content analysis. The *Objectivity* requirement relies on a researcher who is well-versed in the art of interviewing techniques. Developing a correct style of asking open-ended non-bias questions to extract necessary data is a skill accomplished only through proper training and experience. To meet the *Systematic* requirement, valid hypotheses must be created prior to performing data collection. This requirement again points to the necessity of experience prior to conducting qualitative research. The researcher must be experienced not only in qualitative collection and analysis techniques, but also must possess experience in his/her field to formulate and test valid hypotheses. *Generality* highlights the need for proper user analysis prior to conducting qualitative research. The researcher must have a representative population to validate results for the larger population.

The "concept book approach" is a method of conducting content analysis of qualitative data that was originally developed by Dr. Ernest Dichter and then later proposed by Mostyn (1985). It diagrams thirteen (13) steps for conducting content analysis of open-ended material. The steps are designed to ensure that the three basic requirements of content analysis given above are met during qualitative research. Each of the thirteen (13) steps is set forth below, accompanied by a description of the processes each step entails.

1. *Briefing* – Make sure the research problem is completely understood.
2. *Sampling* – Ensure that the sample is representative enough to generalize the findings to the larger target population.
3. *Associating* – Use personal experiences to aid in hypothesis creation. Is some previous work relevant to this research problem?
4. *Hypothesis Development* – Create a binder containing testable hypotheses on separate sheets of paper. Hypotheses can be created by way of *associating* or using an understanding of the dynamics of human behavior. During the study, order the hypotheses pages according to relevance with most relevant at the front and least relevant at the back.
5. *Hypothesis Testing* – Ask open-ended questions using the funnel approach, starting with a general question and then moving to specifics, to collect data regarding hypotheses during interviews. Researchers must be willing to relinquish initial hypotheses and create new ones according to participants' responses.
6. *Immersion* – Remain immersed in the data throughout interviews and analysis. Due to the potentially ambiguous nature of qualitative data, qualitative researchers must have access to interview events whether by written transcript or by audio or video recording. This allows researchers to review events exactly as they happened in order to uncover how and why something is occurring. Because communication includes more than semantics and syntax, it would be beneficial to record sessions for further review. Researchers must have the experience and ability to focus on key aspects of the interview without being sidetracked by irrelevant data. At this stage, two modifications to the concept book will occur. Modifications to original hypotheses will be made to mirror what is occurring in the data. Also, the data will uncover new hypotheses, which will be appended to the concept book.
7. *Categorization* – Create appropriate, exhaustive, and mutually exclusive categories to organize data in the concept book. Define a category label to represent each hypothesis and/or idea, link each category to the support or rebuttal of a hypothesis, and code raw qualitative data using these labels.

8. *Incubation* – For a refresher, re-read the concept book in its entirety and utilize several days to reflect upon the various ideas.
9. *Synthesis* – Attempt to locate patterns and relationships which could lead to a dominant concept. Revisit raw data coding and ensure that it makes sense.
10. *Culling* – Condense raw data in preparation of the final report. It is impractical to record every observation in the final report, therefore remove unsupported original hypotheses, non-sustained ideas formed in the *Immersion* and *Categorization* steps, and ideas that are confusing and contradictory.
11. *Interpretation* – Use intuition, creativity, education, and experience to interpret the data. Attempt to find meaning and implications in the data.
12. *Write* – Write up the report. Mostyn (1985) gives five important guidelines that qualitative researchers must bear in mind when writing up the report:
 - Be sure to include the *incidence of occurrence* when discussing a key concept or finding. In other words, be specific about how many and what demographics of respondents are involved.
 - All feelings expressed by respondents should be given a *direction and intensity*. Use adverbs such as “extremely”, “barely”, and “moderately” to describe the *intensity* of the feelings. Clearly state at what object the feeling is *directed*, or in other words what is responsible for the feeling.
 - Concentrate not only on what is said, but also on what is not said. Pay particular interest to pauses, stutters, slips of the tongue, and use of fillers such as “um” as these may provide as much insight as the ideas themselves.
 - Attempt to identify how *salient* respondent-stated attitudes are aligned with their behavior.
 - Respondents often must attempt rationally to verbalize responses concerning thoughts that might possess an irrational quality, for example, trying to communicate the occurrences of a dream. Attempt to discover the meaning of such responses.

13. *Rethink* – Review the research objectives and determine if they have been met. Make sure the interpretations conveyed in the report are supported by the data.

Usability specialists spend a substantial amount of time collecting, analyzing, and reporting qualitative data. Expertise in qualitative research methods often determines the effectiveness of the usability specialists. Modifying an existing method of qualitative data research used in a different discipline is an efficient strategy for effectively creating methods of dealing with qualitative data in usability engineering. This thesis proposes a usability assessment methodology that makes use of a multi-stepped approach to qualitative data similar to the concept book method.

3.4 Other Related Work

A substantial amount of VE research focuses on development and/or comparison of individual input and visual devices used to support VEs. Because this thesis concentrated on VE software issues, this section does not report on research that focuses entirely on VE physical devices. One piece of evidence which demonstrates the growing interest in developing VE usability methodologies was the First International Workshop on Usability Evaluation for Virtual Environments in the United Kingdom. The workshop focused on approaches currently used in evaluating VEs (Tromp, Hand, Kaur, Istance, and Steed 1998). This intentions of this workshop was to focus on:

- What are the specific constraints of VE evaluation.
- What are possible solutions for these constraints.
- What a suitable methods for VE evaluation.
- What are the results of past and current VE evaluation efforts.

We classified the VE research we examined into six categories of ongoing VE efforts:

- Creation of testbed VE applications designed to compare the usability of specific navigation, object selection, and object manipulation interaction techniques.
- Creation of VE design guidelines to assist developers in VE development.

- Researching objective and subjective methods of evaluating abstract VE concepts such as presence and situational awareness.
- Ad-hoc empirical evaluations performed on existing VEs.
- Empirical evaluations similar to the previous category, but altered in an attempt to retrieve VE-specific data.
- VE development that does not mention any usability evaluation. A majority of the current VE research falls into this category. Since such a large number of research efforts do not report any usability engineering efforts, we chose not to single out any one offender specifically.

3.4.1 Testbed Evaluation

Several current VE usability research efforts are developing testbeds to evaluate novel interaction techniques supporting navigation, object selection, and object manipulation. Bowman defines testbeds as “environments and tasks that involve all of the important aspects of a task, that test each component of a technique, that consider outside influences (factors other than the interaction technique) on performance, and that have multiple performance measures” (Bowman 1998). Poupyrev, Weghorst, Billingham, and Ichikawa define their Virtual Reality Manipulation Assessment Testbed (VRMAT) as, “a flexible, easy re-configurable, experimental tool which allows in-depth studies of immersive manipulation” (Poupyrev, *et al.* 1997). Although a main focus of VE testbed evaluation is to compare new interaction techniques, research efforts are also developing a taxonomy of VE interaction techniques to assist VE developers in determining which interaction technique is the most usable and efficient for specific tasks. Both efforts followed a traditional empirical evaluation procedure by using representative users. The main objective of the VRMAT evaluations was to “measure user performance, using some criteria, while they accomplish tasks” (Poupyrev, *et al.* 1997). Bowman’s research reportedly collected data on “quantitative measures such as speed and accuracy, HCI concerns such as ease of use and ease of learning, and more subjective metrics such as spatial awareness, presence and user comfort” (Bowman 1998). Although Bowman does include qualitative as well as quantitative measures, he does not reveal the process he used to collect such data.

An additional related piece of research was conducted in a joint effort by researchers at Virginia Tech and researchers at the Naval Research Laboratory in Washington DC (Hix, *et al.* 1999). The research presents a structured user-centered usability engineering approach to the design and evaluation of a battlefield visualization created at the Naval Research Laboratory Virtual Reality Lab. The research encompasses an iterative application of heuristic evaluation, formative evaluation, and summative evaluation. Although the research revealed several user-performed generic tasks such as object manipulation, object selection, object querying, query response, and object aggregation, the researchers chose to concentrate their efforts on navigation within the VE, because it “profoundly affects all other user tasks” (Hix, *et al.* 1999). The focus was on using the military battlefield visualization VE as a testbed for evaluating user navigation techniques. In this way it is somewhat similar to the testbeds created by the Bowman and Poupyrev research; however the former two testbeds were created for more generic assessment and comparison.

Evaluating and comparing individual interaction techniques is important to development of usability engineering methodologies when it results in a set of taxonomies providing valuable information concerning which interaction technique is most suited for a specific task. However, the evaluations are executed in a controlled testbed application rather than real world applications and are applicable in design guidance and assessment scaffolding. Therefore, these evaluations are not sufficiently thorough to form a complete VE-specific usability design methodology.

3.4.2 Researching Quantitative and Qualitative Measures for Presence

Another popular topic in VE research is the attempt to discover successful methods of providing quantitative and/or qualitative measures on abstract VE concepts such as presence and situational awareness. Although there is still an active debate within the VE community as to the exact correlation between presence, situational awareness, and task performance; most researchers agree that presence does play a role in the effectiveness of a VE. Therefore methods of measuring presence and situational awareness are important techniques for usability engineers.

Some of the research on presence focuses on validating the positive correlation between task performance and presence. Snow conducted research to “provide a ratio-scale measure of perceived presence in a VE, to explore the effects of a number of environmental parameters on this measure and construct empirical models of these effects, and to relate perceived presence to user performance” (Snow 1996). Snow conducted three empirical experiments in which he instructed participants to perform a set of tasks. He measured perceived presence using a technique known as free-modulus magnitude estimation. Although Snow did report a strong correlation between the manipulation of VE parameters and participants’ subjective feeling of presence, the correlation between perceived presence and task performance was reported as weak.

Boyd conducted empirical evaluations comparing an immersive VE with two non-immersive VEs (Boyd 1997). The evaluation tasks incorporated both navigation and searching. Boyd’s primary concern was to ascertain whether “immersion in a virtual environment increases usability when the task requires search and navigation, specifically egolocomotion (the act of moving the user’s viewpoint through the VE as if they themselves are moving)” (Boyd 1997). Boyd collected quantitative data concerning time to complete task for each of the three VEs that differed only in human-computer interfaces. The immersive VE used a Head Mounted Display (HMD) and a head tracked walking metaphor, while one of the non-immersive interfaces used a monitor, hand tracking, and a “puppet metaphor” for navigation, the other used a monitor, hand tracking, and a “vehicle metaphor” for navigation. Results of ten trials revealed the immersive system was most efficient in task completion. Although Boyd does not give exact raw data values, he does plot the mean trial times in seconds for the three separate paradigms. Boyd reports that for all of the times except two, “mean trial time is lower for the immersive design than for the other two designs, often by a large factor.”

Other research targets establishing methods of measuring presence. These efforts can be separated into two camps; qualitative research and quantitative research. Although it appears that both camps agree that presence is a subjective feeling of an individual “being in” a VE, quantitative researchers attempt to identify measurable physical components of the system that create a sense of presence. The qualitative camp attempts to provide tools to elicit subjective responses from users concerning presence

such as questionnaires. Witmer and Singer propose using a combination of two questionnaires; one to measure an individual's ability to experience presence, and another to measure a VEs ability to foster presence (Witmer and Singer 1998). The immersive tendencies questionnaire (ITQ) is used to measure an individual's ability to perceive presence, while the presence questionnaire (PQ) measures the individual's perceived presence in the VE. Witmer and Singer report that the combination of the ITQ and PQ are reliable measures of presence. They also conclude that there is a "consistent positive relation between presence and task performance in VEs"; however, similar to Snow's findings, this relationship is described as weak.

The research of Prothero, Parker, Furness, and Wells (Prothero, *et al.* 1995) is one example of attempting to discover a quantitative measurement of presence. This research attempts to link concepts of presence (defined by the researchers as "an illusion of position and orientation") and vection (defined as visually-induced illusory self-motion). If a link is proven to exist, research already conducted on vection could be applied to presence. Researchers plan to conduct experiments to evaluate participants' ability to distinguish between conflicting virtual and real cues, incorporating conflicting inertial and visual yaw oscillations. They hope to find a relationship between vection, subjective measurements of presence using a questionnaire, and the proposed objective measurement of presence.

Although research on presence is extremely important to VE-specific HCI, like testbed evaluations, it also cannot function independently as a complete usability engineering methodology. The findings of research on presence, and the manner in which this research is being conducted, should be incorporated into a usability engineering methodology as later discussed in this thesis.

3.4.3 VE Design Guidelines Development

Recently there have been a number of research projects focusing on producing sets of VE design guidelines. Hix and Gabbard (1998) created a multi-dimensional framework of usability characteristics specifically for VEs. This framework provides VE designers and evaluators with structured guidelines and other information addressing unique VE-specific interaction aspects. The framework is unique in the

comprehensiveness of its coverage and facilitates further VE interaction research using the framework as the starting point.

Kaur (1998) developed usability guidelines for designing VEs using a theory of interaction that has three connected models:

1. Task action model – “describing purposeful behaviour in planning and carrying out specific actions as part of user’s task or current goal/intention, and then evaluating the success of actions.”
2. Explore navigate model – “describing opportunistic and less goal-directed behaviour when the user explores and navigates through the environment.”
3. System initiative model - “describing reactive behaviour to system prompts and events, and to the system taking interaction control from the user.”(Kaur 1999).

Kaur’s objective was to use interaction modeling as a theoretical base to develop VE design guidelines. Her three models of interaction are an elaboration of Norman’s cycle of interaction. Her current work involves refining these theoretical models for use in designing and evaluating VEs.

3.4.4 Traditional Empirical Evaluation

While some researchers such as Kaur are developing usability guidelines for VEs, others are considering usability issues as part of the design and development of VEs. Examples of the latter tend to be educational VEs, such as ScienceSpace and NICE (Narrative-based, Immersive, Constructionist/Collaborative Environments), which are designed to support learning. During the design and development of ScienceSpace, educational VEs for teaching difficult science concepts such as Newtonian physics, Salzman, Dede, and Loftin (Salzman, *et al.* 1995) performed formative evaluations to examine three aspects of the interaction: usability, learning, and usability vs. learning. Participants in these formative evaluations included high school students and physics educators and researchers. During these evaluations, which helped in shaping the design and refinement of ScienceSpace, participants performed a series of selected activities thinking-aloud, while wearing a head-mounted binocular display. Participant data included researchers’ observation, questionnaire, and interview feedback. An important

aspect of the ScienceSpace evaluations was that participants also received multi-sensory (such as spatialized sound and haptic) cues while performing activities. Thus, the researchers found that students were more engaged in learning activities when more multi-sensory cues were provided to them.

3.4.5 Altered Empirical Evaluation

The NICE project, an experiential learning environment designed to engage young children in authentic activity, such as building persisting virtual worlds through collaboration, has an innovative evaluation framework that also examines usability issues (Roussos, Johnson, Moher, Leigh, Vasilakis, and Barnes, 1999). NICE is a virtual environment that allows users to manipulate plants in a garden as well as communicate with others that are located within the environment.

The evaluation framework has six categories: technical, orientation, affective, cognitive, pedagogical, and collaborative VR. The usability issues, which were part of the technical category, focused on: the time it took young children to learn the NICE interface on the CAVE or Immersadesk, physical and emotional comfort, and comprehension of instructions. Usability results revealed that the large size of stereo glasses caused discomfort to young children, primarily second grade students.

Both ScienceSpace and NICE formative evaluations aim to increase usability of the VEs. However, evaluation framework of these and other VEs lack a thorough heuristic evaluation by experts before attempting formative evaluations with representative users of the VEs.

4 Initial VE-Specific Usability Evaluation Method

4.1 Introduction to Initial VE-Specific Usability Evaluation Method

Our initial usability evaluation method consisted of two assessment steps, taken from traditional usability evaluation processes:

1. One or more VE usability specialists performed a usability inspection of the application. The inspection process was used to locate obvious usability issues and recommended redesign.
2. An empirical evaluation placed representative users in task-based scenarios comprised of specific benchmark tasks.

Combining usability inspections and empirical usability evaluations is not a new idea. Nielsen and Mack (1994) state that, “usability inspection methods are well suited as part of an iterative design process where they can be combined with other usability evaluation methods”. Nemire (1993) echoes this opinion by stating, “a combination of the two techniques can provide the most cost-effective solution.” Finally, Hix, Swan, Gabbard, McGee, Durbin, and King (Hix, *et al.* 1999) successfully assessed the navigation metaphor used in a real-time battlefield visualization virtual environment by utilizing an iterative application of usability inspection followed by empirical evaluation.

Combining the two usability evaluation techniques provided two benefits for VE assessment:

1. It facilitated more comprehensive coverage of existing usability issues since each technique has been proven to identify unique issues. Nielsen (1994) states, “each technique discovers usability issues that are often overlooked by the other.” Mack and Nielsen (1994) discuss the benefits of combining usability inspections and empirical evaluations as “user testing and usability inspection have a large degree of non-overlap in the usability problems they find.”
2. It provided a more economic solution than empirical evaluation alone. Because usability inspections are typically cheaper than empirical evaluations, any major issues that are fixed in the inspection phase will save evaluators from expending resources on these issues during empirical evaluations. VE usability specialists are familiar with interaction taxonomies, and therefore

can potentially remove many “obvious” usability violations upon initial inspection. As a result, participants are shielded from awkward or ineffective interaction techniques, thereby increasing the probability of uncovering domain-specific usability issues during empirical evaluations. Mack and Nielsen (1994) discuss the advantage of performing a usability inspection prior to a usability evaluation by saying, “a typical strategy is to apply a usability inspection method first to clean up the interface as much as possible, then to subject the revised design to the user.” Figure 5 shows the interface proceeding through a usability evaluation process that combines usability inspections and empirical evaluations.

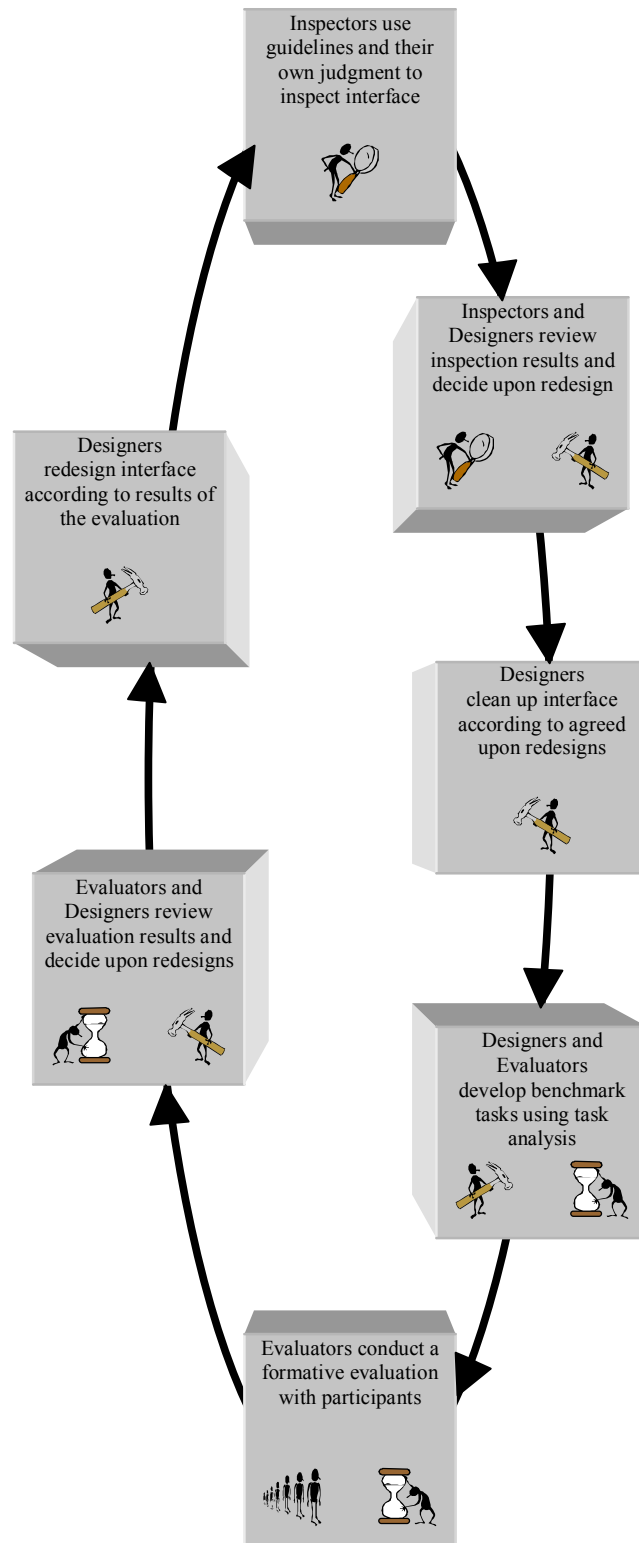


Figure 5: Initial Usability Engineering Methodology

The next logical step was to determine exactly which specific usability inspection

and empirical evaluation methods to utilize. Once again we drew from the previous work of Hix and Gabbard (1999) and adopted a strategy of coupling heuristic inspection with formative empirical evaluation. Heuristic inspection is built on the premise of “discount usability” and results in extensive coverage of usability issues at a fraction of the resource cost of other usability inspection techniques, such as cognitive walkthroughs. A heuristic inspection attempts to capitalize on the knowledge of usability specialists to discover as many usability issues as possible without the involvement of participants. Heuristic inspections also do not require usability specialists to have knowledge of behavioral science. This increases the number of available usability specialists and reduces resource requirements.

Formative evaluations attempt to collect large amounts of data from a few participants. A formative evaluation differs from other empirical evaluation techniques, such as summative evaluation, that require a larger population of participants in order to perform stringent data analyses. We chose heuristic inspection and formative evaluation as our evaluation techniques because of their proven background working together and their efficiency attributes.

