1-1-2010

Reflecting on the Design and Implementation Issues of Virtual Environments

Chadwick A. Wingrave
University of Central Florida

Joseph J. LaViola Jr.
University of Central Florida

Chadwick A. Wingrave*
Joseph J. LaViola, Jr.
School of Electrical Engineering &
Computer Science
University of Central Florida
Harris Corporation Engineering
Center
Orlando, Florida, 32816-2362

Reflecting on the Design and Implementation Issues of Virtual Environments

Abstract

We present a candid reflection on the issues surrounding virtual environment design and implementation (VEDI) in order to: (1) motivate the topic as a research-worthy undertaking, and (2) attempt a comprehensive listing of impeding VEDI issues so they can be addressed. In order to structure this reflection, an idealized model of VEDI is presented. This model, investigated using mixed methods, resulted in 67 distinct issues along the model's transitions and pathways. These were clustered into 11 themes and used to support five VEDI research challenges.

1 Introduction

Through reflection, [the practitioner] can surface and criticize the tacit understandings that have grown up around the repetitive experiences of a specialized practice, and can make new sense of the situations of uncertainty or uniqueness which he may allow himself to experience.
Schön, 1983, p. 61

Virtual environment design and implementation (VEDI) has long been known to be demanding by its practitioners. Evidence for this is seen in the workshops focused on aspects of VEDI in the early and mid-1990's (Green & Jacob, 1991; Herndon, Van Dam, & Gleicher, 1994) as well as a more recent CHI workshop (Shaer, Jacob, Green, & Luyten, 2008) and the SEARIS series of workshops at IEEE Virtual Reality (2008, 2009) and OOPSLA (2009). Additionally, several important papers have been written through the years commenting on specific VEDI issues; some examples include: VE specification (Jacob, Deligiannidis, & Morrison, 1999), difficulties of interface design and implementation (Myers, 1993), issues with callback architectures (Myers, 1991), and issues with reuse and toolkit death (Steed, 2008). To show how widely held this belief is, one simply has to look at the number of virtual environment (VE) and 3D user interface (3DUI) papers, from the 1990s up to the present day, that begin by discussing the challenges of VEDI. From this, we conclude that in contrast to the tremendous advances in processing and rendering power available for VEs, little has changed regarding their design and implementation.

This work reflects on the design and implementation of virtual environments (VEs) in order to: (1) motivate the topic as a research-worthy undertaking, and

*Correspondence to cwingrav@eecs.ucf.edu.

(2) attempt a comprehensive listing of impeding VEDI issues so they can be addressed. First, why is this important? Despite the issues in VEDI, some successful applications have been built and VEs continue to be actively researched. However, VEs offer greater possibilities yet if they can be built and maintained faster and cheaper than current approaches allow. Arguably more important, many other interface types exist that share aspects of VEDI; and if VEDI cannot be improved, the potential for a wider stagnation exists with implications for gaming, entertainment, and other reality-based interfaces, including "virtual, mixed and augmented reality, tangible interaction, ubiquitous and pervasive computing, context-aware computing, handheld, or mobile interaction, perceptual and affective computing as well as lightweight, tacit or passive interaction" (Jacob et al., 2008). Identifying these issues and coming to a consensus as a community is the first step in eliminating them.

A reflection using multiple investigative techniques is discussed in Section 2. An idealized VEDI model is presented in Section 3 to drive issue identification and organization. The outcome of this model is given in the many issues clustered into themes in Section 4. Section 5 is the complete listing of issues, organized by the idealized VEDI model and cross-referenced with the themes. Lastly, we explore the insight gained by looking at the themes and issues and identify five VEDI research challenges, discussed in Section 6.

2 Issue Identification Methodology

We used a mixed methods approach to identify issues in VEDI. This incorporated our experiences, the experiences of novice developers and practitioners, and their artifacts and code.

2.1 Investigation: Interviews, Artifacts, and Code

This first method, the investigation, collected information from many representations of VE systems. First, interviews were conducted with VE practitioners with their practitioner artifacts. Practitioner artifacts are any form of communication relating to a VE for the practitioner's use, such as meeting notes, UML diagrams, implemented code, graphics, hand-drawn pictures, diagrams, journal entries, conversations, and so on. The interviews were semi-structured and based on the cognitive dimensions questionnaire (CDQ; Blackwell & Green, 2000). Six interviews were performed representing six different universities on four different continents. Second, additional practitioner artifacts were obtained from the 3DI Group at Virginia Tech and from members of the 3DUI mailing list (3DUI Listserv, n.d.). A full report of the collection can be found in Wingrave (2008). Lastly, source code was collected from internal projects, open source projects, and solicitation from the 3DUI Listserv. There were a total of 12 samples of source code selected in all, which covered common 3D interface tasks (Bowman, Kruijff, LaViola, & Poupyrev, 2004).

2.2 Rerepresentation

Developer language and pseudocode descriptions of 3D interaction were collected from nine participating developers, novices regarding VEDI. This provided insight as to how developers, without the bias of existing VEDI tools and any tool-imposed mindset, would naturally think about VEs. These developers watched and rerepresented in their own words a series of 3D interaction video clips (Wingrave, 2008). A total of 2 hours was spent with each participant.