4.2 Assessing Weaknesses of Initial VE-Specific Usability Evaluation Method

Since this was our initial evaluation of a VE using these usability evaluation methods, we added minimal modifications to facilitate VE-specific assessment. Our goal was two-fold:

1. To perform a usability assessment given traditional usability evaluation methods, and
2. To discover limitations in traditional methods when applied to the evaluation of VEs.

Due to the added complexity of VEs, we expected that heuristic inspection would require more than Nielsen’s suggested heuristics. Nielsen’s ten heuristics were created as a result of a factor analysis of 249 usability issues (Nielsen 1994). These 249 usability issues were gathered from usability evaluations on either GUI or command-line applications, not VEs, and therefore do not include unique VE-specific usability issues. We concluded that simply applying these traditional heuristics in a usability inspection

would be based upon a “dangerous” and erroneous assumption that VEs contain only those usability issues that exist in GUI or command-line applications. Until a comprehensive set of VE-specific usability issues has been determined using factor analysis, performing a usability inspection on a VE using existing heuristics would not result in a comprehensive or even appropriate usability evaluation.

We also expected that formative evaluation used in its traditional form would not effectively evaluate unique issues encountered in assessing VE usability. Most formative evaluations are designed to collect qualitative and quantitative data based on participants completing benchmark tasks created using task analysis. Because most GUI applications conform to a particular standard interaction style (e.g. WIMP), current formative evaluations are less concerned with interaction metaphors leveraged by an application and concentrate more on discovering domain-specific usability issues. However, the current state of VEs is not conducive to this approach. As discussed previously, there are no standardized VE interaction styles, and many VEs utilize unique interaction metaphors. Therefore, formative evaluations must be flexible enough to handle evaluation of complex and non-standard interaction styles by discovering unique interaction metaphors that exist within the VE and developing tasks specifically to test those metaphors.

4.3 Making Modifications to Traditional Usability Evaluation Methods

To ensure that our inspection methodology would include coverage of usability issues unique to VEs, we needed a set of VE-specific usability heuristics to couple with Nielsen’s heuristics. We decided the set of VE usability design guidelines contained in the *Framework of Usability Characteristics in Virtual Environments* provided a “reasonable starting point for usability evaluation” (Gabbard and Hix 1999). However, because there are 195 guidelines in the *Framework*, we referred to this usability inspection technique as a guidelines-based inspection rather than a heuristic inspection. Gray and Salzman (1998) agree that short lists used for heuristic inspections are referred to as heuristics, while long lists are referred to as guidelines. Therefore, in our initial usability evaluation methodology, we proposed that we perform a guidelines-based inspection of the VE in the spirit of Nielsen’s heuristic inspection, using guidelines available in the *Framework of Usability Characteristics in Virtual Environments*.

Our formative evaluation modification to facilitate the assessment of unique interaction metaphors relied heavily on the guidelines-based inspection. If inspectors encounter unique interaction techniques that are not clearly in violation of well-established usability guidelines, then they can recommend that a task specifically designed to assess usability of that particular technique be included in the formative evaluation. Therefore, formative evaluations include tasks assessing the usability of innovative interaction styles that were “ear-marked” by usability specialists during a guidelines-based inspection as requiring further evaluation. This strategy does not replace task analysis, but rather is coupled with task analysis to create a set of comprehensive benchmark tasks. Figure 6 shows these modifications to the initial usability methodology.

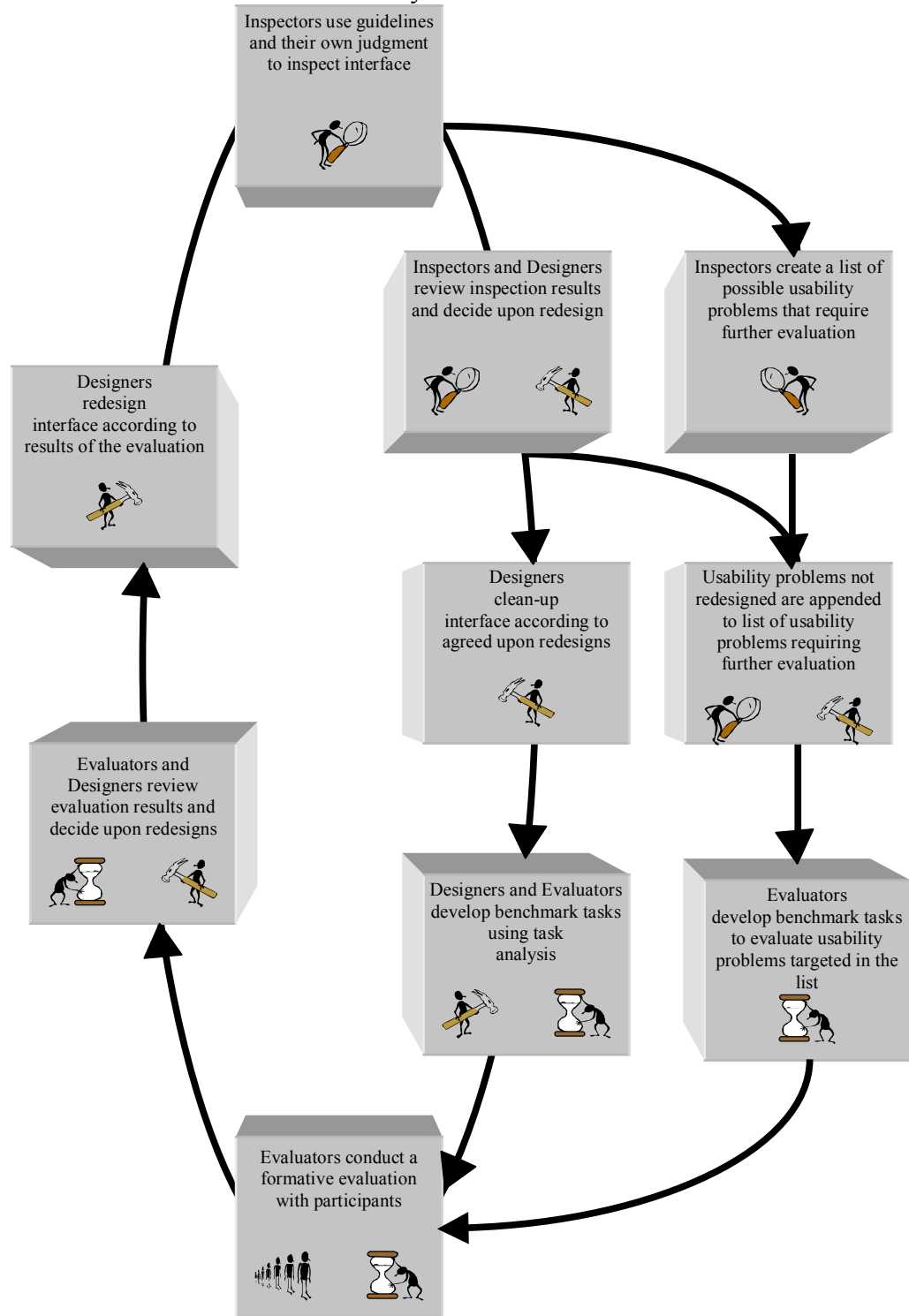


Figure 6: Modified Evaluation Method

5 Applying the Usability Evaluation Method to Crumbs

5.1 Crumbs Introduction

Researchers at Virginia Tech worked in collaboration with the NCSA Biological Imaging Group to apply this usability evaluation technique to an immersive visualization VE developed at NCSA. The VE, known as Crumbs, runs in a CAVE and is used primarily as a tracking tool for biological and medical imaging. It provides users with a way to visualize dense volumetric datasets. Crumbs' developers leveraged advantages of visualization and immersion to facilitate identification of complex biological structures (Brady, *et al.* 1995). Users can also mark and measure individual structures in the dataset.

Crumbs provides users with environmental objects, 3D widgets, and menus to interact with the system. The primary method of user interaction is by means of a wand, the CAVE default input device. The wand is a six degrees of freedom device used in Crumbs to provide similar functionality to a mouse in the standard GUI paradigm. The wand is used to

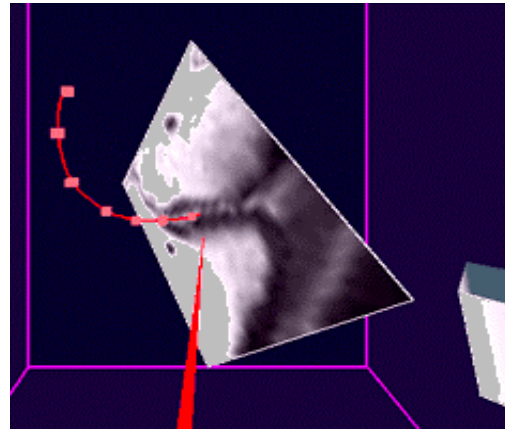


Figure 7: Crumbs Fiber Tracing

manipulate a volumetric dataset and mark points as well as navigate “pop-up” menus. Crumbs also supports voice input as a second method to perform commands on menu items. Crumbs has other objects that assist the user in visualization, marking, and measurement tasks. Below is a list of the most important of these objects and a description of their job responsibilities.

- *data volume object* – A box containing the volumetric data set loaded into Crumbs.
- *crumb object* – An object used to mark specific structures in the data set. A line is drawn between subsequently created crumb objects to create a line segment referred to as a fiber.
- *clearbox object* – A box of a specified width used to render specific regions of the data volume in a specified resolution. A clearbox object is used to visualize certain sections of the data set located in the data volume object.

- *sliceplane object* – Visualizes a single slice in the data volume at a specified resolution.
- *trashcan object* – Object used to delete other objects. Objects are removed from the environment by “dragging and dropping” them into the trashcan.
- *colormap object* – A 3D widget allowing the assignment of specific density values of the data set to specific colors. This widget relies on a color cube metaphor.
- *auxiliary colormap object* – A 3D widget performing two functions. First it allows restriction of viewable density values in the data volume. Secondly it provides a visual representation of the color to density mapping currently supported by the colormap object.
- *opacity object* – A 3D widget allowing specification of opacity values for specific density values of the data set.
- *sword* – Metaphor used to describe the physical appearance of the wand in the virtual environment. The sword relates to the wand as the cursor relates to the mouse in a GUI application.

5.2 Guidelines-Based Evaluation of Crumbs

We systematically applied the guidelines in the *Framework of Usability Characteristics in Virtual Environments* as a guidelines-based inspection of Crumbs, as detailed in the following sections.

5.2.1 Method

Because the *Framework* contains 195 guidelines, it was not practical to require the usability specialists to memorize the guidelines prior to the evaluation. Nor was it considered feasible to require the specialists to take a list of guidelines into a CAVE. Therefore, usability specialists were directed to evaluate Crumbs one *Framework* section at a time. We knew that nineteen separate evaluations of Crumbs would prove time-consuming because the *Framework* contained nineteen sub-sections. However, since VEs vary widely in scope and devices, we expected many guidelines would not pertain to a particular application, and therefore several evaluations would be very short in duration.

The developers of Crumbs were constrained to using a CAVE and a wand. As a result of these constraints, many guidelines located in the *VE UI Presentation Components* and *VE User Interface Input Mechanisms* were not applicable in this inspection. These guidelines target VE designers and focus on issues crucial for determining appropriate input devices and presentation components for VE systems. Consequently, once these devices and components have been selected and software development has begun, designers likely will not have sufficient time or resources to alter these original decisions. Therefore, the assessment question changed from the best possible VE system, to the best implementation of the available VE, taking into account constraints of input devices and presentation components already in place. As a result, our inspection concentrated primarily on the *Users and User Tasks* section as well as the *Virtual Model* section of the *Framework*.

Following Nielsen's original method of performing heuristic inspections, our guidelines-based inspection method was based entirely on usability specialists performing free exploration of Crumbs to examine specific features of interaction metaphors without following specific tasks. We chose to rely on free exploration with the knowledge that we would perform a formative evaluation subsequently concentrating on pre-defined tasks and scenarios. With free exploration, usability specialists were not forced to concentrate on task-related issues and thus were able to focus entirely on assessing interaction metaphors for usability issues.

5.2.2 Evaluators

Guidelines-based inspection of Crumbs took place at the Virginia Tech CAVE facility and involved collaboration of VE usability specialists from Virginia Tech and Crumbs' designers from NCSA. A single usability specialist performed the guidelines-based inspection. Although, utilizing only a single usability specialist had obvious limitations, it did provide advantages regarding the intentions of this research. Since this was an inaugural attempt at performing our proposed guidelines-based inspection methodology, we intended to use this opportunity as a pilot test. Therefore, we decided not to include multiple usability specialists in the guidelines-based usability inspection methodology until we were certain that the methodology was sound. This initial attempt

was designed to iron out rough spots in the inspection process prior to investing larger resources.

5.2.3 Results

According to Bowman (1998), at the lowest level, VE tasks can be broken into three categories: locomotion, object selection, and object manipulation. Locomotion is the task of interactively moving the viewpoint within the environment. Object selection is the task of selecting one or more objects in the environment. Object manipulation is the ability to act upon currently selected objects. Two vertical tasks sit on top of these core tasks: system commands and domain-specific tasks. Crumbs' usability inspection focused on Crumbs' ability to facilitate system commands and domain-specific tasks using locomotion, object selection, and object manipulation. Table 3 summarizes the usability issues and possible redesign recommendations as given to Crumbs' developers. Sixteen problems were identified and the usability specialist worked in cooperation with developers to incorporate several redesign recommendations into Crumbs.

Framework Guideline	Usability Issue	Redesign Recommendation
Accommodate natural, unforced interaction for users of varied age, gender, stature, and size.	Using the wand to utilize cascading menus is somewhat difficult. Natural arm motion includes a tendency to arc the movement instead of a straight movement. This causes change in original menu item prior to accessing sub-menu.	Redesign cascading menu interaction to display only the sub-menu when the cascading menu item is "clicked", and remove cascading menu item when either a menu item or some other position in environment is "clicked". This interaction technique is popular with several GUI standards.
Language and labeling for commands should clearly and concisely reflect meaning.	Crumbs' menu system uses language such as "Toggle Spline" with no symbolic representation whether the spline is currently in use or not. This means user must know the state of Crumbs prior to entering a menu.	Simply using an icon to represent when a certain utility is in use would potentially eliminate this added task.

Framework Guideline	Usability Issue	Redesign Recommendation
Pay close attention to the visual, aural, and haptic organization of presentation (e.g., eliminate unnecessary information, minimize overall and local density, group related information, and emphasize information related to user tasks).	Users are allowed to perform commands that are not valid considering system state. For example, the user can execute the “Toggle Spline” menu item when there are less than two crumbs placed.	“Gray out” commands that are not valid.
Provide accurate depiction of location and orientation of surfaces.	Cognitive affordance used for orientation within the dataset cube object (having the axis of the cube color coded) requires the user to remember which color correlates to which axis.	Design a task in the formative evaluation to test usability of color coded axis in facilitating situational awareness of dataset orientation.
Provide stepwise, subtask refinement including the ability to undo.	Positioning multiple crumb objects in the dataset is a stackable task, and Crumbs should consider a quick method of unstacking (i.e., undo).	Design a task in the formative evaluation to determine if the user desires an undo capability.
Strive to maintain interface consistency across the application.	Crumbs is not consistent with its use of the term “crumbs” within the menu system. Identical items are also referred to as “points”.	Make the interface consistent using either “crumbs” or “points”.
Strive to maintain interface consistency across the application.	A direct manipulation technique, dragging an object to the trash can, is used to delete objects. However, an indirect manipulation technique, selecting a menu item, is used to exit or close objects.	Offer the user a direct manipulation method of closing objects.
Strive to maintain interface consistency across the application.	Not all cascading menu items are followed with a trailing semi-colon.	Ensure all cascading menu items are followed with a consistent cognitive affordance.
Strive to maintain interface consistency across the application.	Whenever the colormap or the opacity objects are active, any attempt to use the middle wand button (no	Crumbs should only manipulate the object that is currently selected by pointing.

Framework Guideline	Usability Issue	Redesign Recommendation
	matter where in the environment the user is pointing) affects the Colormap and Opacity objects. This is not consistent with direct manipulation interaction metaphor used by other objects.	
Strive to maintain interface consistency across the application.	Selection of object is facilitated using the left wand button, selection of sub-objects - individual parts within an object - is facilitated using the middle button.	Selection of an entity whether it be object or sub-object should be facilitated using the same action. Therefore restrict selection methods to wand left button clicks.
System messages should be worded in a clear, constructive manner so as to encourage user engagement (as opposed to user alienation).	Selecting an existing crumb object generates an audio cue “Ouch”. Also, attempting to manipulate one of the opacity objects generates the seemingly sarcastic comment “good luck”.	Change audio cues to clearly indicate that a crumb has been selected. Change audio cue to clearly indicate manipulation of the opacity object.
Take into account user experience (i.e., support both expert and novice users).	Use of the opacity object is not initially intuitive and requires training for proper use.	Design a task in the formative evaluation to assess usability of the opacity object.
Take into account user experience (i.e., support both expert and novice users).	Most cascading menu items are presented with a trailing semi-colon. Most applications use ellipses (...) or an arrow (➔) to indicate a cascading menu.	Change semicolons to arrows or ellipses.
Take into account user experience (i.e., support both expert and novice users).	Use of the colorbox object is not initially intuitive and requires training for proper use.	Design a task in the formative evaluation to test the usability of the colorbox object.
Take into account users’ technical aptitudes (e.g., orientation, spatial visualization, and spatial memory). Pay close attention to the visual,	The only method of feedback available in Crumbs for “Arc Length” execution is audio. This is not functional for hearing impaired individuals or	Create a visual object to display arc length.

Framework Guideline	Usability Issue	Redesign Recommendation
aural, and haptic organization of presentation.	hardware setups that do not have audio equipment. The number is printed to the unix window on the workstation, but should be included in CAVE environment.	

Table 2: Guidelines-Based Usability Issues and Redesign Suggestions

Crumbs' developers chose to follow the redesign recommendations that primarily involved usability issues concerning the menu system.

1. Menu item labels were modified to convey their associated task more precisely.
2. All menu items that had previously used a *toggle* <item> label convention, such as "Toggle Spline", were modified to make use of a <item Off> and <item On> label convention. For example, "Toggle Spline" was changed to "Spline Off" and "Spline On".
3. Foreground color used for menu item labels was changed from red to yellow to provide more contrast with the light blue menu background.
4. The word "points" no longer appeared on the menu system; it was instead switched to "crumbs" to maintain metaphor consistency.
5. All menu item labels for cascading menus were given ellipses (...) as a cognitive affordance.

However, Crumbs' developers decided not to act upon some usability issues reported in the inspection, based primarily on their assessment of cost justification of proposed redesign efforts. Several suggestions would have required designers to redesign large sections of Crumbs. It was the designers' opinion that the redesign effort either required too many resources to compensate for gain in usability or that the time required for redesign would exceed the timetable set for formative evaluation. Therefore several reported usability issues were not corrected prior to formative evaluation. Two usability issues designers chose not to redesign are:

1. Change audio cues to indicate clearly that a crumb has been selected and manipulation of the opacity object.
2. Offer the user a direct manipulation method of closing objects.

The guidelines-based inspection also identified several possible usability issues that usability specialists and developers decided to postpone for redesign consideration until the issues were validated as problems using formative evaluations. These usability issues were not in direct violation of any VE-specific guideline, but usability specialists performing the guidelines-based inspection believed they needed further assessment to ascertain whether they were indeed a usability issue:

1. Usability of the opacity object.
2. Whether the user desires an undo capability.

Tasks were subsequently created in the formative evaluation to require participant interaction with issues that were either marked as “require further evaluation” or were not redesigned.

5.3 Formative Evaluation of Crumbs

After modifications suggested by the guidelines-based inspection were completed by Crumbs developers, we performed a formative evaluation of Crumbs as described in the following sections.

5.3.1 Method

Formative evaluation of Crumbs took place at NCSA and included collaboration of VE usability specialists from both NCSA and Virginia Tech. This initial evaluation served a dual purpose: as a pilot test to iron out potential difficulties in evaluation procedures and as an evaluation of Crumbs using domain expert participants.

5.3.2 Participants

This evaluation targeted expert participants with both a working knowledge of Crumbs and experience performing similar tasks as ones used in the evaluation. From prior use of Crumbs, expert participants had experience interacting within an immersive three-dimensional VE and required little or no training on the application. Although

expert participants varied in their individual application experience, VE experience, and domain experience, their familiarity with the application and tasks provided enough consistency to validate categorizing them as experts. Five people were chosen to participate in the formative evaluation based on the suggestions of Crumbs developers. The first participant was used as a pilot participant. Table 4 lists the amount of time each participant had spent working with any VE environment, the amount of time each participant had previously spent working with Crumbs, and for what purpose each participant had previously used Crumbs.

Participant #	Crumbs Experience in Months	VE Experience in Months	Use of Crumbs
1*	2	6	Demonstrations
2	1	7	Demonstrations, analyzing Crumbs interactions, look at potential applications for VR
3	.5	.5	Visualizing cone-beam optical tomographic data
4	1.5	1.5	Interactive segmentation project
5	4	4	Study of the biology of fertilization

* Was used in the Pilot Study

Table 3: Participant Experience Comparison

Due to the infancy of immersive environments in the medical and biological visualization fields, there are few users that have experience using Crumbs. Users possessing such experience are mostly located at NCSA, and have close ties with Crumbs' developers. This constraint on participant selection introduced a potential threat to the validity of this study because of the very small pool of representative Crumbs' users. This situation is not unusual for a very specialized application such as Crumbs. However, participant backgrounds were diverse enough to provide a reasonable generalization to the larger population of potential Crumbs' users.

5.3.3 Equipment

All formative evaluation sessions occurred in the NCSA CAVE. We used a stopwatch to record elapsed time for participants to complete certain tasks. During the pilot test, we used two concurrent streams of video to record the evaluation. One stream

used a scan converter to record the front wall of the CAVE, while the second stream used a video camera to record participant reactions. A wireless microphone was used to record the participant's comments. Because the front wall of the CAVE is displayed in stereo, recording it proved to be less useful than we hoped and therefore was not used in evaluation sessions following the pilot study.

5.3.4 User Tasks

Tasks used in the formative evaluation were created from two separate processes:

1. *Task Analysis* - usability specialists requested assistance from Crumbs' developers to construct a set of user tasks representative of common system use.
2. *Guidelines-based inspection* - tasks were added to assess both usability issues not modified prior to evaluation and usability issues requiring further assessment.

Three sets of tasks were used for the evaluation. Two sets concentrated on tasks used to visualize and measure structures located in a visualized dataset. The other set of tasks addressed the validity of Crumbs' sonification in representing various values of volumetric density.

One dataset was created using a spiral CAT scan. The resulting dataset is a human's backbone vertebrae. The spine data volume contains seven vertebrae of the lower back of a human male. The user's task was to position seven markers at the three dimensional centroid of each vertebra. This "negative" task of placing a marker in a hole required participants to select the size and resolution of the imaging tools to maintain sufficient detail while still viewing global context of the hole boundaries.

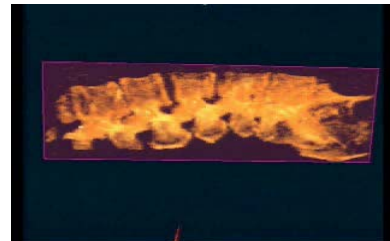


Figure 8: Vertebrae Dataset

Another dataset was created using laser scanning confocal microscopy. The resulting dataset is the tail of a fruit fly sperm. This dataset was created to research the length and pattern of fruit fly sperm tails found within fertilized fruit fly eggs. The sperm tail data volume consists of a one-dimensional strand that is curving and wrapping around

itself in three-dimensional space. Participants were presented with a partially traced tail, and their main task was to complete the tracing. Participants needed to position themselves and viewing tools in such a manner that tail location was easily visible. Participants placed markers along the tail in the highest density regions of the strand. This task assessed whether participants could place a marker in a “positive” location, that is, if participants could place a marker within a dense region where the feature being traced was specifically visible.

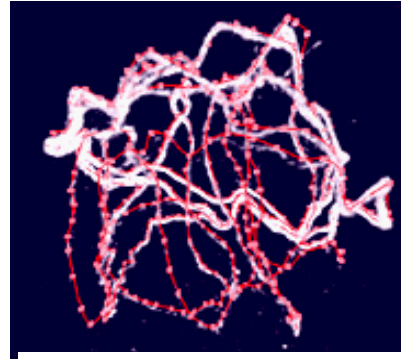


Figure 9: Sperm Tail Dataset

For the third dataset, Crumbs’ developers created Slider, an application that facilitates the investigation of Crumbs’ sonification implementation. Slider specifies a certain audio output to represent a specific density. The task set that assessed Crumbs’ sonification used Slider to provide participants with aural stimuli used in Crumbs to represent various density values.

5.3.5 Procedures

Participants were brought individually into the NCSA CAVE facility to perform the formative evaluation. This is a valid setting because the scarcity of CAVEs will require most users of Crumbs to work at similar laboratories. Evaluation procedures were broken into three categories:

1. Pre-evaluation procedures:
 - a. Participant Permission Form was the initial introduction to Crumbs and to this particular evaluation. It also included a signed participant consent form. (See Appendix A)
 - b. Crumbs CAVE Evaluation Pre-Test Survey allowed evaluators to collect data regarding participants’ background and familiarity with Crumbs. (See Appendix B)
 - c. Participant Instructions further introduced Crumbs’ interaction techniques, explained the opacity object and the colormap object to participants, and introduced evaluation procedures. (See Appendix C)

- d. Prior to starting each evaluation, the sperm tail data volume was loaded and participants were given a brief tutorial and some time for unstructured investigation of Crumbs.
2. Evaluation procedures:
 - a. The spine dataset was loaded and participants were instructed to complete a set of tasks. See Data #1: Task # 1 through 16 in Appendix D for a complete listing of tasks performed on the spine dataset.
 - b. Participants were escorted to a chair for completion of the second set of tasks that focused entirely on sonification. Slider was executed and the evaluators stepped through the list of tasks for the second section. See Data #2: Task # 1 through 5 in Appendix D for a complete listing of tasks performed using the slider application.
 - c. Participants were escorted back into the CAVE and the sperm dataset was loaded into Crumbs. The participant was instructed to complete a set of tasks. See Data #3: Task # 1 through 16 in Appendix D for a complete listing of tasks performed on the sperm dataset.
3. Post-evaluation procedures:
 - a. The final step in the evaluation was completion of a questionnaire by the participant, followed by an open question and answer session where the participant was free to give comments and suggestions on Crumbs and this evaluation.

5.3.5.1 Pilot Study Procedures

The pilot study was conducted with two evaluators. Evaluator #1 functioned as the mediator and collected data on the timing and error form (Appendix H). Evaluator #2 controlled the video recording and collected data on Participant Comment Form (Appendix I). Evaluator #1 was standing in close proximity to the participant to assist as a mediator, while Evaluator #2 was sitting outside the CAVE, as shown in Figure 10.

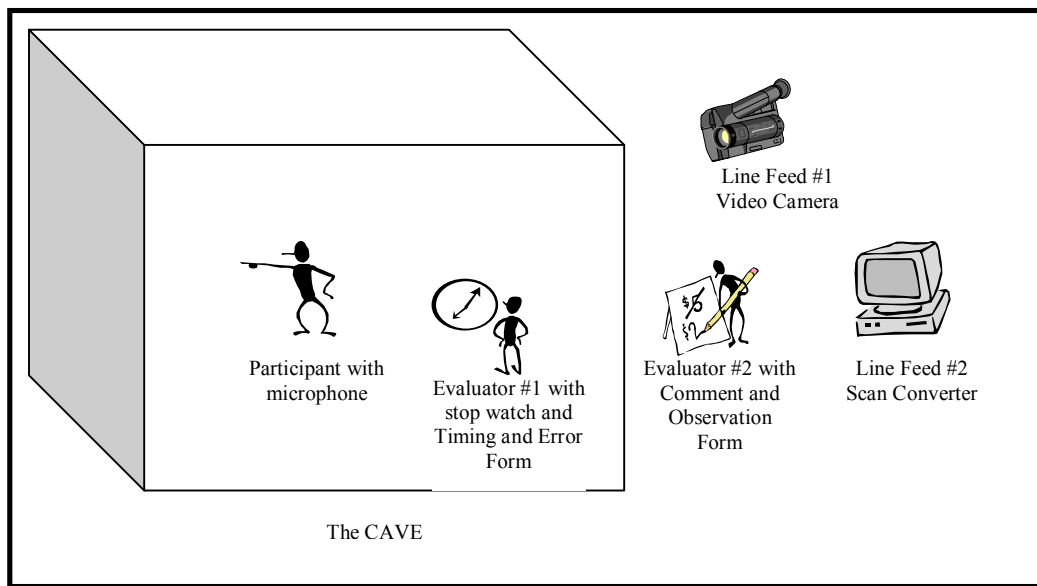


Figure 10: Pilot Study Physical Setup

For each task, the two evaluators and one participant went through a sequence of discrete steps to ensure that everyone was ready to start the next step. During the pilot study, each task followed this sequence of steps:

1. Evaluator #1 reads the task out loud.
2. Evaluator #1 asks participant if they understand the task.
3. Evaluator #1 indicates verbally that the task should begin and starts the stopwatch.
4. Participant performs task.
5. Participant verbally indicates that task is complete.
6. Go to next task.

This sequence of steps proved to be the cause of much confusion during the pilot study. Evaluator #2 often was not sure which task the participant was currently performing. Also, between tasks, while data were being recorded, the participant would often interact with the application and at times perform steps in preparation for what they believed would be the next task. This invalidated any task completion data on subsequent tasks.

5.3.5.2 Main Evaluation Procedures

Based on procedural problems discovered during the pilot study, we modified the physical setup for the main evaluation. The principal difference was positioning Evaluator #1 outside the CAVE. This alteration was made for three reasons:

1. Evaluator #1 positioned in the CAVE, created the potential for them to interrupt a participant during task completion by standing in a physical location that was required for user interaction with Crumbs. This interference occurred during the pilot study when the participant wanted to select an object that was located behind and to the left of the evaluator.
2. The NCSA CAVE facility has wires across the floor connected to interaction devices. Quite frequently, if the evaluator is not paying close attention, these wires get tangled up in the evaluator's feet, distracting the participant from the task at hand.
3. Evaluator #1 positioned in the CAVE, mentally interfered with the participant's feeling of presence. The evaluator's close physical proximity to the participant affected the participant's ability to remove themselves from the physical surroundings of the CAVE and mentally transport themselves into Crumbs' VE.

Another difference between the pilot study and the main evaluations was the use of a third evaluator. Because we felt that the pilot test evaluators were overloaded with responsibilities, most of the main evaluations were conducted with three evaluators. Evaluator #1 continued to function as the mediator and collected data on the timing and error form (Appendix H), but limited his data collection to recording critical incidents and counting errors. Evaluator #2 continued to control the video recording and collected data on the participant comment and evaluator observation form (Appendix I). Evaluator #3 handled the stopwatch and recorded elapsed completion time for each task. All three evaluators were located outside the CAVE in all evaluations following the pilot study, as shown in Figure 11.

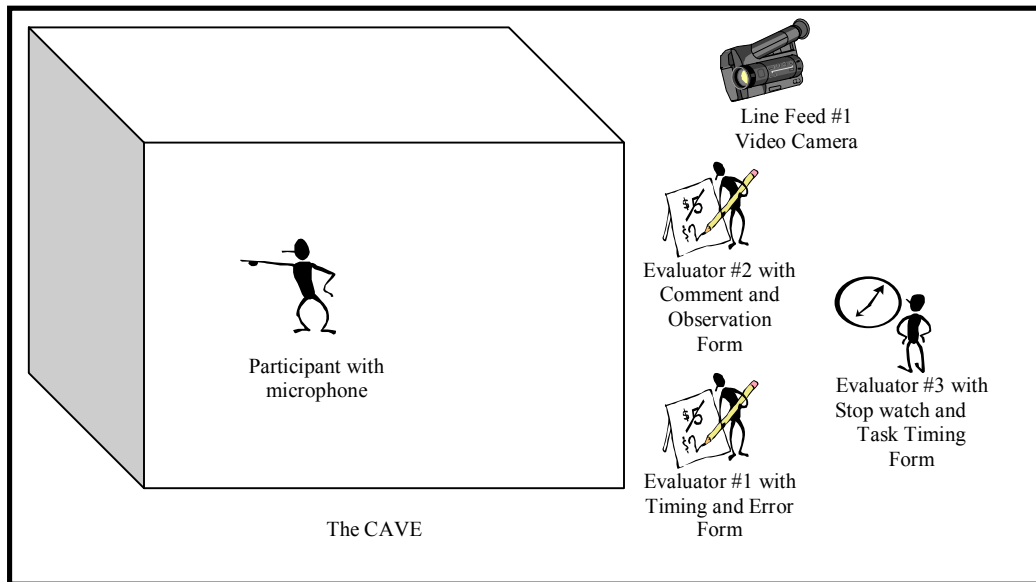


Figure 11: Main Evaluation Physical Setup

Evaluators determined that a more structured set of steps must be enforced to ensure collection of useful data. Also, the additional structure ensured that all evaluators were recording data for the same task. An exact phrase, “Begin task” was used consistently by Evaluator #1 to begin each task, and participants were instructed to use “Task finished” to indicate the end of each task. Also, participants were asked to refrain from interacting with the application between tasks and to discuss the previous task. Although these changes did somewhat interrupt the natural flow of certain domain tasks and potentially negatively affected the participant’s feeling of presence, they were deemed necessary by the evaluators to collect the desired data accurately. The pilot study sequence of steps was altered to:

1. Evaluator #1 says the number of the current task.
2. Evaluator #2 and #3 indicate that they are prepared to take data on announced task.
3. Evaluator #1 reads task out loud.
4. Evaluator #1 asks participant if they understand task.
5. Evaluator #1 occasionally reminds participant that they should verbally indicate when they are finished with a task.
6. Participant turns toward front wall of CAVE.
7. Evaluator #1 says “Begin task” to indicate that the task should begin.

8. Evaluator #3 starts stop watch.
9. Evaluator #1 records task errors.
10. Evaluator #2 records participant comments and evaluator observations.
11. Evaluator #2 controls video equipment.
12. Participant says “Task finished” to indicate task completion and turns toward the evaluators and away from the environment.
13. Evaluator #3 records the time.
14. Evaluator #1 asks participant if they have any comments.
15. Evaluator #2 records any comments.
16. Go to next task.

This sequence of steps worked well to orchestrate two or three evaluators and a participant during the main evaluation sessions.

5.4 Results

Because Crumbs is functionally unique, there were no comparative quantitative data. Due to this absence of comparative data, to discover usability issues, we chose to focus largely on qualitative data recorded in the form of critical incidents (Hix and Hartson 1993). Quantitative data in the form of task completion times, task error counts, and questionnaire scores were collected, but were used mostly to support qualitative findings. Participants were encouraged to talk during and between each task to facilitate qualitative data collection. Evaluators recorded participant comments and wrote down any factors the evaluator attributed to the cause of the comment or an observed task-related error. Qualitative data recorded during pilot testing were also evaluated. The following section discusses usability strengths and usability issues.

5.4.1 Strengths

5.4.1.1 Wand as selection metaphor

Crumbs relies upon ray casting for selection using the wand location and orientation as the focal point of the ray. The tip of the ray acts as a “hot spot”. The positioning of the ray tip at time of selection determines the object selected. The ray has a constant length and is referred to in Crumbs’ documentation as the sword. Selection is

done in general circumstances using the left button of the wand, but selection of objects within objects is done using the middle button. For the most part, the “sword” caused no observable problems, and the metaphor appeared to be well understood by all participants. Several participants made reference to sword size and one participant made the suggestion to provide application users with the ability to control sword size. There were no notable instances in the quantitative tasks where sword size would have been beneficial to task completion. One participant wanted objects in the environment that are near the sword or at the end of the sword “to turn a different color so one can see when the end of the sword is approaching them”. This comment is interesting because most objects in Crumbs turn red when the sword is in a position to select them. Evidently, this highlighting was not sufficient in assisting this participant in object selection. Question 6 of the questionnaire asked participants to rank usability of the sword as a selection tool. The result of this question was a score of 3.25 out of 4. This relatively high score indicates that participants were reasonably satisfied with use of the sword.

5.4.1.2 Menu as command metaphor

The primary method of performing commands in Crumbs is by use of the menu system. For the most part, locating and selecting commands from the menu was observed to be easy and free of errors. The few problems we observed during the evaluation were connected with Crumbs’ implementation of the menu metaphor and not with the metaphor itself.

There were a number of selection tasks that required participants to use Crumbs’ menu system.

Task #	Description	Average Error Count	Average Completion Time
1.1	Turn on the low-resolution full visualization of the spinal cord dataset.	0.5	7.455
1.11	Determine the length of the arc created by the set of points.	0.25	34.54
3.7	Save the crumbs.	0	8.89
3.12	Save the crumbs.	0.33	9.52
3.15	Save the crumbs.	0	6.75

Table 4: Menu Benchmark Tasks

Most quantitative data regarding error counts on menu item tasks demonstrated no usability errors. Tasks that required use of the menu system (tasks 1.1, 1.11, 3.7, 3.12, and 3.15 – see Appendix D) had average error counts of 0.5, 0.25, 0, 0.33, and 0 respectively. All task completion times, except for task 1.11 of these five tasks, were under 10 seconds, further demonstrating the effectiveness of the menu system. Task 1.11, however, had an average task completion time triple that of the other three tasks. It took participants an average of 34 seconds to browse the menu system and locate the menu item that allowed them to determine the length of the fiber created by connecting all placed crumbs. When given the task of determining fiber lengths, two participants simply stated, “I don’t know how to do that” and waited for assistance from an evaluator. After participants realized the task involved use of the menu system, location of the fiber length menu item caused many of the participants to spend a substantial amount of time exploring the menu system looking for the command. This indicates that there is a usability issue associated with finding this menu item.

One participant made a statement concerning location of one of the other menu items. When prompted to save the crumbs, the participant displayed his discontent with the location of the “Save Crumbs” menu item stating, “(t)he save command should be on top of the menu instead of on the bottom”. He was the only participant that made such a claim, and the evaluators did not observe any other participant having difficulty with location of this menu item. Finally, Question 2 of the questionnaire asked participants to rate usability of the menu system. The menu system received an average score of 3.25 out of 4.

5.4.1.3 Inclusion of Audio Feedback

Crumbs has varying levels of audio feedback complexity that vary according to specific tasks. It provides simple audio annotations to indicate the completion of certain tasks such as deleting a crumb. It also makes use of a complex sonification feedback in an attempt to assist users in the placement of crumbs. The success of the sonification is not addressed in this section.

Task #	Description	Average Error Count	Average Completion Time
3.9	Delete the selected crumb	0	8.62
3.10	Delete the last 2 crumbs marked in the previous task.	0.33	23.26
3.13	Delete the last 3 crumbs marked in the previous task.	0.67	46.09

Table 5: Audio Feedback Benchmark Tasks

Tasks (see Appendix D) were created to assess the benefits of the task completion audio feedback. Tasks 3.9 and 3.10 involved deleting three crumbs with audio feedback in place. Task 3.13 asked participants to remove three crumbs without audio feedback. Since the combination of tasks 3.9 and 3.10 are equivalent to task 3.13, the results were compared accordingly. Since the audio feedback assisted tasks were executed prior to the unassisted task, one would hypothesize that error rates and completion times would decrease due to familiarity and learnability of the tasks. Tasks 3.9 and 3.10 resulted in values of 0.33 for average error count and 31.79 seconds for task completion time. This is compared with the error count of 0.67 and completion time of 46.1 seconds for task 3.13. Because the sound was not available for task 3.13, the increase in errors and task completion time is attributed to the lack of audio feedback.

5.4.1.4 Location awareness in data volume

A common problem seen in VEs is the ability to lose location and self-awareness with respect to other objects sharing the environment. Crumbs facilitates location awareness by implementing the object in hand navigation metaphor and color-coding data volume axis. Quantitative data demonstrated no usability issues associated with the awareness.

Task #	Description	Average Error Count	Average Completion Time
1.5	Scale the data volume to a workable size.	0	6.82
1.6	Manipulate the data volume to a viewpoint where you are looking down the spine.	0.25	7.62
3.3	Scale data volume for easy viewing.	0.25	11.02

Table 6: Local Awareness Benchmark Tasks

Tasks 1.5 and 1.6 and task 3.3 (See Appendix D) directly assess usability of the data volume object. These tasks required participants to scale and orient the data volume rendering object to facilitate visualization of the dataset. These tasks had average error counts of 0, 0.25, and 0.25 respectively. Average completion times for the tasks were 6.82, 7.63, and 11.02 seconds. Question 9 of the questionnaire (Appendix F) asked the participant if they were ever disoriented within the data volume. The average score on the question was 3.125 on a scale of 1 to 4, which would indicate that Crumbs did support awareness. Qualitatively participants never mentioned being lost or disoriented in the data volume. A more thorough plan to evaluate awareness is needed before we can be confident about Crumbs' support of situational awareness.

5.4.2 Usability Issues

Assessment of quantitative and qualitative data resulted in a number of usability issues. These problems were further investigated using Gabbard's framework to discover issues that violated one or more guidelines. All references to "the framework" in the following sections are referring to Gabbard's work. Certain problems not only violated the set of specific VE guidelines, but also violated general HCI guidelines and these violations were also noted. Each usability issue was assigned a unique number that is used only as a reference (not to be potentially confused with an ordinal ranking for importance of that issue, for example). For each issue we discuss specific VE and general usability guidelines it violates, and we also discuss possible strategies for redesigning Crumbs to address the issue.

5.4.2.1 Issue #1: Awareness of middle wand button for scale/mark mode

5.4.2.1.1 Description

The middle wand button is a facilitator of multiple actions within Crumbs. The middle button's primary functions are to scale the data volume, manipulate objects, and change the settings of crumb objects such as the colormap and opacity objects. To switch between button functionality modes, the user selects a menu item to issue the appropriate command. Although the menu system provides a cognitive affordance for the current mode, participants must constantly bring up the menu system and check the middle

button sub-menu to ensure the current mode is the desired one. Evaluators observed two participants who constantly brought up the middle button sub-menu to check the middle button mode. This cognitive affordance helped to eliminate errors for participants at the cost of increased task completion times. One participant specifically addressed multi-functionality for the middle button saying, “(t)he middle button has too many functions,” and “(s)ometimes I inadvertently drop a crumb when I think I am going to scale.”

The result of pressing the middle button is further complicated because it is used as the method of altering setting for the colormap and opacity objects. In these cases, Crumbs does not provide a cognitive affordance of the mode change. For example, evaluators witnessed situations where participants were manipulating the setting of either the opacity or colormap objects and needed to scale the data volume prior to continuing their task with the current object. When participants attempted to alter the data volume by setting the middle button mode to scaling using the system commands provided by the menu system, they in effect changed the mode so that they could no longer manipulate opacity object settings. Being unaware of this change in mode, participants attempted to manipulate opacity object settings, but in turn actually scaled the data volume. To reset the middle button mode to allow for opacity tool manipulation, Crumbs requires users to exit the opacity tool and then re-edit it.