2.3 Personal Experiences

The personal experiences of the authors also play a role in this investigation. In a sense, we embody the knowledge we are seeking and used tools such as the CDQ, thematic analysis, and model driven exploration to elicit this knowledge. Both authors have years of experience designing and implementing VEs for academic and industrial applications. Meeting notes and personal journals were investigated. Additionally, both authors have multiple experiences with students learning VEDI.

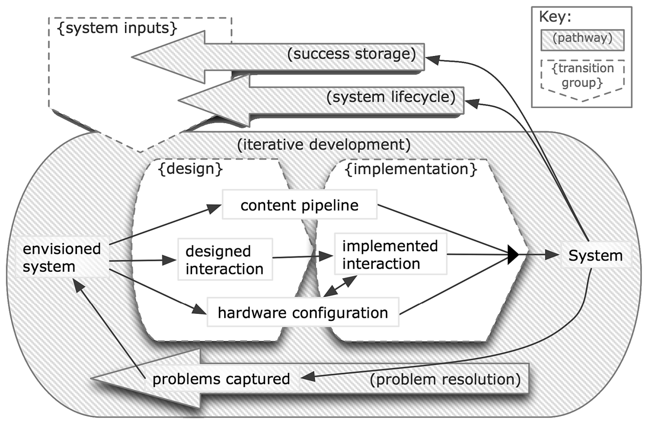

Figure 1. *This idealized VEDI model contains transitions and pathways that are used to structure and explore the identified issues. Starting from the system inputs (upper left), we see how through design and implementation (center) a system is built and through problem resolution (bottom) the system is improved. Throughout the system's lifecycle, requirements change and successful systems have an impact on the state of the art.*

3 Idealized Model of VEDI

An idealized VEDI model is presented in Figure 1 to assist in uncovering and organizing VEDI issues reported in Section 5. The model is a cyclic network of stages with directed transitions between them. Because many stages and transitions are understandable as higher-level structures, stages and transitions are assigned into transition groups and pathways, respectively. This assignment enables issue identification on both specific and broad issues.

3.1 Model Pathways

The four identified VEDI model pathways are iterative development, problem resolution, success storage, and system lifecycle. Iterative development refers to the cycle of software development generally described as design, implementation, and evaluation. Problem resolution refers to the identification of problems and the generation of solutions in VEDI. System lifecycle refers to the maintenance and requirements changes for a VE system. Success storage refers to the pathway where successful VEs impact the state of the art, including: practices, tools, functionality, and body of knowledge.

3.2 Model Transition Groups

The three identified VEDI transition groups are design, implementation, and system inputs. The design section focuses on turning the practitioner's envisioned system into a recognizable plan for its eventual implementation. This includes interface design as well as the design of hardware and content. The implementation section follows from the design section, turning the design into an executable system. Lastly, system inputs refers to the ideas and resources that shape the practitioner's envisioned system, including available hardware, the system requirements, and the current state of the art in VEDI.

3.3 Model Stages

A stage in the model refers to an identifiable goal in the VEDI system. The most identifiable VEDI stages are in regard to the iterative development pathway. These stages include: envisioned system, designed interaction, implemented interaction, content pipeline, hardware configuration, system, and problems captured. The envisioned system stage refers to the practitioner's internal understanding of what the VE is to do. This stage takes into account the practitioner's understanding of requirements, available hardware and the current state of the art. The designed interaction stage is when the practitioner has created a 3DUI plan, such as scenarios, use cases, storyboards, paper prototyping, and so on. The implemented interaction stage occurs when the design has proceeded into an executable representation. Commonly, the designed interaction and the implemented interaction are iterative, capturing issues and impacting the envisioned system accordingly. The content pipeline refers to the development of models, graphics, sounds, animations, scripted behaviors, and such that are used in VEs. The hardware configuration refers to the selected tracking, displays, haptics, props, and other specialized devices for the VE. Both the content pipeline and hardware configuration have design and implementation elements to them but were not divided into separate design and implementation stages due to a lack of a clear divide. The content pipeline is composed of much faster iterations of design and implementation while hardware configuration is often the result of the designed interaction. The system stage refers to a complete VE. This does not necessarily imply a finished system, as the system could be merely the beginning of the iterative development pathway.

The system inputs section contains the final stages of state of the art, available hardware, and requirements. The state of the art refers to the current tools, best community practices, the practitioner's experiences, and their personal libraries of code. The available hardware refers to the physical devices that are available at the current time, and to some extent, the access that the practitioner has to that hardware (purchase, borrow, build, etc.). Requirements here refer to the standard software engineering definition of requirements (Davis, 1993).

3.4 Model Transitions

A transition in the model refers to two things: how one stage influences the next; and in the iterative development pathway, the transformation of the VE to the next stage. Because transitions deal with change, that is where issues occur and are thus the focus of this work. The issues attributed to each transition will be discussed in detail in Section 5.

4 Eleven Themes for Reflection

Themes were developed to organize the large number of identified issues. These were developed by a manual clustering process and are presented below in three theme categories for improved presentation purposes. For traceability, themes and issues are cross-referenced.

4.1 Themes Regarding VEDI Human Aspects

4.1.1 Multiple Varied Skills (T:MVS; Issues 12, 31, 36, and 37). Practitioners require multiple and varied skills for implementation, construction, and content design.

4.1.2 Human Experience and Perception (T:HEP; Issues 1, 13, 42, 43, 44, 46, and 47). A VE user brings human experiences and evolutionary adaptations to bear on their perceptions to create the VE experience; predicting user behavior and supporting the experience is difficult.

4.1.3 Content (T:C; Issues 14, 15, 32, and 33). Attaining proper and effective content for a VE is problematic.

4.2 Themes Regarding VE Design

4.2.1 Design Knowledge (T:DK; Issues 7, 48, 50, 51, 52, 57, and 58). The current means of creating, storing, and reporting VEDI experiences are not effective.

4.2.2 Iterative Prototyping (T:IP; Issues 2, 3, 4, 8, 9, 10, and 53). VEDI requires more iterative prototyping than typical systems and suffers more acutely from the process.

4.2.3 Models: Representation and Reuse (T:MRR; Issues 5, 20, 59, 60, 61, 62, and 63). Despite community efforts to standardize and build better tools, it remains easier to build than to reuse, and models are not widespread in use.

4.2.4 Complex, Chaotic, and Difficult (T:CCD; Issues 6, 11, 21, 22, 23, 38, 39, 45, and 54). The nature of VEDIs is that they are complex in dealing with many different and connected parts, chaotic in that they contain hidden and unpredictable connections, and difficult in that solutions require intelligence, skill, and experience to understand and address.

4.3 Themes Regarding VE Development

4.3.1 Real-Time Operation (T:RTO; Issues 24, 25, 26, 27, and 34). VEs require extremely fast updates to remain interactive, which is at odds with the algorithmic complexity of VEs and the desire for high quality content.

4.3.2 Callbacks and Events (T:C&E; Issues 28, 29, 30, and 35). Callbacks and event handling, the dominant architectures for VEDI, do not organize system complexity and lead to poorly maintainable and extendible systems.

4.3.3 Hardware (T:H; Issues 16, 17, 18, 19, 40, 41, and 49). The best hardware configuration for a VE is hard to identify and obtain.

4.3.4 Tools and Community (T:T&C; Issues 55, 56, 64, 65, 66, and 67). Practitioner communities are separated by tools that do not allow for shared implementations and the growth of communal understanding.

5 Virtual Environment Design and Implementation Issues

Each pathway and transition of the idealized model of VEDI in Figure 1 has its issues, many of which are interrelated. Below, the issues are listed by pathway and transition, with interrelation accounted for by cross references and connecting the issue to a theme. By realizing these issues and their impact on VEDI, we begin the first step toward their resolution.

5.1 Iterative Development Pathway

The iterative development pathway is composed of iterating cycles of design (see Section 5.1.1), implementation (see Section 5.1.2), and problem resolution (see Section 5.1.3) until such time as the necessary usability is achieved, the budget is depleted, a deadline is reached, the project is cancelled, or the system ends its operational lifetime.

5.1.0.1 Issue 1 (T:HEP): Many Issues Are Not Identified Until Late in Development when a Human Can Enter the VE, Because VEs Are a Human Experience. Issues can stem from the interaction between content, interface, and hardware affordances, as well as difficult to predict human factors such as fatigue, stress, and nausea. For example, in a fear of heights rehabilitator, users were heavily distressed that they could not see their feet, generating new requirements in a nearly complete VE.

5.1.0.2 Issue 2 (T:IP): Implementation Is Iterative but the Iteration also Causes Design Changes, Which Forces the Implementation of a Moving Target. Practitioners are not always aware of their design assumptions (issue 60) as there are too many details to fully explore during design. While practitioners might know what they want to happen, they do not know how to implement it with their current tools. They are forced to implement following their tool's *path of least resistance* (Myers, Hudson, & Pausch, 2000).

5.1.0.3 Issue 3 (T:IP): As on the Fly Design Changes Are More Prevalent in VEDI (Issue 1), They Lack the Full Impact Analysis that Is Performed During Design. Over time, practitioners' understanding of the system and their informal practitioner artifacts become out of date, compounding this and other issues (issue 2).

5.1.0.4 Issue 4 (T:IP): The Power of Thought Experiments in Iterative Design Is Not Fully Appreciated and Formalized in 3DUIs, So the Original Intent Is Often Lost in Iteration (Issue 2). Practitioners deal with VEDI issues by an informal process of problem statement and hypothesis checking, a reflection in action (Schön, 1983), in journals and on scratch paper. This is not supported by their design and implementation tools.

5.1.0.5 Issue 5 (T:MRR): Higher Level Models of Systems Are Helpful in System Building but Without Round Trip Engineering (Selic, 2003)*, They Do Not Fit into Iterative Development (Issue 2).* Higher level representations have been used to generate code and stub files, but this is generally a one-way process. Representation changes caused by iteration overwrite the existing development.

5.1.0.6 Issue 6 (T:CCD): VEDI's "Hidden Dependencies" (Blackwell & Green, 2000) *Result in No Portion of the Implementation Ever Being Fully Complete.* Each change has the potential to chaotically impact the system, so every iteration (issue 2) requires the entire system to be fully tested and debugged. In contrast to WIMP implementations, VE events and callbacks are not scoped to individual panels or windows, operate on nondiscrete data, have temporary states that operate over time and in parallel, and have more contextual variations (Jacob et al., 1999). Additionally, this is further exacerbated by the common use of global variables, threads, and distributed and parallel implementations.