5.4.2.1.2 Guidelines

In reference to using modes in an application, Hix and Hartson (1993) state, “the designer should be careful to distinguish different interaction modes for the user, so the user clearly knows at all times which mode is active.” The issue also violates the guideline of the Gabbard framework that states, “emphasis should be placed on information relating to user tasks.”

5.4.2.1.3 Redesign Suggestion

Hix and Hartson (1993) give a redesign consideration to facilitate mode awareness in a graphical editor saying, “the shape of the cursor might change to indicate whether the editor is in the mode for creating circles or lines.” Building off this suggestion, modifying sword appearance to indicate current mode would supply a cognitive affordance and could improve usability in Crumbs.

5.4.2.2 Issue #2: Inconsistent use of direct manipulation in removing objects

5.4.2.2.1 Description

Crumbs has three classifications of objects:

- System persistent objects. Objects that cannot be removed from environment. This classification includes sliceplane creation object, crumb object, trashcan object, colormap #2 object, data volume object, and sword object.
- Single instance closable objects. These objects are used to manipulate Crumbs system parameter settings. There can be zero or one instance of the object in the environment. This classification includes the colormap object #1 and opacity object.
- Multiple instance disposable objects. Crumbs allows multiple instances of these types of objects. These objects can be removed permanently from the system. This classification includes the crumb object, sliceplane object, and clearbox object.

The issue concerns removing objects from the environment. Only the last two classifications of objects are removable so this issue does not impact system persistent objects. Crumbs provides two very different methods for removing objects from the system according to classification. Removing objects belonging to the multiple instance disposable object classification is throwing the object away, and Crumbs provides an appropriate direct manipulation metaphor of dragging the object to the trashcan object to execute this task. Removing objects belonging to the single instance closable object classification just temporarily closes the object, and Crumbs requires selection of a menu item to facilitate the task. Although there is a logical difference between deleting and closing objects, the strategy of providing a direct manipulation method for one and not the other caused errors. Upon adjusting the data volume's opacity settings, participants often wanted to remove the opacity object.

Task #	Description	Average Error Count	Average Completion Time
1.4	Remove the opacity tool from view.	0.5	17.15
3.5	Remove the opacity tool from view.	0.33	8.8

Table 7: Removing Objects Benchmark Tasks

Task 1.4 and task 3.5 (See Appendix D) were identical tasks that asked participants to remove the opacity tool from the environment. Two of the five participants attempted to remove the object by dragging it to the trashcan. This confusion was confirmed when one of the participants stated, “(h)ow do I remove it.” Task 1.4 was completed with an average error count of 0.5 errors and an average task completion time of 17.12 seconds. Task 3.5 was completed with an average error count of 0.33 errors and an average task completion time of 8.80 seconds. The improvement in both error count and task completion time from task 1.4 to task 3.5 can be associated with task learnability. Following the evaluation, one participant expressed that he thought Crumbs required a “more interactive interface.” He specifically used this instance of “colormaps cannot be deleted by dragging to the trash.” Another issue concerning conflicting use of direct manipulation was brought up by one of the participants following the evaluation. That participant addressed what he believed to be an inconsistency between the manipulation method used to control orientation of the data volume box and other objects in the system. He wanted the application to “allow the user to rotate an individual object like a clear box, because they are hard to orient properly.” When the evaluator asked the participant to expand on the suggestion, he revealed that he had a hard time using the “object in hand” metaphor provided by the application to orient objects. The data volume box, on the other hand, separates the orientation control from the translation control. The participant felt this method allowed for more precise control.

This suggestion is consistent with a VE-specific guideline from Gabbard that states, “multiple (integral) Degrees of Freedom (DOF) input is well-suited for coarse positioning tasks, but not for tasks which require precision.” The guidelines further suggest to “assess the extent to which DOFs are integrable and separable within the context of representative user tasks.” According to the participant, the three DOFs controlling translation should be mapped to one input device. The three DOF controlling rotations should also be mapped to one input device. However the rotation DOF and translation DOF are separable and should not be mapped to the same input device.

5.4.2.2.2 Guidelines

Providing contradictory methods of accomplishing similar tasks conflicts with a framework VE guideline from Gabbard that specifies, “(t)he look and feel of command presentation, be it visual, aural, or haptic, should be consistent within a single interface.” This issue is also addressed by Hix and Hartson (1993), as “similar things are expected to be done in similar ways.”

5.4.2.2.3 Redesign Suggestion

It is a common opinion that VE applications should use direct manipulation techniques whenever feasible. Therefore, to remove the direct manipulation inconsistency, Crumbs could provide a direct manipulation method of exiting or closing objects. Simply using the same metaphor of “dragging” the object to the trashcan object is not sufficient since this action symbolizes “deleting” the object in a way that is different from “closing” the object. One possibility is to provide a separate object, maybe a tool box object, which a user “drags” objects into if they want to close them.

5.4.2.3 Issue #3: Inconsistent method of selection

5.4.2.3.1 Description

Crumbs uses a consistent selection technique for most objects. However, several objects are complex objects that act as containers for other objects we will label “dependents.” This is similar to a toolbar in a WIMP interaction application. The toolbar is the complex object and the individual buttons are the dependent objects. GUIs facilitate selection as a “point and click” activity, using a mouse to position a cursor over the desired object and a mouse button to perform the selection. This same activity is used for selection of complex objects (i.e., windows, dialogs, toolbars, etc.) as well as dependent objects (i.e., buttons, scroll bars, elevators, etc.). Crumbs attempts to leverage this experience substituting a wand for the mouse, a sword for the cursor, and a wand button for a mouse button. However, GUIs use the same mouse button to perform the selection of both complex and dependent objects; whereas Crumbs uses two separate wand buttons. Complex objects are selected in Crumbs by placing the tip of the sword within the object’s boundaries and clicking the left wand button. Selection of dependent objects is facilitated by placing the tip of the sword in the general proximity of a

dependent object and clicking the middle wand button. Evaluators recorded several critical incidents that involved participants attempting to use the left wand button to select dependent objects. This use of different buttons for selecting complex and dependent objects is counter-intuitive due to its deviation from popular GUI interaction styles.

5.4.2.3.2 Guidelines

The first guideline in the Gabbard framework states, “(t)ake into account user experience.” Only following the GUI interaction style in particular cases caused participant confusion. VEs should either adopt a WIMP interaction technique fully or try to avoid it. If a participant is familiar with an interaction technique they will assume it is consistent. Requiring separate methods to perform similar tasks conflicts with the framework VE guideline that specifies, “(t)he look and feel of command presentation, be it visual, aural, or haptic, should be consistent within a single interface.” Hix and Hartson (1993). also guide designers that, “(s)imilar things are expected to be done in similar ways.” Although there is a difference between selection of a complex object and selection a dependent object, the difference is not distinct enough to warrant use of two separate selection methods.

5.4.2.3.3 Redesign Suggestion

“If something is done a certain way in an interface task, users expect the same thing to be done the same way throughout the rest of the interface.” (Hix and Hartson 1993). The selection method used to select dependent objects should mirror the technique used to select complex objects. Crumbs needs to adopt an interaction technique used in a common GUI WIMP application. Certain locations on a GUI toolbar are used specifically for manipulating the toolbar as a whole. A change in cursor appearance is used is a cognitive affordance, allowing users to know when they are in position to manipulate the object as a whole. A single button on the mouse is used for selection tasks. In a similar fashion, the left button on the wand should be used exclusively for selection of objects in Crumbs. The sword should change appearance to indicate when the tip is located in a complex object’s anchor position.

5.4.2.4 Issue #4: Awareness of current position in crumb placement task

5.4.2.4.1 Description

Participants consistently lost their place when dropping crumbs in the sperm tail data set. The sperm tail marking task required continual manipulation of clear box and data volume objects to improve data visualization attributes in order to facilitate accuracy of crumb placement. While attempting to manipulate these objects to locate more of the sperm tail, participants lost the location of the last crumb placed. Participants made statements such as, “I lost my place” and “I seem to have lost my way.” Because their attention was directed elsewhere, participants became unaware of the current location of the last crumb placed and spent a substantial amount of time trying to relocate it.

5.4.2.4.2 Guidelines

Continually keeping track of the previously placed crumb object in a structure that requires placement of a multitude of crumb objects in close proximity requires substantial memory management on users. This stresses users’ mental aptitudes. A framework guideline warns, “(t)ake into account users’ technical aptitudes.” Requiring users to keep track of crumb object positioning mentally also contrasts one of Nielsen’s ten heuristics that stresses “recognition rather than recall.”

5.4.2.4.3 Redesign Suggestion

Changing physical properties of the two crumb objects located at both ends of the fiber would allow users to rely on recognition rather than recall. Crumb objects could be enlarged or a different color could be used to act as a cognitive affordance.

5.4.2.5 Issue #5: Occlusion of 3D widgets

5.4.2.5.1 Description

Crumbs provides a set of interaction objects, “3D widgets”, to support altering data volume opacity and color settings. Because these interaction objects are 3D artifacts, they can be hidden from view by the data volume object or other interaction objects. This inability to locate an interaction object was unsettling to participants and led to multiple errors and prolonged task completion times, especially in instances where

the user specifically requests to activate the object with the desire to interact with it. If an interaction object did not appear the instance an “edit” command was initiated, most participants concluded that the “edit” command had failed and instinctively attempted to re-execute the command. This critical incident occurred during completion of task 1.3 when participants were instructed to manipulate the opacity tool. When participants chose to edit the opacity object, the object was originally located behind the data volume object, and participants were unaware of the current situation. All participants initially failed to locate the opacity object. Because the task encompassed both location and manipulation of the opacity object, the task completion and error counts cannot be fully attributed to the occlusion issue. One participant attributed lack of awareness of the opacity tool location as, “(t)he scene is too dark for me to see.” After some time, most participants realized the problem and took appropriate steps to correct it.

5.4.2.5.2 Guidelines

A guideline in the instructs VE application designers to focus on “attention to visual organization of the display.” It further specifies that “guidelines include minimizing overall and local density, and emphasizing information related to user tasks.”

5.4.2.5.3 Redesign Suggestion

Interaction objects must remain visible to facilitate use. Requiring users to locate an interaction object they had just requested in three-space prior to interacting with it proved to be counter-intuitive and confusing. Although this usability issue only occurred when participants scaled the data volume large enough to occlude the interaction objects, the lack of awareness in this situation consistently led to errors and frustration. Addition of an audio cue as feedback to the “edit opacity” or “edit colormap” commands would allow users to be aware of command execution. The interaction objects Crumbs provides to manipulate the data visualization box should remain visible when they are opened. Crumbs could facilitate this strategy by providing a persistent, always visible, complex object that is used as a container of the other objects. To visualize the data volume behind the container, it could be moved to a different location in the environment.

5.4.2.6 Issue #6: Inappropriate use or lack of audio annotations

5.4.2.6.1 Description

Crumbs effectively uses audio annotations as feedback for certain tasks such as crumb placement, but it provides sarcastic and uninformative audio annotations for other tasks and some tasks result in no audio annotation. Two of the five participants reported after the evaluation that audio cues were either “unprofessional” or that they “are not related to the actions that they are associated with.” The primary example of uninformative usage of audio annotation is a result of selecting a crumb object. Crumbs responds to this task with an “ouch” audio cue. “Ouch” is normally attributed as a sign that something is wrong and unless the user is aware of the task mapping this could cause unnecessary concern. The primary example of sarcastic audio usage is a result of manipulating the opacity object. This task results in a “good luck” audio cue. One participant, when commenting about the “good luck” annotation, said, “(t)his is really annoying.” A good example of a task that lacked an audio cue is saving a group of crumbs objects. One participant specifically attributed this lack of an audio cue as causing confusion.

5.4.2.6.2 Guidelines

This issue is addressed by two framework guidelines. Both the “ouch” and “good luck” messages conflict with the framework guideline that states, “(s)ystem messages should be worded in a clear, constructive manner so as to encourage user engagement (as opposed to user alienation).” Another guideline states, “(l)anguage and labeling for commands should clearly and concisely reflect meaning.”

5.4.2.6.3 Redesign Suggestion

Provide informative audio annotations and replies for all appropriate user tasks. Replace the sarcastic “good luck” with an appropriate “manipulating opacity” cue. Replace the “ouch” cue with a sound that is unique to crumb selection task, but cannot be misconstrued as indication of an error.

5.4.2.7 Issue #7: Arm movement facilitating cascading menus

5.4.2.7.1 Description

In Crumbs, transitioning from a cascading menu item to its sub-menu items requires lateral movement of the sword tip. If, during the course of traversing from parent menu item to sub-menu item, the sword tip moves vertically outside the desired parent menu item prior to moving into the desired sub-menu, the sub-menu disappears and the new menu item corresponding to current sword tip location is highlighted. This behavior is consistent with WIMP cascading menu interactions. The issue arises due to the physical means utilized to accomplish this task. A WIMP application requires manipulation of a mouse to control a cursor for this task. In most cases, movement of the mouse is minimal and users execute the task using straight lateral movements of the wrist. Evaluators noted that Crumbs' participants attempted to perform this motion using a gross motor movement of the arm instead of their wrist. Natural lateral arm displacement is for only a limited distance. Once this distance is reached, the motion deteriorates into an arcing motion. Due to this arcing motion, one participant was unable to execute a menu item located on a sub-menu on the first try. This issue is exacerbated by the length of the menu item label. Crumbs adjusts width of a menu dynamically to allow encapsulation of the longest length menu command. Menus with longer widths require more arm movement and increase the likelihood of a usability issue. This issue becomes especially problematic for users with shorter than average arm lengths.

5.4.2.7.2 Guidelines

This issue conflicts with the framework guideline that instructs, “accommodate natural, unforced interaction for users of varied age, gender, stature, and size” and “input devices should make use of user physical constraints and affordances.”

5.4.2.7.3 Redesign Suggestion

One course of action is to design menu item labels that are limited in size yet still clearly and concisely reflect meaning. Another suggestion is to document the preferred method of manipulating a menu laterally by rotating the wrist. This physical action allows for a greater distance of straight vertical transitioning.

5.4.2.8 Issue #8: Menu truncation due to sword location

5.4.2.8.1 Description

The menu system is programmed as a pop-up menu that appears at the location of the sword tip. Crumbs truncates the menu if the sword is located close to the physical limitations of the presentation component, in our case the top of the CAVE walls and the exterior edges of the side walls. This requires users to:

1. Release the wand button to remove the menu,
2. Position the wand in a position that truncation will not occur, and
3. Click the right wand button to bring the menu up.

During the evaluation, one participant attempted to utilize the menu system with the sword tip located in close proximity to the top of the front wall. The menu was displayed according to positioning of the sword without regard to the sword location. Consequently the top half of the menu was severed by the top of the front wall. Although this occurred only once during evaluation it appeared to cause enough frustration to warrant inclusion as a usability issue.

5.4.2.8.2 Guidelines

Nielsen's "Error Prevention" heuristic is applicable. At the time of the button click, Crumbs can calculate sword tip location and prevent menu truncation.

5.4.2.8.3 Redesign Suggestion

Because this issue is the result of the physical limitations of a four-wall CAVE, and six wall CAVEs would not demonstrate the same usability issue, redesigning Crumbs to eliminate this usability issue may not be an efficient appropriation of resources. However, if four wall CAVEs are the primary physical architecture, then Crumbs needs to detect when a menu will be severed by a wall boundary and adjust the menu location accordingly to show the entire menu. WIMP applications solve this issue in exactly this manner.

5.4.2.9 Issue #9: Different crumb modifications use similar interactions

5.4.2.9.1 Description

Other than placing crumbs in the wrong location, the primary error observed involving crumb manipulation was a creation/selection issue, i.e. creating a crumb while intending to select an existing crumb or vice versa. Individual crumb object manipulation for both selection and creation is accomplished using the middle wand button. Crumbs decides which action (creation or selection) the user is attempting to perform depending on the location of the sword tip. If the sword tip is located in close proximity to an already existing crumb object, then that crumb object is selected. If no crumb object exists within a certain distance, then a new crumb object is created. This dual task mapping proved problematic in the following two instances:

1. Attempting to create a crumb in a dataset consisting of previously marked densely positioned crumbs.
2. Attempting to select a crumb object but failing to position the tip of the wand within the acceptable distance of that particular crumb object. This resulted in the undesirable creation of a new crumb object that had to be deleted prior to continuing the attempt to select the crumb object.

This issue did not prove to be a problem in the spine dataset because the task required participants to manipulate crumbs located a substantial distance apart. However, in the sperm tail dataset, crumbs were located in closer proximity, and this issue caused multiple creation/selection errors for two thirds of the participants.

Task #	Description	Average Error Count	Average Completion Time
3.8	Select the last marked crumb.	1	10.94

Table 8: Crumb Modification Benchmark Task

In particular, task 3.8 (see Appendix D) requested participants to select an already positioned crumb. Although this task did not take long to complete (10.94 seconds), the task resulted in an average error count of one error.

5.4.2.9.2 Guidelines

In reference to using modes in an application, Hix and Hartson (1993) state, “(w)hen it is used, the designer should be careful to distinguish different interaction modes for the user, so the user clearly knows at all times which mode is active.” The

issue also violates the Gabbard guideline that states, “emphasis should be placed on information relating to user tasks.”

5.4.2.9.3 Redesign Suggestion

Because Crumbs makes use of the left wand button as a selection mechanism, then the selection of all objects in the VE should follow the same strategy. Selection of crumb objects should also function in a consistent manner. This method would eliminate the usability issues that occur when trying to create new crumb objects and selection of existing crumb objects. Crumb object creation should be facilitated using the middle wand button; while crumb object selection should be facilitated using the left wand button.

5.4.2.10 Issue #10: Awareness of colormap object box value

5.4.2.10.1 Description

The values of the colormap object are controlled by a set of nine cube shaped dependent objects that can be selected and moved. Each cube represents a different density value. Moving a cube within the colormap object alters the color of the specific density in the dataset the cube represents. However, cubes are identical in shape and color and lack any distinguishable unique cognitive affordance. The only distinguishable attribute is an audio feedback that gives the number of the cube from 1 to 9. Crumbs does provide the persistent auxiliary colormap object that provides a mapping between structure density and color settings. Nonetheless, the connection between the two colormap objects was not realized by any of the participants, and Crumbs provides no indication that the two are related. Therefore, participants resorted to selecting cubes randomly, and attempting to locate the one that represents the desired density value.

Task #	Description	Average Error Count	Average Completion Time
1.12	Adjust the colormap so the most dense values of the dataset is red and the least is yellow.	3	92.66

Table 9: Colormap Awareness Benchmark Task

Task 1.12 (see Appendix D) requested participants to manipulate colormap settings. This task resulted in a high error count of three, indicating the degree of difficulty in completing the task. Error counts would have been higher, but several participants displayed signs of confusion and finally gave up prior to task completion. The task also resulted in a longer than expected average task completion time of 92.7 seconds. When asked to manipulate the colormap object, participants used phrases such as, “I’m not sure what the boxes represent”, “I have no idea where these things are”, and “(n)ot sure I did that right”. Continual use of the phrase “not sure” illustrates the lack of intuitiveness of the object itself. One participant even expressed that he did not know how or even why to use the colormap. Participants were asked to rate the usability of the colormap object on question 5 of the questionnaire. The poor average score of 1.5 (out of 4) is consistent with participants’ average error count and task completion times in attempting to manipulate the colormap object.

5.4.2.10.2 Guidelines

One guideline in Gabbard’s framework, suggests that language and labeling for commands should clearly and concisely reflect meaning. The audio cue of stating the cube “number” does not directly map to cube functionality. Also, the lack of any visual identifier fails to suggest any unique characteristic attributed to each cube. Another guideline suggests that a VE should exploit real-world experience by mapping desired functionality to everyday items. This novel use of a color cube for mapping colors to density regions is not something with which most users are familiar.

5.4.2.10.3 Redesign Suggestion

To facilitate the selection of the desired value widget, a unique visual cognitive affordance could be added to each value widget. For simplicity, the number used in the audio annotation could be placed to represent the cube on each side of the cube. One participant suggested including textual interface representations of objects. Participants wanted the ability to look at data values that corresponded to visual representations of objects. For example, users did not want to use the color cube to specify colors for certain densities, they should have the choice of using a more conventional textual based interface. This approach could potentially promote use of pre-existing experience

possessed by users, prior to working with the application. This theme could be expanded to encompass use of a textual interface for the opacity object, data volume density at specific crumbs, etc.

5.4.2.11 Issue #11: Use of color box metaphor in colormap object

5.4.2.11.1 Description

Crumbs designers chose to leverage a color cube metaphor already used in VR literature to demonstrate the spectrum in colors associated with the RGB color scheme for the colormap object. Placing one of the individual cubes within the color box assigns the specific density value associated with that cube to the color assigned to the specified color cube location. However, unless introduced, the color box metaphor was not intuitive and therefore the colormap object represented a substantial usability issue for novice users. When asked to manipulate the colormap object, participants used phrases such as, “I haven’t used the colormap before”, “(n)ot sure what the axes are”, and “(n)ot sure I did that right.” One of the participants admitted to not being at all familiar with the color cube paradigm that the colormap object attempts to leverage.