5.1.1 Design Group. The transitions in the design group are concerned with creating a solution to the practitioner's envisioned system.

Transition from Envisioned System to Designed Interaction. The transition from envisioned system to designed interaction can be thought of as the 3D user interface (3DUI) and system behavior design.

5.1.1.1 Issue 7 (T:DK): 3DUI Design Lacks a Standard Metaphor to Simplify Design. A metaphor can organize the design of an interface between the human and computer such as the ubiquitous desktop and WIMP metaphors (Hutchins, 1987). With no common metaphor in 3DUIs (Bowman et al., 2004), practitioners are forced to design for the experiences, background, strategies, goals, preferences, and innate abilities (Wingrave, Tintner, Walker, Bowman, & Hodges, 2005) of all users (issue 46). As such, it is more craft than engineering.

5.1.1.2 Issue 8 (T:IP): 3DUI Design Is Difficult to Mediate with Current Prototyping Approaches. Wireframes, storyboards, and other prototyping approaches are easily hand sketched or explored with 2D diagramming tools. In 3D, they are more difficult to sketch due to 3D perspectives, have more intermediary steps that operate over time (4D) and are not as well supported by design tools. As such, we found that storyboards were used by only some practitioners, were not the first form of design, and quickly became out of date during iterative design (issue 2). Some storyboards were created *after* the system was built, but this did not inform design.

5.1.1.3 Issue 9 (T:IP): The Earliest Designs of 3DUI Are Found in Practitioner Journals and Scratch Paper But These Were Not Considered Part of the Design. Most practitioners have journals or write design ideas on paper, both functioning as practitioner artifacts (Wingrave, 2008) that were hastily sketched, incomplete, and were of private ideas. As such, they were seen as inferior (practitioners were even embarrassed to share these, apologetic of their condition) and not managed as part of the design and documentation process.

5.1.1.4 Issue 10 (T:IP): Design and Natural Practitioner Representations Make Great Use of Loaded Terms, Simple Diagrams, and Causal Language, But Do Not Match Implementation Representations. For example, practitioners can use words such as point, grab, and travel to communicate design simply and without unnecessary details. Causal statements such as "When X then Y" are also commonly used. Some support for this can be seen in Alice (Alice.org, n.d.) and Concept-Oriented Design (Wingrave & Bowman, 2008; Wingrave, LaViola, & Bowman, 2009) in the use of loaded terms and causal language.

5.1.1.5 Issue 11 (T:CCD): Unintended Behavior and Errors Result from the Inability of Practitioners to Mentally Simulate Every Interaction in a System. In the investigation, even an expert describing a simple raycasting selection technique failed to identify all its behaviors. This issue is compounded by the constant design changes (issue 3) that require constant repetition of this simulation. In Wingrave et al. (2009), an attempt was made to quantify system complexity through an analysis of spaces and demonstrate that exhaustive consideration is unfeasible with typical representations.

Transition from Envisioned System to Content Pipeline. The transition from envisioned system to content pipeline is about designing the look and feel of the system.

5.1.1.6 Issue 12 (T:MVS): Practitioners Do Not Necessarily Possess the Broad Range of Skills to Create the Varied Content Modalities for VEs. Content creation requires the ability to generate graphics, images, audio, 3D models, animations, touch and vibration content, smells, tastes, and so on. Few practitioners have the skills or resources to create professional content.

5.1.1.7 Issue 13 (T:HEP): Humans Are Extremely Adept at Noticing Unrealistic Content. The more realistic and humanlike, the more humans are able to identify anomalies. This is especially true during moving content. In applications using higher realism for greater presence, this is problematic. Some have claimed nearly realistic humans invoke a feeling of revulsion, called the uncanny valley (Mori, 1970).

5.1.1.8 Issue 14 (T:C): Finding Content Is Problematic. It is difficult to find content that: has a consistent look and feel, is the correct file format and is of the proper detail (see Transition from Content Pipeline to System, below). As such, many VEs have a simplified look and feel, are composed of mixed-themed models, or have an odd mix of high and low detail.

5.1.1.9 Issue 15 (T:C): Users Expect Hollywood Glitz. Movie interfaces are often impractical due to issues such as arm fatigue in Minority Report or the effectiveness of file cabinets in Disclosure. As well, the glitz of movie interfaces is difficult to build and style (issue 12).

Transition from Envisioned System to Hardware Configuration. The hardware selection for a VE greatly impacts the potential immersion levels and interfaces that can be designed.

5.1.1.10 Issue 16 (T:H): Hardware Selection Does Not Adhere Strictly to an Analysis of Requirements. The cause for this is the expense and installation time of hardware. Instead, hardware is generally selected for a system because it has already been purchased, there is expertise in-house, it is currently installed, or tool support exists. As such, there is little opportunity to iterate with and test multiple hardware configurations.

5.1.1.11 Issue 17 (T:H): There Is No Best Hardware Solution, Leading to Issue 18. For three practitioners, the same tracking system may be good because of its small trackers, bad because of poor latency, or unusable as it does not track enough points. Tracking hardware is about trade-offs in update rate, latency, encumbrance, cost, tracked area, tethering, interference, and so on. Display hardware is about trade-offs in cost, field of regard, field of view, brightness, pixel resolution, pixel density, space requirements, power consumption, time to failure, fragility, stereo capability and quality, shared views, occlusion of the real world, installation issues, maintenance, and so on. Haptics and audio hardware have their own associated trade-offs. Quantifying these trade-offs is problematic. As an example, an early prototype of the SSWIM technique (Wingrave, Haciahmetoglu, & Bowman, 2006) failed because the technique blocked the hardware's acoustic sensor; but a magnetic tracker would not have had the same issue.

5.1.1.12 Issue 18 (T:H): There Is No Standard Hardware Platform Because of Issue 17. VEs are often customized to a particular installation and thus not transferable between practitioners. Practitioners have to be aware of how the hardware affects all aspects of their system.

5.1.1.13 Issue 19 (T:H): Cables Encumber and Entangle. VE hardware has many cables that quickly become tangled on the user and with each other. Compounding the problem is the expense and fragility of the hardware that is knocked around during the untangling process. While wireless devices are becoming more common, they are more expensive and their use of batteries creates new issues.

5.1.2 Implementation Group. The transitions in the implementation group are concerned with realizing the design in an implemented and executing system.

Transition from Designed Interaction to Implemented Interaction. This transition is about the issues regarding the implementation of the system's interaction.

5.1.2.1 Issue 20 (T:MRR): Several Notions Are Important for 3D Interfaces But Are Not Always Supported by the Tools. The problem has been referred to as the "tools not speaking the language of the programmer" (Conway et al., 2000). A reason could be that tool creators have a graphics viewpoint, not an interaction viewpoint. Nearly every interviewee expressed a desire for plug and play interfaces or the ability to build interfaces as easily and interchangeably as LEGO blocks. Some functionality could assist practitioners as follows. (1) Tools that support interaction over time and simplify animations; (2) the interdependence of code is problematic, so tool support for mutual exclusion and race condition detection would help; (3) interaction has a system context and state that needs to be managed through formalized means; (4) the ability to temporarily switch on and off particular functionality; (5) support of more natural statements regarding sets of things in interaction (Weinberg, 1998) for statements such as, "select all ducks with green heads" or "move all the molecules within 5 ft of the user."

5.1.2.2 Issue 21 (T:CCD): Implementation Is Complex. Each additional feature increases the number of actions and states of the user, environment, and interface. Since each action must be considered with regard to each state, linear growth in actions and states results in nonlinear growth of complexity. Failing to examine all cases leads to unexpected system behavior (issue 11). As such, systems having only slightly more functionality become much more complicated to design and implement. This problem has been noted as a classical problem of software engineering (Brooks, 1987), referred to as the state space explosion (Harel & Politi, 1998).

5.1.2.3 Issue 22 (T:CCD): "Graphics Transformations Are Brutal," Said an Interviewee. Practitioners have trouble understanding coordinate systems and rotation axes, which they can freely simulate in their heads and discuss with others. One reason might be a lack of *progressive evaluation* (T. Green, 1989), the ability to check work as it is being performed, as graphics and interaction are only experienced as a finished product. Second, matrices and quaternions are write-only representations, in that interpreting them can be difficult. Third, matrices can decompose into alternate but equivalent angular representations, making direct comparisons difficult. Fourth, gimbal lock is problematic in matrix representations. Lastly, quaternions, which address some of these issues, are not widely understood by practitioners or supported in tools.

5.1.2.4 Issue 23 (T:CCD): Everything Is an Algorithm. Nearly all functionality requires a nontrivial algorithm or mathematical description such as collision detection, gesture recognition, simulated physics, intersections, animations, and so on.

5.1.2.5 Issue 24 (T:RTO): VEs Can Be Computationally Constrained. This requires practitioners to consider runtime performance and ensure interactive rates while they simulate the system in their heads (issue 11). A system that runs too slowly could even affect the user's performance, experience, and health (Park & Kenyon, 1999; Ellis, Young, Adelstein, & Ehrlich, 1999; LaViola, 2000).

5.1.2.6 Issue 25 (T:RTO): Heuristics Improve Performance (Issue 24) But Heuristic Assumptions (Issue 60) Cause Problems. Heuristic assumptions can simplify an algorithm and reduce computation, but when these assumptions do not hold in other systems or over time, the algorithm is no longer a solution. In collision detection, assumptions include: detection only on a 2D plane, that objects move slowly, or that bounding boxes or spheres are sufficiently accurate. When assumptions change, the solution is no longer applicable.

5.1.2.7 Issue 26 (T:RTO): Concurrency, Threading, Optimized Code, Multiprocessing, and Distributed Computation Create Problems. VEs commonly separate the simulation thread from the visualization thread so they can update at their own rates (Shaw, M. Green, Liang, & Sun, 1993); however, writing code for these architectures can be problematic. Distributed VEs have several issues, making them ". . . number among the most complex software systems ever constructed" (Stytz, 1996, p. 21).

5.1.2.8 Issue 27 (T:RTO): Concurrent Operations on Scenegraphs Are Problematic for Reasons Other than just Mutual Exclusion. A scenegraph animation could be implemented by setting the value of a scenegraph's node, but this has the potential to overwrite other animations. In another approach, an animation could operate by incrementing a node's value, but this is more complex to implement and debug, and numerically prone to error. Alternatively, a node could be inserted into the scenegraph, but this creates a deeper scenegraph, and the ordering of the nodes can be problematic and limit reuse.