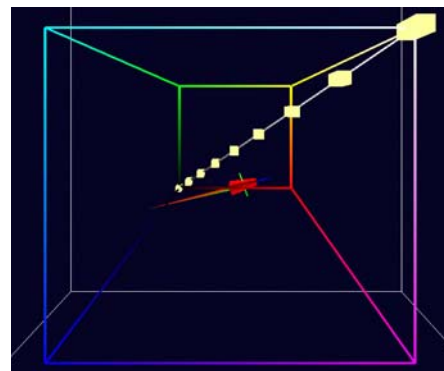


Figure 12: Colormap Object

5.4.2.11.2 Guidelines

Using an interaction metaphor that is not widely known places added emphasis on Crumbs experience for ease of use. This emphasis conflicts with the *Framework* guideline that suggests VE designers take into account user experience and design interfaces that support both expert and novice users.

5.4.2.11.3 Redesign Suggestion

Color for independent densities could be manipulated by allowing users to select a particular density value, and then give numeric values for the red, green and blue components of the RGB scale using either a slider widget or some other form of numeric

entry. The color of the particular object that represents a specified density could be adjusted to mimic the color assigned.

5.4.2.12 Issue #12: Two stepped load colormap values interaction

5.4.2.12.1 Description

Crumbs allows users to save predefined colormap and opacity object values in order to allow the saved values to be reloaded at a later time. Both the opacity object values and colormap object values are saved using the “Save Colormap” menu item command. It would seem logical then, that in order to load these values into the environment, the user would only have to execute the “Load Colormap” menu item command.

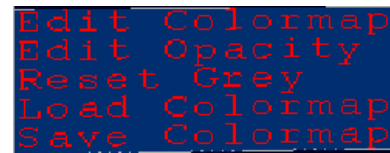


Figure 13: Color Map Sub-menu

However, this is not the case. Loading predefined colormap and opacity object values is a two-step process. First, the user must execute a “Load Colormap.” Second, the user must then also perform another task to visualize the loaded values in the dataset. To make the loaded colormap object values take effect, the user must perform an “Edit Colormap” menu item command.

Task #	Description	Average Error Count	Average Completion Time
1.2	Load the predefined color map to help visualize the spinal cord.	1.25	12.045

Table 10: Load Colormap Benchmark Task

Task 1.2 (see appendix D) specifically instructed participants to load predefined colormap values. None of the participants was aware of the two-step process. Instead they only executed the “Load Colormap” command and considered the task completed. None of the participants realized they had to execute an “Edit Colormap” command without assistance from one of the evaluators. This issue resulted in an average task error count of 1.25 errors and average task completion time of 12.05 seconds. These values are not truly indicative of the usability issue because the most frequent error occurred when a participant thought they were done prior to finishing the task correctly.

5.4.2.12.2 Guidelines

Hix and Hartson (1993) provide a guideline to optimize user operations. They further elaborate, “(d)esigners should strive toward the most effect for the least user effort.”

5.4.2.12.3 Redesign Suggestion

Redesign Crumbs so that the predefined colormap and opacity object values are applied to the dataset directly following a “Load Colormap” menu item command.

5.4.2.13 Issue #13: Opacity object intuitiveness

5.4.2.13.1 Description

The main problem experienced by participants using the opacity object was not physical manipulation, but rather understanding how to manipulate the object to achieve the desired goal. The opacity object is represented as a line graph with the x-axis representing dataset density and the y-axis representing opacity or translucency. Users manipulate opacity object values by assigning points on the line graph to a specific coordinate within the x-axis and y-axis. Most participants were very hesitant about using the opacity object. When the participants did manipulate the opacity object, it appeared that they were “groping” for a strategy to manipulate the object properly. Manipulating the opacity object appeared to be an exercise of “fishing” for the right settings in hopes of “stumbling” onto desired results.

Task #	Description	Average Error Count	Average Completion Time
1.3	Retrace the opacity tool to locate the spinal cord.	2.25	73.2
3.4	Manipulate the opacity tool to locate the sperm tail. (Novice - provide a graph of example opacity settings)	0.75	105.09

Table 11: Opacity Object Benchmark Task

Task 1.3 (see Appendix D) was the initial introduction of the opacity tool into the evaluation and had one of the highest average error counts of all tasks in the study at

2.25. This task also had an average task completion time of 73.2 seconds. Task 1.3 asked participants to manipulate the opacity object by first initiating it and then re-tracing the curve that already existed on the tool. The other task in the evaluation that measured opacity object usability was Task 3.4 (see Appendix D). This task only had an average error count of 0.75, but its average task completion time was 105.09 seconds.

Task 3.4 was more complicated than Task 1.3. This task required participants to manipulate the opacity tool in an attempt to locate a curve that would highlight desired segments of the data volume. This difference accounts for the increase in time from the first task to the second. Decrease in error counts could be associated with participants' familiarity with the tool from previous use or with an inability of the evaluators to determine if the resulting curve was appropriate or just the result of the participant giving up. Another problem participants experienced with the opacity tool was their apparent lack of experience and knowledge of the tool. One participant confirmed this lack of experience and knowledge by asking, "(h)ow do I manipulate the opacity tool." One participant thought that the "opacity is good enough" and did not even attempt to manipulate the object. This participant later conveyed to the evaluators that the developer of Crumbs usually assists on use of the opacity object. Question 4 on the questionnaire directly asked participants to rate the ease of use of the opacity object. Participants rated usability of the opacity object as a 2.25 on a scale of 1 to 4.

5.4.2.13.2 Guidelines

The inability of participants to use the opacity object effectively suggests that it is an interaction object that is best suited for "advanced" users, or more likely that is poorly designed. However, Crumbs does not provide any other method of modifying the opacity of an object according to density values. This issue is similar to the colormap object issue discussed above (in Section 5.4.2.11) and conflicts with the *Framework* guideline that suggests VE designers take into account user experience and design interfaces that support both expert and novice users.

5.4.2.13.3 Redesign Suggestion

Although the opacity object is a substantial usability issue, we do not currently have any suggestions for a more appropriate object for providing the range of

functionality that the opacity object provides. It is evident that a thorough consideration of redesign and/or tutorial on how to manipulate the object is necessary. Crumbs designers should revisit designing a more intuitive method of providing opacity functionality.

5.4.2.14 Issue #14: Opacity object loading two stepped process

5.4.2.14.1 Description

A common error was participants' inability to utilize values loaded from a previously saved opacity object. Participants were not aware that the curve in the opacity object required re-tracing to facilitate environmental changes. When a previously saved colormap object is loaded, the previously saved opacity object settings are also loaded. However, for settings to take effect, the curve on the opacity tool must be retraced. The evaluation included a task to retrace the opacity object curve to familiarize participants with manipulation of the tool and to setup the environment with values that facilitated visualization of the spine dataset. However, requiring participants to retrace the curve proved to be a task requiring evaluators to provide a detailed explanation. Also, retracing the line resulted in errors. For the most part, the re-traced line was a close representative to the original line, but none of the participants could retrace the line exactly as it was loaded.

5.4.2.14.2 Guidelines

Hix and Hartson (1993) provide a guideline to optimize user operations. They further elaborate, “(d)esigners should strive toward the most effect for the least user effort.”

5.4.2.14.3 Redesign Suggestion

Crumbs should automatically modify the appearance of the data volume according to the reloaded opacity object values when the colormap object is reloaded.

5.4.2.15 Issue #15: Usefulness of Sonification to Tasks

5.4.2.15.1 Description

The benefit of sonification was not evident in the sperm tail marking tasks. The participants that completed the task did not appear to rely on sonification at all. Sonification was designed in Crumbs as a three part activity corresponding to the creation and positioning of a crumb. The interlude is designed to signify creation of the crumb. Following the interlude, sonification changes according to the region density currently indicated by the sword. Finally, once the crumb had been “dropped” at a location, the sonification concludes with a postlude. For a majority of crumbs created, both participants finished positioning an individual crumb prior to the sonification completing the interlude. Therefore, they were unable to use sonification for any benefits. One of these participants went so far as to voice their opinion of the sonification saying, “(p)ersonally, for what it is worth, I really hate the music”. The same participant restated this opinion following the evaluation saying, “I do not like the sonification.” The participant that made this statement proved during the evaluation to be one of the more skillful users of crumb object manipulation. A different participant that did not finish the task attempted to use sonification to assist in crumb placement, but expressed that he was unaware of the “target sound” and that the “sound is the same everywhere.”

5.4.2.15.2 Guidelines

Use of sonification to accommodate visual feedback and assist in visualization of data density supported the *Framework* guideline to supply multi-modal interaction information. However, evaluation results suggested that sonification was not an aid to participants. Participants were not able to map the sonification to its purpose, or the amount of time necessary to utilize the sonification mechanism did not match the amount of time required to complete the task.

5.4.2.15.3 Redesign Suggestion

Sonification could be improved by removing or shortening the duration of the interlude and postlude. Most crumb marking tasks were short in duration. The interlude and postlude proved unnecessary and confusing.

5.4.2.16 Issue #16: Precision of crumb placement

5.4.2.16.1 Description

Other than data volume visualization, crumb object manipulation is the main task in the application. Crumbs lacks a quantitative method of determining correctness of crumb placement. Error counts recorded on crumb placement tasks were limited to interaction errors while attempting to place individual crumbs and the number of crumbs placed in the vertebrae task. Neither of these error counts provided a quantitative measure of the ability to mark points in the data volume correctly.

Precise crumb placement often appeared to be a test of estimation rather than precision. Participants supported this observation with their statements completing crumb placement tasks. Participants used comments such as “(a)llright, that is good enough,” “I think I’m done” and “(l)et’s just hope I’m done” to signal their completion of crumb marking tasks. These comments demonstrated the lack of preciseness and closure of the task itself. One participant suggested following the evaluation that Crumbs should provide a method to locate dense structures automatically. This suggestion related to the ability to, as one participant called it, “(s)nap to high density data areas”; i.e., the ability to allow Crumbs to locate dense data volume areas independently for the user.

5.4.2.16.2 Guidelines

This issue violates several *framework* guidelines. The first suggests that the environment “present domain-specific data in a clear, unobtrusive manner such that the information is tightly coupled to the environment and vice-versa.” This issue also violates another guideline that suggests to “emphasize information related to user task.”

5.4.2.16.3 Redesign Suggestion

Crumbs should utilize a textual object that provides the density value of a particular crumb or position of the sword tip. Another improvement would be to supply a way to place a crumb in a high or low density point. Crumbs could also provide a sizable sphere to specify candidate portions of the data volume to use for the activity and a means for the user performing crumb placement.

5.4.3 Conclusions

Evaluators were surprised at the amount of trouble and frustration the participants demonstrated during the evaluation. Tasks were chosen by Crumbs designers as demonstrative tasks and participants were chosen by the designers as representative of “expert” users. Evaluators expected participants to have little trouble completing tasks and to demonstrate considerable command over interaction metaphors and controls in the environment. This was not the case. Few participants demonstrated confidence when maneuvering the more advanced interaction objects (e.g., colormap and opacity). During most of the evaluation, participants demonstrated tendencies more often attributable to novice users during formative evaluations such as nervousness and lack of key application knowledge. The number and type of usability issues resulting from this study suggest that the participants had a lack of familiarity with the system. We believe this apparent unfamiliarity can be traced to the limited number of people who were using the application for more than demonstration purposes. Crumbs is a unique application with great applicable potential. However, as evident in this study, Crumbs developers need to focus on improving usability in order to expand the population of candidate users.

6 Assessment of Evaluation Process

We analyzed the success of our usability evaluation methodology by comparing results of the guidelines-based inspection to results of the formative evaluation.

- We expected the methodology to benefit from both the guidelines-based inspection and formative evaluation because each was necessary and each individual evaluation would discover unique usability issues. This proved to be a correct assumption.
- We expected that usability issues that did not glaringly violate any guideline should be further assessed. This would provide a link between the two usability evaluation methods and allow them to share information. We observed this coupling of evaluation methodologies to be productive.
- We also expected that using results of the guidelines-based inspection as an input mechanism to the creation of benchmark tasks for the formative evaluation could prove to be an effective estimation tool of the severity of usability issues. This estimation proved effective for usability issues that were categorized in the inspection process but not redesigned by the application creators prior to the formative evaluation.

6.1 Unique Usability Issues

Table 5: Usability Issues Unique to Guidelines-Based Inspection lists a number of usability issues found in the guidelines-based inspection that were not found during formative evaluation.

Usability Issues Unique to Guidelines-Based Inspection
Placing crumb objects is a stackable task, and Crumbs should consider a quick method of unstacking (i.e., undo).
The only method of feedback available in the immersive environment for the “Arc Length” execution is audio. This is not functional for hearing impaired individuals or hardware setups that do not have audio equipment. The number is printed to the unix window on the workstation, but should be included into the CAVE environment.
Whenever colormap or opacity objects are active, any attempt to use the middle mouse button (no matter where in the environment a user is pointing) affects the colormap and

opacity objects. This is not consistent with the direct manipulation interaction metaphor that is used
Crumbs permits the user to attempt to select several of its menu items when that menu item is not possible. For example, the user can select the “Toggle Spline” menu item when there are less than two crumbs placed.

Table 12: Usability Issues Unique to Guidelines-Based Inspection

The first unique usability issue listed in Table 5 dealt with classification of a set of tasks as “stackable” by the usability specialist. Results of the guidelines-based inspection suggested that the formative evaluation use a scenario to ascertain whether an “undo” feature was desired for crumb placement. As a result, we created several tasks involving sequential creation and deletion of crumb objects; however, participants did not independently provide any comments or suggestions that an “undo” feature was required or useful. We expect the absence of comments results from the lack of standardization of VE interaction styles. Users of GUI applications have been exposed to GUI applications that support “undo” and therefore have been conditioned to expect it. Most often, VE users are attempting to familiarize themselves with the interaction style unique to the specific application and therefore are not used to auxiliary functionality such as “undo” unless they have experienced it in another VE application. Also, we simply created benchmark tasks; we did not attempt to extract any qualitative information from participants that was not given freely. The lack of an “undo” feature might have been considered a usability issue if participants were asked open-ended questions such as, “(i)s there any functionality that might have made that task easier?” This lack of probing may have been a weakness of our methodology, in retrospect we should have elicited this kind of information from participants.

The second usability issue in Table 5 regards colormap and opacity objects. It appears that the intended use of these objects is as modal objects. In other words, once the colormap or opacity object is utilized, it remains in control of the environment until it has been removed. However, Crumbs has no cognitive affordances to signal this intended use, nor is there any mention in Crumbs’ documentation. When we examined Crumbs void of any pre-determined tasks, we found ourselves attempting to manipulate

other objects prior to closing the opacity or colormap object. This resulted in unexpected results. Although conveyed as a usability issue to Crumbs designers, they chose not to act upon the suggestion prior to formative evaluation. Because tasks were not created to misuse the interface, the usability specialists simply provided benchmark tasks that could possibly lead to this particular usability issue. We believe that none of the participants experienced this usability issue because they simply did not encounter these circumstances when completing the tasks.

The third issue listed in Table 5 discusses Crumbs lack of “graying out” current disallowed menu items. Again, this issue was discovered when we performed the Crumbs inspection without a pre-defined set of scenarios or tasks. Because a major form of Crumbs interaction utilizes the menu system, we completed a comprehensive evaluation of the menu system and tried combinations of menu items that probably would not be performed following a set of tasks. Although Crumbs developers did not act on this usability issue prior to formative evaluation, we chose not to include tasks that would require participants to perform tasks out of order. Therefore, there were no tasks that directly tested whether the lack of “graying out” menu items was a usability issue.

The final usability issue unique to the guidelines-based inspection was audio feedback for menu items. We thought that an audio annotation providing current length of the arc should be supplemented with a visual representation of the numeric value to support users with hearing problems and installations that did not support audio. Several usability guidelines ensure interfaces are evaluated for multi-modal interaction. Because our formative evaluation population did not have a hearing impaired individual, the problem was not discovered during formative evaluation.

Table 6 lists a number of usability issues found during formative evaluation that were not found as usability issues during guidelines-based inspection. There are a number of reasons why usability issues discovered using formative evaluations might be overlooked in a guidelines-based inspection.

- Formative evaluations follow a set of predefined tasks of a domain-specific representative scenario. Therefore, participants are probably completing more in-depth tasks than the free explorative tasks often performed during guidelines-based inspection.

- Whereas VE-specific usability specialists have knowledge of VE interaction metaphors, formative evaluation participants have domain-specific knowledge and are more aware of an application's lack of important tasks.

During formative evaluation, we observed that several usability issues were a direct result of participants completing very involved structure marking tasks. These usability issues included awareness of middle wand mode, awareness of current position, and similar interactions for different crumb modifications. Since the usability specialist chose not to complete a complex marking task these usability issues were not uncovered during guidelines-based inspection.

We observed no reasons why formative evaluation was unique in identifying usability issues concerning the occlusion of 3D widgets and menu truncation. Our opinion is that these usability issues were discovered as a result of simply including more participants. Widget occlusion was a result of benchmark task order and menu truncation was a result of the random placement of the sword location by multiple users when attempting to utilize the menu system.

The two-stepped colormap interaction problem involved expert use of the colormap object. Because guidelines-based inspection resulted in a recommendation to further evaluate the colormap object interface, we included benchmark tasks in the formative evaluation to test the colormap object comprehensively. Including these benchmark tasks led to colormap usability issue that was not obvious during guidelines-based inspection.

Usability Issues Unique to the Formative Evaluation
Awareness of middle wand button for scale/mark mode
Awareness of current position in crumb object placement task
Occlusion of 3D widgets
Menu truncation due to sword location
Different crumb modification use similar interactions
Two-stepped load colormap object interaction

Table 13: Usability Issues Unique to the Formative Evaluation

6.2 Usability Issues Requiring Further Investigation

Table 7 lists three usability issues requiring further evaluation after the guidelines-based inspection. Of the three, two resulted from usability issues identified during formative evaluation. The one that did not prove to be a usability issue, categorizing crumb placement as a stackable task, was discussed previously in Section 6.1. This section will focus on the other two.

Crumbs attempts to leverage interaction techniques similar to those of GUIs for most of its interface. However, interaction metaphors used to manipulate the colorbox and opacity objects do not follow that strategy. Our opinion is that unique interaction metaphors used to facilitate these two objects did not completely violate any guidelines, but could cause usability issues and should be further investigated. This opinion allowed us to leverage our experience collected during guidelines based inspection and use it to create formative evaluation benchmark tasks. This use of guidelines based opinions to create specific formative evaluation tasks is the additional step we believe is essential to coupling usability inspections and formative evaluations to evaluate VEs. Usability specialists use their in-depth VE knowledge and experience to create tasks for formative evaluation. This step also preserves the opinions of those performing the guidelines-based inspection. In historical coupling of these two usability evaluation methodologies, usability inspection was used simply to remove blatant usability issues prior to the more expensive formative evaluation. However, there was no mechanism to transfer usability concerns that were not directly marked as usability issues forward to formative evaluation.

Inclusion of specifically designed benchmark tasks to further evaluate the colorbox object's interface intuitiveness, consequently led to two separate usability issues concerning the colorbox object:

1. Participants were unaware of the values that the interface value widgets represented. (See discussion of formative evaluation usability issue #10 in Section 5.4.2.10 for detailed discussion.)
2. The color cube metaphor was not effective. (See discussion of formative evaluation usability issue #11 in Section 5.4.2.11 for detailed discussion.)

We were not quite as successful in discovering opacity object usability issues. Benchmark tasks requiring users to interact with this object supported the assumption that it was a usability issue, but we were not able to pinpoint exactly what was causing the problem. The opacity object interface is simply hard to manipulate and requires substantial experience for efficient use.

Inspection Issue	Formative Eval. Issue
Use of the opacity object is not initially intuitive and requires training for proper use.	Opacity object intuitiveness.
Placing crumb objects is a stackable task and Crumbs should consider a quick method of unstacking (i.e. undo).	See Section 6.1 on unique usability issues.
Use of the colorbox object is not initially intuitive and requires training for proper use.	Awareness of colormap object box value and use of colorbox metaphor in colormap object.

Table 14: Usability Issues Marked for Further Evaluation

We do not suggest that these usability issues would have escaped detection using the traditional method of task analysis for driving formative evaluation benchmark task creation. However, these results do support the benefit of leveraging the usability specialist's experience to pinpoint possible usability issue areas and then using this information to create benchmark tasks that fully evaluate these potential problems.

6.3 Assessing the Impact of Usability Issues Specified in the Inspection

We discovered that another benefit of our VE usability evaluation methodology is the use formative evaluation to assess usability issues that were discovered during the guidelines-based inspection, but not acted upon by the Crumbs developers. When it is unclear to application developers whether a usability issue is severe enough to warrant the effort and resources for a redesign, then benchmark tasks can assess the extent of the usability issue during formative evaluation. Crumbs developers decided not to commit the time and resources to correct many usability issues reported during guidelines-based inspection. So during formative evaluation we had participants perform tasks to give us more insight as to the severity of these usability issues. Four of these usability issues

were confirmed during formative evaluation. We did not collect any data to support the others as usability issues.

During formative evaluation we were aware of the potential for these usability issues and were prepared to collect qualitative and quantitative data to support the claim. We also created questions on the questionnaire to evaluate certain potential usability issues further. The four potential usability issues confirmed using formative evaluation were:

1. Use of inappropriate audio annotations.
2. Inconsistent use of direct manipulation.
3. Unnatural arm movement to facilitate cascading menus.
4. Inconsistent selection method.

7 Concept Book Usability Engineering Methodology

As we discussed in the introduction (Section 1) of this thesis, a focus of our research at Virginia Tech is to cultivate powerful and efficient VE usability engineering methodologies using an evolutionary approach. Webster (1983) defines evolution as “an unfolding process of development or change” and “a movement that is part of a series.” Therefore evolution is not a bounded set of events. Rather, it is a never-ending process of adaptation. Attempting to facilitate an evolutionary approach leads to a cyclic strategy similar to the one discussed in the introduction of this thesis. We believe the success of our original proposed usability engineering methodology confirmed that an evolutionary approach was a valid means to create usability engineering methodologies that support VEs. We then turned our focus from the results of the initial proposed VE usability engineering methodology to the next iteration of the evolutionary cycle. In this chapter we discuss the weaknesses we observed in the proposed usability engineering methodology and propose new adaptations to the methodology based on our experiences.

7.1 Weaknesses of the Initial Proposed Usability Methodology

Several aspects of the methodology warranted improvement. These improvements involved both modifications to the individual guidelines-based inspection methodology and formative evaluation methodology, as well as strengthening coordination between the two. It was the unique coupling of the guidelines-based inspection and formative evaluation methodologies that we observed took full advantage of Gabbard’s *Framework of Usability Characteristics in VEs* and all the usability specialists involved. Therefore, in this cycle, we decided to introduce adaptations that would strengthen coordination between the two. The following sections address first the weaknesses of the individual usability engineering methodologies of guideline-based evaluation and formative evaluation performed in the previous cycle and then discuss the weaknesses observed in sharing information between the two.

7.1.1 Guideline-based Inspection

Because the guidelines-based inspection made use of the *Framework of Usability Characteristics in VEs*, there was very little need of improvement observed as far as

coverage of VE-specific usability issues. However, the usability specialist did observe that several basic HCI guidelines or heuristics were not included in the *Framework's* guideline list, such as promotion of direct manipulation interaction metaphors and correct use of foreground and background colors. This absence caused some hesitation on the part of usability specialists who encountered a usability issue but were unable to assign a specific guideline that was violated. Usability specialists are familiar with traditional usability heuristics and guidelines and during the course of their career append their own set of heuristics gained by experience. Furthermore, many traditional guidelines and heuristics apply for VEs. Therefore restricting usability specialists to simply assuring that the VE adheres to the guidelines provided in the *Framework of Usability Characteristics in VEs* does not take full advantage of the usability specialists' expertise and is not in the best interest of the VE.

Most of the observed limitations concerning our guidelines-based inspection revolved around the way in which it was executed. The most critical observed fault of the guidelines-based inspection methodology was the length of time it required a single usability specialist to complete. This observation is a historical fault of any guidelines-based usability engineering strategy, but one that we must find methods of circumnavigating if we are to reach our goal of not only producing powerful VE usability engineering methodologies but also efficient ones. Several individual issues increased the length of time required to complete the inspection. First, we attempted to conduct a guidelines-based inspection strategy similar to a heuristic evaluation; however, instead of ten heuristics we employed 195 guidelines. Because the set of guidelines was new to the usability specialist, he had to keep a copy of the guidelines with them throughout the inspection process. We attempted to facilitate this by breaking the guidelines into sections and asking the usability specialist to inspect the corresponding aspects of Crumbs using the correlating guidelines. However, this accommodation involved a single usability specialist conducting nineteen separate guideline-based inspections and demanded a substantial amount of the usability specialist's time.

Another cause of the large demand for the usability specialist's time involved the confusion surrounding the domain-specific tasks and the correct use of unique interaction metaphors. The usability specialist who performed the guidelines-based inspection was

not given any guidance on how to efficiently use the application nor was he introduced to a set of representative tasks. Therefore, the usability specialist attempted to create his own set of tasks dynamically and evaluated how well Crumbs facilitated those tasks. He inspected each environment object and evaluated how usable they were individually. Except for obvious task coordination, the usability specialists did not have any idea of the normal flow of tasks and therefore were very restricted in being able to evaluate task flow. The usability specialist also had to spend a substantial amount of time with the unique interaction metaphors of the colormap object and opacity object in determining their purpose and how to manipulate them.

7.1.2 Formative Evaluation

One key finding of the formative evaluation was distinguishing the difference between collecting pertinent data on specific physical tasks and documenting a participant's cognitive state. We had hypothesized prior to completing the formative evaluation that the usability of tasks such as modifying object attributes and object selection could be evaluated using traditional objective measures of task completion time and error rate. However, quantitative measures are only effective at either indicating "what" is happening or comparing two results. Quantitative measures do not help to answer "why" or "how" something is happening. As usability specialists, we were not happy simply reporting that a usability issue existed, we were much more interested in interpreting why a usability issue existed. For this task we needed to collect subjective or qualitative data. During the course of the evaluation, participants appeared apprehensive about utilizing a concurrent verbal protocol even though they had been instructed to do so by the usability specialists. This often led to the quantitative indication that a usability issue existed without an appropriate amount of qualitative data for interpreting the phenomena causing the usability issue (e.g., the opacity object intuitiveness issue).

Our limited attempt at providing quantitative data concerning the subjective concept presence was not comprehensive enough to support any definitive claim about Crumbs' ability to support presence. We had hypothesized that qualitative data would be collected concerning presence during the concurrent verbal protocol, but as we previously mentioned, participants did not initiate many comments. As discussed earlier, we also

added a question on the follow-up questionnaire that directly asked the participants subjective opinion of Crumbs' ability to support presence. This single quantitative question alone clearly was not a thorough assessment of presence, and future work will focus on developing a strategy for strengthening quantitative and qualitative measures of presence.

Another concept we inappropriately evaluated quantitatively and qualitatively was situational awareness. Since Crumbs is a single-user event driven VE, void of any intelligent agents, situational awareness did not play as large a role in this evaluation as it will in multi-user dynamic VEs. However, the concept of situational awareness encompasses the traditional HCI concept of interpreting the current state of the environment. Additionally, the results of the formative evaluation categorize several usability issues that were attributed to participant's inability to interpret the current situation. Therefore, it was our experience that situational awareness can be measured both quantitatively and qualitatively, and benchmark tasks should be created to ensure the participants are able to interpret to current state of the system. We also attempted to gain some quantitative data concerning awareness of a certain situation using a single question in the follow-up questionnaire. After observing the evaluation, it was our opinion that postponing questioning regarding situational awareness until after the evaluation is not as effective as asking the questions either during the situation or directly following it. This is congruent with typical results in formative evaluation when using this approach.

7.1.3 Guideline-based Inspection and Formative Evaluation Coordination

Although the coordination demonstrated in the first evolutionary cycle between guideline-based inspection and formative evaluation proved very useful in tailoring formative evaluation to assess the usability of issues brought up in guidelines-based inspection, we observed certain areas where the communication process between the two broke down. The only communication between the usability specialist performing the guidelines-based usability inspection and usability specialists performing the formative evaluation was a usability issue table. This table listed usability issues that the specialists performing the guidelines-based inspection both specified should be further evaluated and usability issues Crumbs designers chose not to correct. This process did motivate the

creation of benchmark tasks to assess the usability of certain interaction metaphors, but it did not provide a detailed description of why the usability specialists believed the metaphor required further evaluation. Therefore, the formative evaluation was not able to leverage the experience and ideas of the usability specialists fully.

7.2 Modifications to Initial Proposed Usability Methods

Our approach to proposing modifications to the usability engineering methodology evaluated in the first evolutionary cycle was first to attempt to correct the observed individual weaknesses of the guidelines-based inspection method and the formative evaluation method and then to describe a detailed coordination of the two to correct the observed communication limitations.

7.2.1 Guideline-based Inspection

The following four modification suggestions to the guideline-based inspection methodology are made in response to the weaknesses we observed while performing the guidelines-based inspection of Crumbs.

7.2.1.1 Inclusion of Usability Specialists' Experience and Traditional HCI Guidelines

In our experience with usability specialists performing guidelines-based inspections, we observed the tendency of the usability specialists to depend entirely on the guidelines to discover and classify usability issues. Usability specialists often doubted their own judgment of a usability issue's existence if the problem did not neatly fit into one of the specified guidelines. This hesitancy and blind reliance upon the guidelines were not the approach we intended for conducting a guidelines-based usability inspection. We desire the guidelines listed in the *Framework of Usability Characteristics in VEs* be utilized as a scaffolding or resource, not the only lone indicator of usability issues. Each usability specialist possesses his or her own experiences and opinion concerning usability assessment. To remove, restrict or hamper this knowledge and interpretation skill set during the guideline-based inspection process would be a misuse of resources. Also, as already mentioned, the *Framework of Usability Characteristics in VEs* does not include all traditional GUI-based usability guidelines and heuristics that

also apply to VEs. Therefore we propose that the *Framework of Usability Characteristics in VEs* be used as a resource that supports discovery of VE-specific usability issues, and in addition that usability specialists also rely on other heuristics and guidelines present into their repertoire and past experience.

7.2.1.2 Usage Scenarios

As mentioned in the Section 3.1.1.2, Nielsen (1994) encourages the use of usage scenarios if inspectors are not familiar with the domain of the application. Although inclusion of usage scenarios may increase the amount of time required to complete a guidelines-based inspection by requiring a two-pass methodology, usage scenarios provide the benefit of introducing usability specialists to representative tasks. Usage scenarios also acquaint the usability specialists with how objects in the VE are designed to coordinate and communicate. This knowledge will allow the guidelines-based inspector to assess object coordination and communication issues more efficiently and effectively. Usage scenarios also provide usability specialists with a strategy for conducting the second pass of the inspection process which allows usability specialists more freedom to create their own tasks and inspect other parts of the interaction not covered in usage scenarios.

7.2.1.3 Object Interaction Guide

Because there are no interaction metaphors that are standard and universally accepted in the VE community, many new VEs attempt to leverage unique interaction metaphors to support navigation, object selection and object manipulation. Requiring usability specialists to “discover” and create each interaction metaphor independently for each application could result in a major increase in required inspection time. For example, while inspecting a new VE, the author was attempting to navigate through the environment. Unsuccessfully, the author attempted to use several common interaction metaphors that are utilized in similar VEs for supporting navigation. After attempting to utilize all the common interaction metaphors, the author finally was forced to ask for assistance from the application developer. Without personal interaction with and

assistance from the application developer, the author could not effectively utilize the navigation interaction metaphor.

Navigation interaction metaphors can be broken down into three components: direction/target selection, velocity/acceleration selection, and input conditions (Bowman, Koller and Hodges 1998). Each of these components can be further broken down into separate methods of supporting each component. In a similar fashion all object selection and manipulation interaction metaphors can also be broken down into components and sub-components. Therefore, when developing VE applications, designers are free to select the exact metaphor they plan to rely upon to facilitate navigation, object selection and object manipulation. This selection process was not made available to the usability specialist executing the guideline-based inspection of Crumbs. Therefore - before they were able to begin assessing the usability of object selection, object manipulation, and navigation within the system - the usability specialist was first forced to discover and become familiar with Crumbs' individual interaction metaphors, without outside assistance. Because Crumbs utilizes a number of different object manipulation metaphors, this task proved challenging and time-consuming. Again, to reiterate, by no means do we attempt to discourage the use of unique interaction metaphors in development of VEs. Nonetheless, to expedite a guidelines-based inspection process, usability specialists should be given reference material that explains and illustrates individual interaction metaphors used in the application.

7.2.1.4 Guidelines Reduction

In an attempt to provide coverage of a large variety of VEs, the *Framework of Usability Characteristics in VEs* includes guidelines to support usability engineering of a wide variety of VEs. There are a total of 195 guidelines covering topics such as setting, avatars, collaborative task coordination, etc. Some guidelines are not applicable to all VEs. For example, all guidelines that concerned multi-user collaborative VEs are not relevant to a single-user inspection. (This was the case with the guidelines-based inspection of Crumbs.) Also, because Crumbs was so near the completion of its development prior to the guidelines-based inspection, numerous issues such as choosing appropriate input devices and display components were not applicable. Crumbs'

developers had already made major decisions concerning hardware and expended substantial resources. Facilitating a hardware change at the stage of Crumbs' development when the inspection was performed is most often too costly to justify. However, since the usability specialists were not aware of these constraints and were relatively new to the guidelines, they were required to assess whether Crumbs adhered to all 195 guidelines. This unnecessary effort wasted the usability specialists' time, and this waste will multiply as more usability specialists are added to the inspection process. Therefore, prior to executing a guidelines-based usability inspection, a usability specialist assisted by application designers should proceed through the list of guidelines and remove those that do not apply, either because of the stage of development of the application, the particular characteristics of the application, etc. For instance, removing multi-user guidelines and pre-selected input and output device guidelines for Crumbs would have reduced the number of guidelines from 195 to 78, a reduction of sixty percent (60%).

7.2.2 Formative Evaluation

Since our strategy for use of a usability engineering methodology is to uncover the cause of the greatest number of usability issues as efficiently as possible, we promote the collection and interpretation of both qualitative and quantitative data. However, we observed that quantitative data are used primarily to support qualitative data in cases where there are no established metrics against which to compare quantitative data (e.g., error counts and task completion time). Therefore, without abandoning the collection of quantitative data, we must strengthen our methods of collecting qualitative data to determine the root cause of usability issues. This section suggests modifications concentrating on adapting data acquisition procedures utilized during the formative evaluation to support assessment of VE concepts such as presence and situational awareness. Section 7.2.3 contains suggestions for capturing qualitative data in those cases where participants do not perform a concurrent verbal protocol.

7.2.2.1 Qualitative Data

To strengthen our ability to collect qualitative data, we approach formative evaluation in a similar fashion to a behavioral scientist performing an interview. Instead

of relying totally on quantitative data and qualitative data collected by previous methods (e.g., collecting critical incidents and concurrent verbal protocol), we contend that subjective questions should be asked during and between tasks. These questions would be similar to a behavioral science technique known as intensive “semi-structured” interviewing (Brenner 1985). This technique dovetails well with formative evaluation, because the aim of formative evaluation is to discover a wide range of usability issues from a small number of participants. Likewise, intensive “semi-structured” interviewing is designed to utilize a small number of participants and ask a large number of meaningful questions.

7.2.2.1.1 Strengthening Situational Awareness Measurement

Because we have already advocated a strategy to direct the focus of the participant away from the environment between tasks to limit the amount of “play,” we promote utilizing Endlsey’s (1987) Situation Awareness Global Assessment Technique (SAGAT). SAGAT promotes interrupting a scenario occasionally by asking participants questions regarding what is currently taking place in the environment. Answers to these questions are then evaluated against what is actually taking place in the environment. The number of right answers constitutes the SAGAT score and is used for interpretation. One criticism of SAGAT is that introduces a high degree of “artificiality” because it interrupts the natural task flow affiliated with completing a particular scenario. However, as stated earlier, we believe such interruptions are necessary in evaluating VEs, and therefore the interruptions would not be a unique factor attributed solely to SAGAT. The advantage of SAGAT is its ability to extract objective information from the participant, without subjecting the data to the natural degradation that occurs over time. In other words, asking questions during the evaluation you will naturally generate more complete and accurate responses than waiting until after the evaluation.

7.2.2.1.2 Strengthening Presence Measurement

Finding an objective, quantitative measure for presence is a “hot” topic in the VE field because researchers commonly believe that a user’s sense of presence has a direct and substantial impact on his or her ability to complete tasks. Many of the guidelines listed in the *Framework of Usability Characteristics in VEs* indirectly address presence,

such as setting, user embodiment, avatars, etc. Therefore one of our goals was to develop a strategy to assess and measure presence during formative evaluation. One possibility is extending the follow-up questionnaire to encompass more questions pertaining to presence similar to Witmer and Singer's presence questionnaire (Witmer and Singer 1998). Another possible method of measuring presence is the frame of reference conflict resolution proposed by Barfield and Weghorst (1993). Frame of reference conflict resolution assesses how participants resolve situations where the virtual world and the real world are in conflict. Another indication of presence proposed by Held and Durlach is the degree to which a participant reacts to unexpected stimuli present in the surroundings (Held and Durlach 1991). (A personal example of such a reaction would be physically ducking in an attempt to evade a fireball launched during a VR game.) These real-world physical reactions to unreal virtual events suggest that the user was experiencing a strong sense of presence, at least at that particular point in time. Tasks in the formative evaluation could be structured to place participants in situations where their virtual surrounding conflicts with their actual surrounding. For example, assume that in a CAVE the person is actually facing the right wall of the CAVE instead of the front wall. The participant would then be asked to locate objects in the environment. If the participant responds using terms such as "to the left" rather than "on the front wall," you would conclude that the user is defining location in virtually instead of physically and infer a high degree of presence.

7.2.3 Modified Concept Book

Because we concentrated our efforts on collecting and interpreting qualitative data, we researched methods relied upon by other scientific communities that perform similar activities. To formulate our usability engineering methodology, we reviewed and modified a content analysis of qualitative data technique referred to as the Concept Book Approach. A detailed discussion of the Concept Book Approach (Mostyn 1985) to content analysis of qualitative data is found in Section 3.3. The Concept Book Approach is a thirteen-step strategy for the collection and interpretation of qualitative data. The tool used to collect the qualitative data is an interview. The thirteen steps can be divided into three categories:

1. preparing for the interview.
 - a. address briefing.
 - b. sampling.
 - c. associating.
 - d. hypothesis development
2. conducting the interview.
 - a. hypothesis testing.
 - b. immersion.
3. interpreting and recording the interview results.
 - a. categorization.
 - b. incubation.
 - c. synthesis.
 - d. culling.
 - e. interpretation.
 - f. write.
 - g. rethink.

For usability specialists and application designers to locate usability issues in VEs efficiently and effectively, our usability engineering methodology should be designed to collect and interpret the maximum amount of qualitative data from participants. Therefore, our proposed modified concept book VE usability engineering methodology replaces the interview with a formative evaluation. (Actually, we also propose the use of interview procedures to promote the collection of relevant qualitative data.) The following sections provide a step-by-step discussion of the methodology, as modified, to relate to VE usability engineering.

7.2.3.1 Preparation for the Formative Evaluation

There are several usability engineering activities that must be completed prior to conducting a formative evaluation on VEs. The traditional activities include user analysis and task analysis. We agree that completion of these activities is imperative for a successful formative evaluation. However, our research and that of others (Hix, Swan,

Gabbard, McGee, Dubrin, & King 1999) suggests that a usability inspection must also precede a formative evaluation because most VEs incorporate unique interfaces. The following four sections derive their titles from the first four steps in the Concept Book Approach. For each step, we first provide a brief explanation of the original activity. Next we describe our proposals for what should take place during each step in the Modified Concept Book usability engineering methodology.

7.2.3.1.1 Briefing

Mostyn characterized the first step in the Concept Book Approach to content analysis as a period where evaluators conducting the interviews must ensure that they completely understand the research problem (Mostyn 1985). If there are any questions, Mostyn instructs those responsible for the evaluation to question the “clients or sponsor until you are sure you understand the antecedents” (Mostyn 1985). However, usability specialists understand that the full purpose of conducting usability evaluation studies is to improve the usability of an application. Therefore, usability specialists must shift their focus from that utilized in the Concept Book Usability approach of understanding the research problem to understanding the application domain that is to be evaluated. During this step usability specialists collaborate with system designers to perform a user task analysis. The result of this analysis is a set of representative tasks and scenarios, indicative of normal system usage. Also during this step, usability specialists and system designers create an interaction metaphor guide. Each interaction metaphor leveraged in the application is detailed and briefly described to assist usability specialists associated with the guidelines-based inspection and formative evaluation.

7.2.3.1.2 Sampling

Sampling refers to the activity of selecting a group of representative users to participate in the evaluation process. In this step we answer the question, “are the representative users a valid generalization to the overall population of perspective users.” Because we are conducting a formative evaluation, instead of a summative evaluation, this step is not as important as it would be if we were executing a more expensive type of empirical usability evaluation. This step becomes less important because formative evaluations utilize fewer participants, and the limited number of participants makes

usually precludes performing valid statistical analysis. However, we do promote performing a user analysis at this step in an attempt to create a set of evaluation participants that is as representative as possible.

7.2.3.1.3 Associating

During this step the Concept Book Approach urges interviewers to rely upon their knowledge and experience to aid in creation of hypotheses for which data will be collected during the interview. Also during this step, we believe that it is necessary to employ usability specialists to conduct the evaluation. As mentioned throughout this paper, the VE field is dynamic and poses a number of unique characteristics. It is imperative that VE usability specialists perform these evaluations. Using VE-specific guidelines alone may not be sufficient because the dynamic nature of the VE field might change between creation of the guidelines and performance of the evaluation. VE usability specialists are far more likely to be aware of the current state of the field and thus be able to implement their knowledge of usability engineering processes. Beyond promoting the use of VE usability specialists, the associating step requires the assistance of one or more usability specialists who have been briefed on the application to familiarize themselves with the *Framework of Usability Characteristics in VEs* and to remove those guidelines that do not apply to each particular application. The need for and benefits derived from guideline reduction is described in Section 7.2.1.4.

7.2.3.1.4 Hypothesis Development

Hypothesis development refers to the process of creating testable hypotheses applicable to the research problem by way of association or knowledge of human behavior. The result of conducting this step in the Concept Book Approach is a binder of hypotheses, one per page, ordered from front to back by opinionated significance. In a similar fashion, we also advocate creating a formatted list of usability issues prior to conducting the formative evaluation. This list will allow usability specialists performing the formative evaluation to direct their attention to certain aspects of the interface marked as possible usability issues in an attempt to collect subjective and objective data.