5.1.2.9 Issue 28 (T:C&E): The Dominant Architectures for 3D Interfaces, Event-Based and Callback Architectures, Are Poorly Suited to Operate Over Time and Break Traceability. In a single interface, a button press event could select an object, release an object, add another object to the list of selected, travel, bring up a menu, and so on. In this way, the state of the system, or the context in which the event/callback is occurring, must be manually determined through a series of conditional statements on global variables and/or by determining which callback is currently active. This is both error prone and time consuming as implementation requires splitting functionality from the practitioner's flowing understanding into unencapsulated event handlers and callbacks, losing traceability in the process.

5.1.2.10 Issue 29 (T:C&E): Functionality is Over-Called to Ensure the Correct System State, Leading to Unintended Consequences. While often nondetrimental other than wasting computation (e.g., setting an object's color every time step), over-calling can limit reuse and have unintended consequences.

5.1.2.11 Issue 30 (T:C&E): Event Handling and Callbacks as Implemented in the Tools Can Be Problematic for Many Reasons. First, not all systems support the ability to create new events/callbacks for new hardware, recognition interfaces, or event composition. If they can be created, these new events are problematic, because they must now be maintained (e.g., a new event composed of simultaneous button presses could break if one of the button presses is captured in other code). Second, new events/callbacks may not make sense to others, which limits their reuse (issue 59; i.e., is a "gesture" a hand gesture or a sketching gesture?). Third, higher level events may be more useful, but more difficult to learn or modify for divergent functionality (Myers et al., 2000). Lastly, is an event/callback immediately acted upon, immediately interrupting execution, or is it added to a queue? Imperative languages expect immediate execution, but infinite loops are easily created due to code interdependence.

Transition from Content Pipeline to System. This transition is about correctly bringing content into a VE.

5.1.2.12 Issue 31 (T:MVS): Artists Are Not Practitioners. Gaps in models and improper texturing are not always visible until in a VE and knowledge of level of detail and Z-buffering are not usually the focus of training for graphic artists. The rise of game design is lessening this issue.

5.1.2.13 Issue 32 (T:C): High Quality Content Creates Performance Issues. Though hardware performance has drastically improved, it still does not match the level of quality of human perception. As such, high quality content impacts performance, which is important as VEs require real-time response and lag has been shown to impact presence (Meehan, Razzaque, Whitton, & Brooks, 2003).

5.1.2.14 Issue 33 (T:C): Many Technical Details Hamper Bringing Content into the VE. Content format conversion and loading, especially of animations, is prone to errors and impasses due to technical glitches and limitations of the representations. Content versioning is also problematic. X3D and COLLADA are both working to improve this.

Transition from Implemented Interaction to System. This transition is about debugging and optimizing the system.

5.1.2.15 Issue 34 (T:RTO): Performance Limitations Require Different Implementations that Are More Complex to Debug. Algorithms can take too long to run in the current callback, and so must be implemented in more complex ways, such as in separate threads, distributed among multiple processing units, or split between multiple callbacks (issue 26). Each approach is more difficult to debug than just running a single algorithm.

5.1.2.16 Issue 35 (T:C&E): Debugging Events and Callbacks Is Problematic (Myers, 1991). Events/callbacks often lack metadata or the ability to understand the system context, limiting the ability to know what the actual event/callback is, what caused it, and what it will cause. Multiple triggers can make events even fire twice. Lastly, debugging actions over time is problematic as reality has no breakpoint.

Transition from Hardware Configuration to System. This transition is about building and integrating all the hardware devices and props.

5.1.2.17 Issue 36 (T:MVS): Practitioners Must Be Carpenters, Electricians, Engineers, Artists, and Masters of Duct Tape and Velcro. VEs require the creation of, for example, hardware cable systems, comfortable and configurable straps for devices, customized displays, near-field haptics and camera, tracker and sensor mounts, and sometimes even artistic decorations of hardware to improve immersion.

5.1.2.18 Issue 37 (T:MVS): Practitioners Must Understand Material Properties, Physics, and Optics. Many display types function due to the special properties of materials. Many tracking technologies rely on physics and magnetism. Knowledge of these properties is required to build working systems and avoid expensive mistakes.

Transition Between Hardware Configuration to Implemented Interaction. This transition is about how hardware impacts the user interface.

5.1.2.19 Issue 38 (T:CCD): VE User Interfaces Are Complex and Flood the Practitioner with Problematic Data. Implementing a WIMP interface requires managing a single pointing device, with its position accurately described with a stream of two values (x, y). Implementing a VE interface requires managing several tracked points, described as six (x, y, z, θ, ϕ, ψ) values, at faster update rates with jitter, drift, and commonly experienced tracking loss.

5.1.2.20 Issue 39 (T:CCD): 3DUIs Can Be Recognition-Based Interfaces (Mankoff, Hudson, & Abowd, 2000). Recognition-based interfaces must interpret the data passed to them to determine the correct action to take. In many cases, there can be multiple interpretations. As such, these interfaces need to also allow for recovery of incorrect recognition (Mankoff, Hudson, & Abowd, 2007), which is made problematic by the difficulties in making an undo feature for 3DUIs (Jacob et al., 1999).