The primary method of creating such hypotheses is by executing a guidelines-based inspection leveraging the reduced list of *Framework of Usability Characteristics in*

VEs guidelines created in the Associating step, in addition to experience of the usability specialists conducting the inspection. As previously mentioned, we propose using a two-pass process to conducting a guidelines-based usability inspection. The first pass follows usage scenarios created in the Briefing section to introduce the usability specialists to some representative tasks of the application. The second pass is an open forum, where usability specialists have the freedom to inspect any part of the application and create their own set of tasks. The usability specialists will also have available to them the interaction guide created in the Briefing section to assist them in the inspection process. Similar to the Concept Book Approach, the result of the guidelines-based inspection is also a binder; however, this binder contains known and possible usability issues, known as usability issues. Usability issues are parts of the interface that the usability specialists do not believe violate any of the guidelines, but may nonetheless pose a threat to the usability of the system. One of the primary parts of an interface that is often flagged as a potential usability issues is the use of a unique interaction metaphor. The usability issues and issues are sorted by the opinion of each individual usability specialists based upon usability significance. Each separate page lists the usability issue or problem number, gives a description of the issue or problem, lists the violated guidelines (if applicable), lists the possible causes, and suggests one or more redesign suggestions. The list of possible causes is very important, because this list allows usability specialists in the formative evaluation to structure open-ended questions in an attempt to collect subjective data concerning the usability issue. This list of possible causes also permits the usability specialists conducting the inspection to communicate their experience and knowledge to the usability specialists conducting the formative evaluation. The methodology we have proposed facilitate the use of this information and results in a more efficient use of the usability specialists' time and resources.

Following creation of the usability issues and problem binder, the usability specialists conduct a meeting with the application designers to discuss results of the inspection process. During this meeting, application designers decide which usability issues reported in the binder warrant a redesign effort. Therefore, prior to moving to the next step, application designers are given the opportunity to apply redesign efforts in an attempt to “clean up” usability issues found during the usability inspection process.

Usability issues that are eliminated by redesign are removed from the modified concept book binder. Therefore, the result of this step is a prioritized binder containing a list of usability issues that designers have opted not to act upon, and a list of usability issues that usability specialists conducting the guidelines-based usability inspection marked as needing further evaluation.

7.2.3.2 Conducting the Formative Evaluation

7.2.3.2.1 Hypothesis Testing

Hypothesis testing is the process of conducting an interview by employing the use of open-ended questions and funneling qualitative data. Hypothesis testing in our usability engineering methodology refers to the previously mentioned formative evaluation process. In Section 3.1.3.1 we explained a traditional method of performing a formative evaluation. In chapter 4 and again in this chapter we have modified the traditional formative evaluation approach, adapting it to requirements unique to VEs. The first step in the formative evaluation process is to create a set of benchmark tasks to be performed by users. We already have a set of usage scenarios created in the Briefing step, and we supplement these tasks with tasks created to test the usability issues and issues remaining in the concept book after completion of the preceding step. Tasks must also be created at strategic places in the evaluation process to collect qualitative and quantitative data on presence and situational awareness using an approach similar to SAGAT. Usability specialists create questions regarding the current situation in the application and grade participants' responses.

While conducting a formative evaluation of VEs, our experience indicates that usability specialists should act in a more proactive role than is normally required in a traditional formative evaluation. Tasks attempting to generate data on usability issues, as listed in the concept book, should be accompanied by a set of possible causes. Usability specialists who perform the formative evaluation are responsible for formulating open-ended question to facilitate collection of qualitative data concerning the list of causes. For example, assume the concept book lists a usability issue concerning the inability to visualize an object in the environment, and the list of possible causes includes object color, object size and object texture. The formative evaluation usability specialists, when

faced with that issue, should ask the participant questions trying to extract relevant information as to why they believe the object is hard to visualize. While the participant is performing tasks, usability specialists should be checking off the causes that are associated with the tasks in the concept book binder. If the participant does not independently offer an opinion that satisfies each of the causes listed in the binder, the usability specialists should use a funneling approach to ask a series of open-ended questions to extract the necessary information.

The follow-up questionnaire will also include many of the questions regarding presence that were proposed by Witmer and Singer's presence questionnaire (Witmer and Singer 1998). The result of this step is a collection of both quantitative and qualitative data collected during the formative evaluation.

7.2.3.2.2 Immersion (Recollection)

Immersion, in this case, refers to the use of written, audio, or video recording of the interview process that allows the usability specialist access to what happened at all times. Since we are especially interested in qualitative data, it is often important for usability specialists conducting the formative evaluation to have access to a video or audio recording of the evaluation. This video or audio recording will allow usability specialists to "relive" events and recapture certain cues that might inform certain participants' reactions. Immersion, however is a loaded word in the VE discipline. It normally refers to the ability of a VE to surround users with the virtual world. Therefore, we must rename this step to a term not already in use in the VE community. We chose to name this step "recollection."

7.2.3.3 Interpreting and Recording Results

Following collection of raw data, usability specialists must be able to organize, interpret and record the results.

7.2.3.3.1 Categorizing

Categorization refers to the process of marking each qualitative statement as supporting or rebuking a hypothesis specified in the concept book. The Modified Concept Book Usability Engineering Methodology makes few changes from that

envisioned by the Concept Book Approach. Each hypothesis listed in the concept book should be given a unique code. All qualitative as well as quantitative data should be reviewed in an attempt to categorize each individual piece with an appropriate hypothesis code. Raw data that do not match to or correlate with any concept book hypothesis, should be left without a code and revisited later.

7.2.3.3.2 Incubation

Incubation, in the Concept Book Approach, refers to a period when the researcher reviews the concept book and takes several days to reflect on the presented ideas. Once again, our proposed usability methodology does not diverge from the original intention of this step. We propose that usability specialists review the hypotheses listed in the concept book, review data collected during the evaluation, and then step back from the evaluation to allow thoughts and impressions to incubate. The usability specialists should use this time to reflect subjectively on the evaluation, but more importantly to gain mental and emotional “distance” from the evaluation so they may return refreshed. Often, researchers devote substantial amounts of their resources in short spans of time to a specific research problem. This concentrated effort can lead to researchers continuing to adhere to ideas that are either not supported or weakly supported by the results. Time away from the problem is designed to allow the researcher to approach the research with a refreshed perspective, fewer preconceived notions and hopefully greater independence.

7.2.3.3.3 Synthesis

The Synthesis step in the Concept Book Approach refers to the task of attempting to locate patterns or relationships that could lead to a dominant concept. We propose this step be used in an attempt to discover patterns or relationships in the raw data that were not coded to a specific concept book hypothesis. As noted earlier, usability inspections and empirical evaluations often have a substantial number of non-interevaluation usability issues. Therefore, it is imperative to gather the data that do not pertain to a specific hypothesis created in the guidelines-based usability inspection process and attempt to group those data into categories.

7.2.3.3.4 Culling

The function of this step is to remove the confusing, contradictory, and non-supported data, ideas and hypothesis from the concept book. It is not possible to record every interaction that occurs during the evaluation, and it is the responsibility of usability specialists to truncate the results to include only pertinent data. Therefore, during this step usability specialists should remove from the concept book all ideas that are not supported. They should also remove contradictory and confusing data from the raw data. The end result of the culling step is a trimmed concept book and raw data.

7.2.3.3.5 Interpretation

Interpretation involves usability specialists relying upon their judgment of the raw data to formulate conclusions regarding hypothesis located in the concept book, as well as categories created from data not associated with any concept book issue. Interpretation requires and is based upon experience and expertise. Because large amounts of raw data are qualitative (by design), usability specialists must learn to “read between the lines” and attempt not only to understand the subjective opinion of a participant from what is said, but also to interpret non-verbal communication and what is not said.

7.2.3.3.6 Write

Next usability specialists need to record the evaluation - both its process and task - in written form to submit to application designers. The report provided to designers should highlight strengths and weaknesses of the application, along with guidance as to severity of usability issues and possible redesigns.

7.2.3.3.7 Rethink

The final step in the Concept Book Approach is the task of determining whether the research objectives have been met by reviewing results given in the report. Our research objective is to assess the usability of an application. Therefore this step is not appropriate in our methodology. However, the title of the section does hint at a cyclic approach to revisiting the research objective, and as posited many times in this thesis, usability engineering is an iterative process. A meeting should be conducted between usability specialists who conducted the formative evaluation and designers of the VE

application to discuss the results. This meeting should result in a list of redesigns decisions to address reported usability issues. Following the redesigns, the process should re-iterate to the first step. The final report created in the last step should be transmitted to assist usability specialists who will be performing the next cycle through the methodology.

8 Future Work

A key finding of our research is that the Gabbard *Framework of Usability Characteristics in Virtual Environments* guidelines were successful in uncovering usability issues when applied in a guideline-based inspection. This guidelines-based inspection was also successful in providing guidance in creating user tasks to assess further those issues categorized as possible usability issues. Using guidelines-based inspections to help drive formative evaluation task creation in VEs was at the time this research was performed, a novel idea, and more research is needed to support these findings. The first extension of this work is to perform a usability evaluation using the usability engineering method introduced in chapter 7. This extension would include a second methodology modification cycle and allow HCI researchers to evolve this usability engineering methodology further to eliminate limitations observed during the second pass. Also, to evaluate the usability engineering methodology fully, evaluations need to be completed on multiple VEs that support various VE aspects. The first evaluation was conducted on Crumbs, a single-user, event-driven, small VE. However, Crumbs is not representative of all or even most VEs. Therefore, the usability engineering methodology needs to be performed and tested on VEs that support collaboration between multiple-users, that make use of intelligent agents, and are vast in size and setting. This evaluation may well uncover limitations of the methodology relating to the unique aspects of other VEs.

VE systems can vary greatly in display components. For example, some VEs are designed to execute on desktop machines, while others utilize VE specific display devices such as CAVEs and head mounted displays (HMD). Although these aspects of VEs are covered in the *Framework*, similar evaluations must be performed on each separate output device to ensure our usability engineering methodology effectively discovers usability issues.

One large omission of our work to provide usability engineering methodologies for VEs is the inclusion of multi-user support. Future work should acknowledge and address constraints and limitations of the usability engineering methodology based upon a single user environment. The usability engineering methodology may well require revisions to address the unique aspects of a collaborative VE. One thought is to include a

staggered approach to both the guidelines-based inspection process and the formative evaluation. For example, in the guidelines-based evaluation portion of the methodology, assign one of the usability specialists the role of mediator. Then, during each of the other usability specialists' first inspection pass, require the usability specialists to perform collaborative tasks with the mediator. During the second inspection pass, allow two or more usability specialists to enter the environment and inspect the VE at the same time. Allow the specialists to interact and form their own set of collaborative tasks. This method would permit the mediator to interact with the usability specialists in a predefined way to assist them in inspecting the usability of collaboration on representative tasks.

A similar approach could be used during formative evaluation. Set up would require two separate sites, each with identical hardware and software. The appropriate number of usability specialists, based on our experience we suggest three, would be present at each site to perform the evaluation. The formative evaluation will also be a two pass process, with one process taking place in a controlled environment including usability specialists from the remote site completing a set of predefined tasks. The second pass would introduce a second participant at the remote site and evaluate how two participants complete a set of predefined tasks. Allowing a usability specialist to collaborate during the first pass of both the guidelines-based inspection and the formative evaluation is an attempt to remove uncertainties from the environment. For example, if the goal was to test situational awareness, usability specialists located at the remote site would hide in a predefined place in the environment and the participant or inspector must attempt to locate them.

9 Appendices

9.1 Appendix A: Participant Permission Form

Evaluation of the Crumbs Virtual Reality Application in the CAVE™

Dear Participant,

I invite you to participate in the evaluation of the Crumbs virtual reality (VR) application in the CAVE™ (CAVE Automatic Virtual Environment). The CAVE is a part of the VR facility at the National Center for Supercomputing Applications (NCSA) located on the third floor of the Beckman Institute at the University of Illinois at Urbana-Champaign. The goal for this project is two fold. One is to build a scientific foundation for developing innovative methods in testing the usability of VR programs. Second is to increase awareness of the need of usability evaluations of VR applications.

Crumbs is a viewing tool for volume rendering of regularly sampled data, such as magnetic resonance imaging (MRI), confocal, light microscopy, or computer simulations. The innovative aspects of Crumbs are its high level of interaction and navigation capabilities. Crumbs is currently being used by scientists at several institutions, such as University of Illinois at Chicago, University of California at Berkeley, and University of Chicago. The Crumbs evaluation will use the framework developed by my collaborator Dr. Deborah Hix and her research team at the Virginia Polytechnic Institute and State University (Virginia Tech). Dr. Hix will be conducting a similar project in the CAVE at Virginia Tech. The collaboration between Virginia Tech and University of Illinois on this project will help in identifying ways to enhance Crumbs for viewing 3D data in the CAVE by scientists in the National Computational Science Alliance (Alliance). The Alliance is a partnership of over 50 universities, government, and industry researchers. Virginia Tech is a partner institution in the Alliance.

A total of six people will participate in this project during the spring of 1999. You will complete a survey before and after their participation to provide background information as well as feedback on the tasks. You will be asked to perform simple interaction and navigation tasks using two 3D datasets (sperm tails and spine), with 30 minutes (maximum) per task. The evaluation will be based on observations as you perform these tasks. With your permission, the session will be videotaped for data analysis. Portions of the videotape may be used for presentation of the work at conferences. You face minimal risk as a result of your participation. No names or identifying information will be released without your consent. You will be excluded from this project if you wear electronic devices, such as hearing aids and pacemakers. You may withdraw from this project at any time or refuse to participate without any penalty or lost of any privileges at NCSA facilities.

If you have any questions, please free to call me at the number below, or send an electronic mail at "rbrady@ncsa.uiuc.edu".

I look forward to hearing from you. Thank you for your cooperation.

Sincerely,

Rachael Brady, Principal Investigator
Senior Research Programmer
NCSA and Beckman Institute
(217) 333-3923

(Please cut on this line)

I, _____, agree to participate in the "Evaluation of the
Crumbs Virtual Reality Application in the CAVE" project. I understand that there will be
observations of my interactions in the CAVE. I also understand that there will be surveys
before and after my participation in this project.

☐ I agree to be videotaped during my session.

☐ I DO NOT agree to be videotaped during my session.

(Signed)

(Date)

9.2 Appendix B: Crumbs CAVE Evaluation Pre-Test Survey

We would like to know more about you as well as your Crumbs and virtual reality (VR) background. So, please kindly take a few minutes to complete this form. Thank you._____ **Participant #**

Institutional Information

1. Name
2. Email
3. Title
4. Institution affiliation

Background and Interest in Crumbs (If you haven't previously used Crumbs then skip to next section)

5. Amount of time involved with (approximately in months) using Crumbs
6. Area of study (or research interests)
7. What do you use the tool for?
8. Why did you start using Crumbs?
9. Why do you continue using Crumbs?

Background and Interest in VR

10. What is your interest in VR?
11. How long have you been involved with VR?

12. Have you had any training in VR previously?

13. Have you previously had experience with sonification?

9.3 Appendix C: Participant Instructions

Instructions for Participant in the Crumbs Evaluation

Thank you for agreeing to participate in this experiment. We would like you to help us evaluate an application for visualizing and identifying biological structures using an immersive virtual environment, called Crumbs.

As you can see, Crumbs runs in a CAVE, a large projected immersive environment, for visualization and uses a wand, the default CAVE input device, to manipulate the dataset, manipulate crumbs, and initiate task actions. The wand has two functions. One function of the wand is to provide a method of pointing, clicking, and dragging similar to the functionality of the mouse and cursor on most GUI interaction styles with which you are familiar. The second function is as a set of three perpendicular slice planes with which to view cross-sections of the dataset. For the tasks in this evaluation you will only be required to use the wand as a cursor. The application also uses a menuing system that is similar to menus in most GUIs. The metaphor of “crumbs” is used to understand the main function of the application. Similar to Hansel and Gretel dropping crumbs in the woods so they could mark where they had been, the application allows you to drop “crumbs” in the data volume to mark a desired path. The application supports tasks that fall primarily into three categories: initializing a data volume for visualization, marking and manipulating crumbs, and storage and retrieval of visualization and crumb settings.

The initialization of the dataset for visualization requires the manipulation of various Crumbs objects to modify the data volume settings to differentiate the desired structure from other structure present in the dataset. The application provides several tools that assist in visualizing the dataset. One tool adjusts the opacity of certain density values in the dataset. This allows you to remove the densities in the dataset that are not part of the studied biological structure, so you can view the desired structures more clearly. The second tool is the color map tool. This allows you to assign colors to certain density values that are used to provide contrast between desired structures and undesirable noise. This evaluation includes tasks for using both of these tools.

Marking and manipulating crumbs in a volumetric dataset is the main task supported by the application and therefore is the main focus of this evaluation. The crumbs are manipulated by using a combination of the middle wand button for creation or selection and the wand for maneuvering. Precise placement of Crumbs requires manipulation of the dataset. Tasks have been created to measure the usability of both dataset and crumb manipulation.

Storage and retrieval of visualization and crumb settings requires use of system commands to save the current settings to file and load previously saved setting into the environment. Tasks have also been created to measure the usability of this functionality.

Prior to starting the tasks, a data volume will be loaded and you will be given some time just to play around with it, to try whatever you would like to do. However, during the session, you will be asked to perform several specific tasks using this system. While you are performing some of the specific tasks, we may be timing how well Crumbs helps with these tasks. Therefore we would like for you to work through each task without taking a break; you can take time to relax between tasks if you wish. An evaluator will read a single task to you out loud. You will wait until the evaluator timing the task indicates that the task should begin by saying “start task”. You should then start your interaction with the application to complete the task. Once you feel the task has been completed, you should indicate so by saying “end task”. Once you indicate that

you have finished the task you should stop interacting with the environment and turn toward the evaluators. You are now given the opportunity to further comment on the task you just completed. It is important that you interact with the environment only while data is being collected on the current task. Any other interaction could lead to invalidation of the results.

Because we are interested in why this system is easy or difficult to use we would like you to “think aloud.” That is, we would like you to talk about what you are doing and why you are doing it. You should talk about what you expected to happen that perhaps did not when you perform an action. You should indicate both the positive (good) and negative (bad) aspects of how you have to perform the tasks. Remember to keep talking throughout the whole session. The evaluator may remind you to talk aloud sometimes and may ask you questions about why you have done something or how you feel about some part of the system. This will help us understand more about the system.

Remember that you are helping us evaluate Crumbs; we are not evaluating you. You should feel free to say whatever you think about any aspect of the system or the tasks you are asked to perform.

Finally, to get your opinion of the system, we will ask you to complete a short questionnaire to rate the system, after you have finished using it.

This session should last a little more than one and a half hours.

Before we begin do you have any question?

9.4 Appendix D: Task Description and Reasoning

Data #	Task #	LAMPS	Benchmark Task	Usability Metric	Task Reasoning
Copy the saved crumb files for the two data volumes into the working directory. Load the spine data volume.					
1	1	S	Turn on the low-resolution full visualization of the spinal cord dataset.	Task Performance Time Number of Errors	Evaluate the execution of <i>system commands</i> using the menu system. (This is to see if this task gets any easier since it was already completed earlier).
1	2	S	Load the predefined color map to help visualize the spinal cord.	Task Performance Time Number of Errors	Evaluate the execution of <i>system commands</i> using the menu system. (This is a two step task that the user must know before hand that they must first edit a color map and then load a color map.)
1	3	SM	Retrace the opacity tool to locate the spinal cord.	Task Performance Time Number of Errors	Both evaluating <i>system</i> , and evaluating <i>object selection/manipulation</i> in manipulating the object. This task was suggested by the guideline evaluation to test the usability of this particular tool.
1	4	SM	Remove the opacity tool from view.	Number of Errors.	The opacity tool is not removed using the trashcan like other objects in the environment. Evaluate the use of the menu to remove the opacity tool. This should be easier since the task was executed in the first dataset.
1	5	M	Scale the data volume to a workable size.	Number of Errors.	Scaling must be executed to properly view the data volume.
1	6	M	Manipulate the data volume to a viewpoint where you are looking down the spine.	Task Performance Time	Position that must be used for marking the points.
1	7	M	Mark a point in the center of the first vertebrae.	Task Performance Time Number of Errors	Evaluate the time and errors that occur on the first attempt to mark the center of the vertebrae. This is a different task then the sperm tail dataset marking task. Therefore take quantitative data to enable the comparison of completing this task as the participant continues.

Data #	Task #	LAMPS	Benchmark Task	Usability Metric	Task Reasoning
1	8	M	Mark a point in the center of the second vertebrae.	Task Performance Time Number of Errors	Evaluate the time and errors that occur on the second attempt to mark the center of the vertebrae.
1	9	M	Mark a point in the center of the third vertebrae.	Task Performance Time Number of Errors	Evaluate the time and errors that occur on the third attempt to mark the center of the vertebrae.
1	10	M	Mark points in the center of all remaining vertebrae.	Task Performance Time Number of Errors	Evaluate the time and errors to complete the task.
1	11	S	Determine the length of the arc created by the set of points.	Task Performance Time Number of Errors	Evaluate the issuing of <i>system commands</i> to determine the arc length.
1	12	SM	Adjust the colormap so the most dense values of the dataset is red and the least is yellow.	Task Performance Time Number of Errors	Both evaluating <i>system commands</i> using the menuing system to use the colormap tool, and evaluating <i>object selection/manipulation</i> in manipulating the object. This task was suggested by the guideline evaluation to test the usability of this particular tool.
Allow the participant to rest. While they are getting a drink, or asking questions, or making comments set the application up to run the test on the aural feedback. This set of tasks will involve the Crumbs slider application and will evaluate the effectiveness of the music in determining the density.					
2	1		Click on approximately position 25. Then click on approximately position 76. Which position is denser.	Errors	Can the user evaluate density based on the acoustics alone. Is there a logical mapping between number of instruments and density.
2	2		Adjust slider from position 1 to the position 51 and ask if the data is getting more or less dense.	Errors	Is there a logical mapping between the variety of timbres the instruments provide and the density it represents.
2	3		Adjust slider from 256 to 205. Ask if the data is getting more or less dense.	Errors	Is there a logical mapping between the variety of timbres the instruments provide and the density it represents.