5.1.2.21 Issue 40 (T:H): Small Configuration Differences Impact the Interface. Affordances of the hardware impact the interface, such as the grip of devices or their size. The availability of buttons, joysticks, and tactile feedback also impact interface design. The lack of a standard hardware platform (issue 18) compounds this issue.

5.1.2.22 Issue 41 (T:H): Display Selection Can Impact what VEs Can Be Used for and Their Effectiveness. CAVEs and projected displays allow users to see themselves and interact with others. HMDs have a larger field of regard. Stereo can improve some tasks, yet leads to other issues. Understanding the benefits of immersion by identifying its key factors is an open area of research (Bowman & McMahon, 2007).

System Integration. This transition is about integrating the content, hardware, and interaction into a complete VE.

5.1.2.23 Issue 42 (T:HEP): Immersion Hinders Traditional Methods of Debugging. Debugging is hindered by HMD or glasses covering the eyes, holding a device, tangled cables, bulky gloves, keyboard and monitor not in front of the practitioner, and so on. Because of this, debugging is often performed in pairs.

5.1.2.24 Issue 43 (T:HEP): "Incorrect" and "Correct" Are Not Always Obvious. Users quickly adapt and adjust to a VE because they respond to an experience as opposed to knowing how to behave (Wingrave, 2001). Users are easily manipulated by rotations while walking (Razzaque, Swapp, Slater, Whitton, & Steed, 2002) and during hand movements (Burns et al., 2005).

5.1.2.25 Issue 44 (T:HEP): Testing and Debugging the VE Experience Is Not Automatable. Automated unit testing compares outputs of a system; but users, to determine their satisfaction, must experience the VE. This is more problematic due to the never-complete testing of VEs (issue 6).

5.1.2.26 Issue 45 (T:CCD): Where Did It Go? It is all too common for content to never be seen because: an object was not added to the scenegraph, texture coordinates are inside out, texturing images cannot be found, scale too large or too small, the model is in different units from the system, incorrect model translation, incorrect user position, the system moved it, the user using the interface moved it, graphics card issues, clipping, it was loaded behind the user, and on and on. This was discussed in terms of the gulf of evaluation in Conway et al. (2000).

5.1.3 Problem Resolution Pathway. Problems arise in all steps of iterative development and are resolved. This pathway is tacit for smaller issues, but formalized system evaluations (Bowman, Gabbard, & Hix, 2002) find larger problems. Identifying solutions is an informal process, where practitioners apply their own experiences and understanding, but some methodologies exist (Pierce & Pausch, 2006; Wingrave, 2009).

Transition from System to Problems Captured. Evaluation of VEs is a difficult task, more so than even in desktop interfaces. The topic was covered extensively in Bowman et al. (2002).

5.1.3.1 Issue 46 (T:HEP): Users Vary Widely in Their Ability to Operate in VEs, Making Results Difficult to Generalize and Coverage Difficult to Judge. The results from college-age gamers are likely to be inapplicable to, say, older adults or nongamers. Demographics and cognitive aptitudes can account for variance (Wingrave et al., 2005) but not all studies incorporate this, and further, there is no standard method of user reporting.

5.1.3.2 Issue 47 (T:HEP): Maintaining Presence Can Be Difficult During Evaluations. Any real world activity, such as noise, sight of the evaluator or passer-by, as well as common evaluation techniques such as the "think aloud" protocol, can inhibit and break presence. As such, it is difficult to observe users from all angles, manage cables, and discuss ideas during evaluations, with only post-experiment discussions possible.

5.1.3.3 Issue 48 (T:DK): It Can Be Difficult to Capture and Report All the Relevant Information in an Experiment. The volume of data is large and recording it can hamper performance of already stressed systems. Not all data are in the system, such as untracked points of the user, real-world objects and activities, and user experiences, aptitudes, and demographics (issue 46).

5.1.3.4 Issue 49 (T:H): VE Hardware and Software Are Not Robust, Impacting Scheduled Appointments with Participants and Experiment Repeatability. Gloves become sweaty and stop working, projectors go out of alignment, cables tangle, software has bugs, and hardware breaks.

Transitions from Problems Captured to Envisioned System. This transition generates solutions to the captured issues.

5.1.3.5 Issue 50 (T:DK): Generating a Solution to Problems for a VE Is More Art than Science. Generating a solution involves understanding the user's perception and mental model of the VE, relying on the practitioner's experience. Assumptions breaking (Pierce & Pausch, 2006) and Flavors are two proposed methods of addressing this issue (Wingrave, 2009).

5.1.3.6 Issue 51 (T:DK): Each Solution Has a Chaotic Effect on the Interaction. The SSWIM technique (Wingrave et al., 2006) aligned the WIM with the world to improve orientation, but this changed their mental model of the WIM from that of a model (Pausch, Burnette, Brockway, & Weiblen, 1995) to that of a map, necessitating a change of animated movements. According to Krueger (1996, p. 130), "The human interface has too many interrelationships and too many possible manifestations for rigorous experimentation to fully explore them all in the near future, especially since all of the attendant trade-offs are changing almost daily."