Data #	Task #	LAMPS	Benchmark Task	Usability Metric	Task Reasoning
2	4		Click on approximately 190 and then on approximately 150.	Errors	Can the user evaluate density based on the acoustics alone. Is there a logical mapping between number of instruments and density.
2	5		Click the slider position on 100, 155, and 220.	Errors	Is there a logical mapping between the variety of timbres the instruments provide and the density it represents.
Allow the participant to rest. While they are getting a drink, or asking questions, or making comments set the application up to run the second dataset through for evaluation. This requires starting up Crumbs with the sperm tail data volume. Make sure the default saved crumb file is in the default directory.					
3	1	S	Load the predefined “crumbs” for this dataset.	Task Performance Time Number of Errors	This task has two reasons. #1 to continue evaluation of issuing <i>system commands</i> using the menu system and to also load a set of pre-positioned crumbs which signal where to begin the marking of crumbs for the first dataset. This ensures each participant begins the task at the same place.
3	2	SM	Invoke the clear box object	Number of Errors	Evaluates usability of invoking the specified object, and this object is necessary to execute subsequent tasks.
3	3	M	Scale data volume for easy viewing.	Number of Errors	Evaluates usability of scaling the data volume. Using the middle button and wand to execute a scaling action.
3	4	SM	Manipulate the opacity tool to locate the sperm tail. (Novice - provide a graph of example opacity settings)	Task Performance Time Number of Errors	Both evaluating <i>system commands</i> using the menuing system to use the opacity tool, and evaluating <i>object selection/manipulation</i> in manipulating the object. This task was suggested by the guideline evaluation to test the usability of this particular tool.
3	5	SM	Remove the opacity tool from view.	Number of Errors.	The opacity tool is not removed using the trashcan like other objects in the environment. Evaluate the use of the menu to remove the opacity tool.
3	6	M	Follow the sperm tail marking sequential points in the tail until the end of the tail has been reached.	Task Performance Time Number of Errors	Evaluate object selection/manipulation and navigation within the complex sperm tail.
3	7	S	Save the crumbs.	Number of Errors	This task has two reasons. #1 to continue evaluation of issuing

Data #	Task #	LAMPS	Benchmark Task	Usability Metric	Task Reasoning
					<i>system commands</i> using the menu system and to save the crumbs for later evaluation.
At this time save this copy of the saved sperm tail participant marked crumb file to the directory that matches the participant number. Give the file the name sperm.1					
3	8	S	Select the last marked crumb	Task Performance Time Number of Errors	Evaluate selection technique for individual crumbs.
3	9	M	Delete the selected crumb	Task Performance Time Number of Errors	Evaluate the manipulation of deleting the a single crumb.
3	10	SM	Delete the last 2 crumbs marked in the previous task.	Number of Errors	Evaluation of the <i>crumb selection</i> and <i>crumb manipulation</i> strategies. This task is a result of the guideline evaluation raising the question of a possible stackable task. This will also evaluate if crumb selection and deletion is easier with practice.
3	11	M	Replace the 3 deleted crumbs.	Task Performance Time Number of Errors	Evaluate object selection/manipulation and navigation within the complex sperm tail.
3	12	S	Save the crumbs.	Number of Errors	This task has two reasons. #1 to continue evaluation of issuing <i>system commands</i> using the menu system and to save the crumbs for later evaluation.
At this time save this copy of the saved sperm tail participant marked crumb file to the directory that matches the participant number. Give the file the name sperm.2. Turn off the sound.					
3	13	SM	Delete the last 3 crumbs marked in the previous task.	Number of Errors	Evaluation of the <i>crumb selection</i> and <i>crumb manipulation</i> strategies. This task is a result of the guideline evaluation raising the question of a possible stackable task. This will also evaluate if crumb selection and deletion is easier with practice.
3	14	M	Replace the 3 deleted crumbs.	Task Performance Time Number of Errors	Evaluate object selection/manipulation and navigation within the complex sperm tail.
3	15	S	Save the crumbs.	Number of Errors	This task has two reasons. #1 to continue evaluation of issuing <i>system commands</i> using the menu system and to save the crumbs for later evaluation.
3	16	SM	Remove the Clear Box, from the environment.	Number of Errors	This allows the evaluation of <i>object manipulation</i> and <i>system commands</i> . Most objects are removed from the system using

Data #	Task #	LAMPS	Benchmark Task	Usability Metric	Task Reasoning
					direct manipulation of the object and the trashcan.
Copy the participant created data volumes into the directory created for the participant with the directory file that matches the participant number.					

Appendix E: Auxiliary Time Recording Form

Crumbs Formative Evaluation					
Form:	Auxiliary Time Recording Form				Date:
Evaluators:		Evaluation #:		Participant #:	

Task Number	Time
1.1	
1.2	
1.3	
1.4	
1.5	
1.6	
1.7	
1.8	
1.9	
1.10	
1.11	
1.12	
3.1	
3.2	
3.3	
3.4	
3.5	
3.6	
3.7	
3.8	
3.9	
3.10	
3.11	
3.12	
3.13	
3.14	
3.15	
3.16	

9.5 Appendix F: Feedback Questionnaire

Please circle the numbers that most appropriately reflect your impression concerning Crumbs.

1. Rate your overall satisfaction with the application:

1 ————— 2 ————— 3 ————— 4
 Very Dissatisfied Dissatisfied Satisfied Very Satisfied

2. Rate the usability of the menu system:

1 ————— 2 ————— 3 ————— 4
 Very Hard to Use Hard to Use Easy to Use Very Easy to Use

3. Rate the effectiveness of the sonification in assisting crumb placement:

1 ————— 2 ————— 3 ————— 4
 Very Ineffective Ineffective Effective Very Effective

4. How easy was the opacity object to use:

1 ————— 2 ————— 3 ————— 4
 Very Difficult Difficult Easy Very Easy

5. How easy was the colormap object to use:

1 ————— 2 ————— 3 ————— 4
 Very Difficult Difficult Easy Very Easy

6. Rate the use of the sword for selection:

1 ————— 2 ————— 3 ————— 4
 Very Hard Hard Easy Very Easy

7. Were the objects as a group easy to manipulate:

1 ————— 2 ————— 3 ————— 4
 Very Hard Hard Easy Very Easy

8. Did you feel as if you were a part of the environment (as if you were a physical entity in the application) :

1 ————— 2 ————— 3 ————— 4
 Felt like an Observer Not really Somewhat Very Much A Part

9. Did you get disoriented in the data volume (Where you ever unaware of the data volumes x, y, and z axis):

1 ————— 2 ————— 3 ————— 4
Always Most of the Some of the Never
 Time Time

10. Rate the effectiveness of the audio annotation:

1 ————— 2 ————— 3 ————— 4
Very Ineffective Effective Very
Ineffective Effective

11. What do you like most about Crumbs?

12. What do you like least about Crumbs?

13. What would you like it to do that it doesn't do?

9.6 Appendix G: Usability Specification Table

Crumbs Formative Evaluation						
Form:		Usability Specification Table			Date:	
Evaluators:		Evaluation #:	Participant #:			

Usability Attribute	Measuring Instrument	Value to be Measured	Current Level	Worst Acceptable Level	Planned Target Level	Best Possible Level	Observed Results
Initial Performance	1.1	Error Count					
	1.1	Task Completion Time					
	1.2	Error Count					
	1.2	Task Completion Time					
	1.3	Error Count					
	1.3	Task Completion Time					
	1.4	Error Count					
	1.5	Error Count					
	1.6	Task Completion Time					
	1.7	Error Count					
	1.7	Task Completion Time					
	1.8	Error Count					

Usability Attribute	Measuring Instrument	Value to be Measured	Current Level	Worst Acceptable Level	Planned Target Level	Best Possible Level	Observed Results
	1.8	Task Completion Time					
	1.9	Error Count					
	1.9	Task Completion Time					
	1.10	Error Count					
	1.10	Task Completion Time					
	1.11	Error Count					
	1.11	Task Completion Time					
	1.12	Error Count					
	1.12	Task Completion Time					
	2.1	Error Count					
	2.2	Error Count					
	2.3	Error Count					
	2.4	Error Count					
	2.5	Error Count					
	3.1	Error Count					
	3.1	Task Completion Time					
	3.2	Error Count					

Usability Attribute	Measuring Instrument	Value to be Measured	Current Level	Worst Acceptable Level	Planned Target Level	Best Possible Level	Observed Results
	3.3	Error Count					
	3.4	Error Count					
	3.4	Task Completion Time					
	3.5	Error Count					
	3.6	Error Count					
	3.6	Task Completion Time					
	3.7	Error Count					
	3.7	Task Completion Time					
	3.8	Error Count					
	3.8	Task Completion Time					
	3.9	Error Count					
	3.9	Task Completion Time					
	3.10	Error Count					
	3.11	Error Count					
	3.11	Task Completion Time					
	3.12	Error Count					
	3.13	Error Count					

Usability Attribute	Measuring Instrument	Value to be Measured	Current Level	Worst Acceptable Level	Planned Target Level	Best Possible Level	Observed Results
	3.14	Error Count					
	3.14	Task Completion Time					
	3.15	Error Count					
	3.16	Error Count					
	q.1	Rating					
	q.2	Rating					
	q.3	Rating					
	q.4	Rating					
	q.5	Rating					
	q.6	Rating					
	q.7	Rating					
	q.8	Rating					
	q.9	Rating					
	q.10	Rating					

9.7 Appendix H: Data Collection Form – Time and Error Count

Crumbs Formative Evaluation					
Form:		Data Collection Form – Time and Error Count		Date:	
Evaluators:		Evaluation #:		Participant #:	

Data #	Task #	LAMPS	Benchmark Task	Usability Metric	Task Reasoning
1	1	S	Turn on the low-resolution full visualization of the spinal cord dataset.	Task Performance Time Number of Errors	Evaluate the execution of <i>system commands</i> using the menu system. (This is to see if this task gets any easier since it was already completed earlier).

Elapsed Time:		Tape Counter:		Error Count:	
Critical Incident #	Error	Description			

Appendix J: Task to category mapping

Task	Awareness	Opacity Obj.	Colormap Obj.	Data Volume Obj.	Crumb Obj.	Selection	Presence	Audio Annotations	Sonification
1.1						X			
1.2			X						
1.3		X							
1.4		X							
1.5				X					
1.6				X					
1.7					X				
1.8					X				
1.9					X				
1.10									
1.11						X			
1.12			X						
2.1									X
2.2									X
2.3									X
2.4									X
2.5									X
3.1					X				
3.2				X					
3.3				X					
3.4		X							
3.5		X							
3.6					X				
3.7						X			
3.8						X			
3.9					X				
3.10					X				
3.11					X				
3.12						X			
3.13					X				
3.14					X				
3.15						X			
3.16				X					
q.1	-	-	-	-	-	-	-		-
q.2						X			
q.3									X
q.4		X							
q.5			X						
q.6						X			

9.9 Appendix K: Raw Analytical Data Error Counts

Task	Part #2	Part #3	Part #4	Part #5	Average	Std. Dev.
1.1	1	0	0	1	0.5	0.57735
1.2	1	1	2	1	1.25	0.5
1.3	3	1	1	4	2.25	1.5
1.4	2	0	0	0	0.5	1
1.5	-	0	0	0	0	0
1.6	0	1	0	0	0.25	0.5
1.7	0	1	0	1	0.5	0.57735
1.8	1	1	1	0	0.75	0.5
1.9	3	1	1	0	1.25	1.258306
1.10	2	3	0	7	3	
1.11	1	0	0	0	0.25	0.5
1.12	3	1	5	3	3	1.632993
2.1	-	-	-	-	-	-
2.2	-	-	-	-	-	-
2.3	-	-	-	-	-	-
2.4	-	-	-	-	-	-
2.5	-	-	-	-	-	-
3.1	0	0	0	0	0	0
3.2	0	0	0	1	0.25	0.5
3.3	0	1	0	0	0.25	0.5
3.4	2	1	0	0	0.75	0.957427
3.5	0	1	0	-	0.333333 3	0.57735
3.6	6	-	1	3	3.33	2.516611
3.7	0	-	0	0	0	0
3.8	1	-	1	1	1	0
3.9	0	-	-	-	-	-
3.10	1	-	0	0	0.33	0.57735
3.11	1	-	0	0	0.33	0.57735
3.12	0	-	0	1	0.33	0.57735
3.13	2	-	0	0	0.67	1.154701
3.14	1	-	0	0	0.33	0.57735
3.15	0	-	0	0	0	0
3.16	1	-	0	0	0.333333 3	0.57735

9.10 Appendix L: Raw Analytical Data Completion Times

Task	Part #2	Part #3	Part #4	Part #5	Average	Std. Dev.
1.1	9.04	2.51	6.37	11.9	7.455	3.995852
1.2	11.56	7.91	15.3	13.41	12.045	3.15
1.3	79.74	41.5	46.15	125.41	73.2	38.75
1.4	52.11	4.9	9.25	.35	17.15	23.47
1.5	-	3.6	9.6	7.27	6.82	3.02
1.6	5.91	7.78	6.31	10.51	7.62	2.08
1.7	87.55	159.22	88.93	31	91.68	52.5
1.8	104.02	47.88	18.72	12.79	45.85	41.7
1.9	84.93	48.03	13.81	43.86	47.66	29.15
1.10	223.72	277.06	54.22	301.62	216.41	96.84
1.11	34.23	60.04	16.84	27.03	34.54	18.44
1.12	76.47	127.94	116.51	49.72	92.66	36.15
2.1	-	-	-	-	-	-
2.2	-	-	-	-	-	-
2.3	-	-	-	-	-	-
2.4	-	-	-	-	-	-
2.5	-	-	-	-	-	-
3.1	11.73	7.25	9.69	11.9	10.14	2.17
3.2	4.87	4.52	6.94	17.87	8.55	6.30
3.3	4.89	7.59	11.39	20.21	11.02	6.68
3.4	224.84	108.12	73.45	13.95	105.09	88.8
3.5	13.54	6.63	6.22	-	8.8	4.11
3.6	481.14	-	369.75	653.79	501.556	143.12
3.7	8.3	-	11.4	6.96	8.89	2.28
3.8	8.57	-	11.25	13	10.94	2.23
3.9	14.5	-	5.81	5.56	8.62	5.09
3.10	44.02	-	15.22	10.53	23.26	18.13
3.11	119.05	-	17.47	26.19	54.24	56.3
3.12	5.57	-	14.85	8.13	9.52	4.79
3.13	98.55	-	22.34	17.37	46.09	45.5
3.14	43.6	-	17.85	37.97	33.14	13.54
3.15	5.4	-	8	6.86	6.75	1.30
3.16	22.8	-	5.78	5.03	11.2	10.05

10 References

- Barfield, W. & Weghorst, S. (1993). The Sense of Presence Within Virtual Environments: A Conceptual Framework. In *Human Computer Interaction: Software and Hardware Interfaces*. UK-Elsevier Science Publishers.
- Bowman, D. (1998). Interaction Techniques for Immersive Virtual Environments: Design, Evaluation, and Application. "Boaster" paper presented at the Human-Computer Interaction Consortium (HCIC) Conference, 1998.
- Bowman, D., Koller, D.; and Hodges, L. (1998). A Methodology for the Evaluation of Travel Techniques for Immersive Virtual Environments. *Virtual Reality: Research, Development, and Applications*, vol. 3, no. 2, 1998, pp. 120-131.
- Boyd, C. (1997). Does Immersion Make a Virtual Environment More Usable?, CHI97 Conference Companion, 325-326.
- Brady, R.; Pixton, J.; Baxter, G.; Moran, P.; Potter, C. S.; Carragher, B. & Belmont, A. (1995). Crumbs: A Virtual Environment Tracking Tool for Biological Imaging. In IEEE Symposium on Frontiers in Biomedical Visualization Proceedings, pages 18-25.
- Brady, R.; Bargar, R.; Choi, I. & Reitzer, J. (1996). Auditory Bread Crumbs for Navigating Volumetric Data, Proceedings of IEEE Visualization '96: Hot Topics. IEEE Computer Society Press, Los Alamitos, Calif., pp 25-27.
- Cuomo, D. L. & Bower, C. D. (1992). Stages of User Activity Model as a Basis for User-Centered Interface Evaluation. *Proceedings of Annual Human Factors Society Conference*. Santa Monica: Human Factors Society, 1254-1258.
- Endsley, M. R. (1987). SAGAT: A Methodology for Measurement of Situation Awareness (NOR 87-83). Hawthorne, CA: Northrop Aircraft Division.
- Gabbard, J. L. (1997). A Taxonomy of Usability Characteristics in Virtual Environments. M.S. Thesis, Virginia Polytechnic Institute and State University. 1997. Document available online at <http://www.vpst.org/jgabbard/ve/framework/>.
- Gray, W. D. & Salzman, M. C. (1998). Damaged merchandise? A Review of Experiments that Compare Usability Evaluation Methods. *Human-Computer Interaction*, 13, 203-261.

- Held, R. & Durlack, N. (1991). Telepresence, Time Delay and Adaptation. In Ellis, S.R. (Ed.) *Pictorial Communication in Virtual and Real Environments*. London, UK: Taylor & Francis.
- Hix, D. & Hartson H. R. (1993). *Developing User Interfaces*. John Wiley & Sons, Inc.
- Hix, D.; Swan, J. E.; Gabbard, J. L.; McGee, M.; Dubrin, J. & King, T. (1999). User-Centered Design and Evaluation of a Real-Time Battlefield Visualization Virtual Environment. In *Proceedings of the Virtual Reality '99 Conference*, Houston, Texas, March 13-17, p. 96-103. 1999.
- Kaur, K. (1998). *Designing Virtual Environments for Usability*. Centre for Human-Computer Interface Design. Ph.D. Dissertation. City University. London.
- Kaur, K. (1997). *Designing Virtual Environments for Usability*. In *Proceedings of the INTERACT '97 Conference on Human-Computer Interaction*, Sydney, Australia, p. 636-639. Document available online at <http://web.soi.city.ac.uk/homes/dj524/kully.html> 1997.
- Mack, R. L. & Nielsen, J. (1994). Executive Summary. In *Usability Inspection Methods*, chapter 1, p. 1-23. John Wiley & Sons, Inc. 1994.
- Nemire, K. (1994). Building Usable Virtual Environment Products. *CyberEdge Journal*, 1994, Volume 4, Number 5, pp. 8-14.
- Nielsen, J. & Molich, R. (1990). Heuristic Evaluation of User Interfaces. In *Proceedings of CHI '90* (pp. 249-256). New York: ACM Press.
- Nielsen, J. (1994). Heuristic evaluation. In *Usability Inspection Methods*, chapter 2, pages 25-62. John Wiley & Sons, Inc. 1994.
- Parent, A. (1998). Life-like Virtual Environments: An Introductory Survey. NRC/ERB-1055 : 29 Pages. January 1998.
- Prothero, J. ; Parker, D.; Furness, T. A. & Wells, M. (1995). Towards a Robust, Quantitative Measure of Presence. In *Proceedings of Conference on Experimental Analysis and Measurement of Situational Awareness*, pp. 359-366.
- Roberson, G., Czerwinski, M. & van Datzich, M. (1997). Immersion in Desktop Virtual Reality. In *VIST '97*. Banff.
- Roussos, M.; Johnson, A., Moher, T.; Leigh, J.; Vasilakis, C. & Barnes, C. (1999). Learning and Building Together in an Immersive Virtual World. To be published in

- Presence, Special Issue on Virtual Environments and Learning, MIT Press.
[Document available online at
<http://taz.eecs.uic.edu/~nice/NICE/PAPERS/PRESENCE/presence.html>]
- Salzman, M. C.; Dede, C. & Loftin, R. B. (1995). Usability and Learning in Educational Virtual Realities. In Proceedings of the Human Factors and Ergonomics Society 39th Annual Meeting, San Diego, California, p. 486-490. [Document available online at
<http://www.virtual.gmu.edu/usabpdf.htm>]
- Sheridan, T.B. (1992). Musings on Telepresence and Virtual Presence. *Presence: Teleoperators and Virtual Environments*, 1(1), pp.120-125.
- Slater, M. & Usoh, M. (1993). Presence in immersive virtual environments. *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 90-96. Piscataway, NJ: Institute of Electrical and Electronic Engineers.
- Smith, S. L. & Mosier, J. N. (1986). Guidelines for Designing User Interface Software, ESD-TR-86-278, Bedford, MA: The MITRE Corporation.
- Snow, M. (1996). Charting Presence in Virtual Environments and Its Effects on Performance. Department of Industrial & Systems Engineering. Ph.D. Dissertation. Virginia Tech. Blacksburg, VA.
- Tromp, J.; Hand, C.; Kaur, K.; Istance, H. & Steed, A. (1998). Usability for VR Interfaces: Topics for discussion. The First International Workshop on Usability Evaluation for Virtual Environments: Methods, Results and Future Directions. De Montfort University, Leicester, United Kingdom, December 17. [Document available online at
<http://www.crg.cs.nott.ac.uk/research/technologies/evaluation/workshop/topics/>]
- Witmer, B. & Singer, M. (1998). Measuring Presence in Virtual Environments: A Presence Questionnaire. *Presence*, Vol. 7, No. 3. pp. 225-240.

VITA

Kent Olen Swartz, the son of John W. and Connie J. Swartz, was born on November 30, 1970, in Stuarts Draft, Virginia. He graduated from Stuarts Draft High School in 1989. He attended James Madison University and graduated from this institution in May 1993, with his B.S. degree in Computer Science and minor in Mathematics. He worked at American Management Systems from May 1993 until August 1997 as a software developer. He then entered the computer science graduate program at Virginia Tech and finished coursework in May of 1999. He then went to work for Oracle Corporation as a development manager. This thesis completes his M.S. degree in Computer Science from Virginia Tech.