5.1.3.7 Issue 52 (T:DK): Solutions Are About Trade-Offs and Understanding and Representing These Trade-Offs Is Problematic. For example, raycasting is faster to point than occlusion selection (Bowman, Johnson, & Hodges, 2001; Wingrave & Bowman, 2005), but nonexperts prefer occlusion selection (Wingrave & Bowman) and occlusion selection gives more fatigue. Guidelines (Bowman et al., 2004) are a current method of representation, but are problematic (see issue 57), with other approaches including claims (Carroll & Rosson, 1992; Payne, Allgood, Chewar, Holbrook, & McCrickard, 2003) and iterative issue solution maps (Wingrave, 2009).

5.2 System Lifecycle Pathway

History has shown that successful systems are rewarded with new requirements (Lehman, 1980). Additionally, VE hardware further impacts systems.

5.2.0.1 Issue 53 (T:IP): The Rewards of a Successful System Are New Requirements (Lehman, 1980) *and Implemented Systems Are Difficult to Retrofit.* This is a given for any system.

5.2.0.2 Issue 54 (T:CCD): Nonpractitioners Are Not Able to Judge and Understand the Time and Cost Impacts of Additional Functionality and Changes. Collaboration can be strained by seemingly simple functionality taking a long time to develop, or when a collaborator's grand ideas are required to be scaled back.

5.2.0.3 Issue 55 (T:T&C): The Software Tools Change Over Time, Making VEDI a Constant Process of Adapting to New Tools. SGI's Performer is a highly optimized tool for rendering large models, but with advances in graphics hardware, tool emphasis moved away from optimization. Reasons for changing tools according to Steed (2008) include that tools lose support, tools are no longer effective compared to other tools, tools are evaluated and determined to be inadequate, tools lack functionality, or tools do not support a hardware or software platform. Importantly, many new tools fail to include historically useful functionality.

5.2.0.4 Issue 56 (T:T&C): Virtual Environments Have a Deep Connection to the Tools and Hardware Used in Their Development, Tying Their Lifetime to the Hardware and Tools. Reasons for this connection include content transferability issues, differences between events and callbacks, and selection of a tool explicitly for a specific feature not common to other tools. Upgrading an existing system may not be worth the effort when hardware support is dropped, hardware breaks, a company closes, a project is abandoned, or the developer leaves.

5.3 Success Storage Pathway

Successful systems advance the state of the art by increasing knowledge, promoting reuse, and feeding back into the community.

5.3.0.1 Issue 57 (T:DK): Guidelines, the Dominant Means of Storing 3D Interface Success, Are Not Serving the Needs of Practitioners. Only one interviewee claimed to look to the literature for design assistance, while others said they did not use guidelines or the literature at all. While this might be seen as practitioner failing, it shows either that the guidelines are too vague, too general, or so straightforward that practitioners tacitly understand them. Claims analysis has been proposed in other fields, but has not been used in VEDI (Sutcliffe, 2000).

5.3.0.2 Issue 58 (T:DK): Guidelines, Created by Experimentation, Have Little External Validity. The conditions affecting a VE experience are still not fully understood (see issues 46 and 51) so the same guideline might not hold under a different VE. This is exacerbated by experimental reporting that often unknowingly, or due to a lack of rigor, leaves important conditions unreported. Due to the lack of system longevity (issue 55), the experimental conditions cannot be retested.

5.3.0.3 Issue 59 (T:MRR): It Is Easier to Build than to Reuse. The “not invented here” mentality unfortunately and regretfully has some truth. It can be better to write code that a practitioner understands than to understand someone else’s complex code with hidden assumptions (issue 60). The net effect is a lack of higher level constructs or a refinement of existing tools for the community; both are lost opportunities for advancing the state of the art.

5.3.0.4 Issue 60 (T:MRR): Practitioners Are Often Unaware of Their Own Assumptions and Intent that Must Be Preserved During Maintenance and Reuse. Using a red raycasting ray has an implicit assumption that selectable objects are not red. An animation could assume that no other animation is acting on an object. Practitioner intent can include the state of data, the content model configuration, the outcome of the task, or the order of how actions will be performed. Creating general solutions to problems is also hindered by algorithmic and computational complexity (issue 25).

5.3.0.5 Issue 61 (T:MRR): Critical Points in the Flow of a System Are Not Available to “Hook” into for Reuse (Polys & Ray, 2006). Often, a system’s state is spread among global variables, callbacks, and events, or is hidden from external code. Whether a state is spread out or hidden, the result is systems that are difficult to extend, with solutions often involving code to implicitly determine a system’s state. This is often imperfect, being based upon assumptions, and difficult to maintain when system changes occur.

5.3.0.6 Issue 62 (T:MRR): Actions Over Time, or Flows of Behavior, Are Difficult to Implement, Maintain, and Reuse. Flows are often woven together across similar events and callbacks and lack proper hooks to the implementation’s internals (issues 28 and 61).

5.3.0.7 Issue 63 (T:MRR): Attempts to Create Formal Models for VEs are Helpful but Tend to Break What Already Works. Model based systems, in addition to requiring time to learn (Pausch, Conway, & Deline, 1992), can impede the use of existing VEDI toolkits, development tools, and coding paradigms with which practitioners are familiar. Additionally, models that lack end to end (Selic, 2003) development, in that they can be used to generate code but lack the ability to absorb back into the model, are not suited for iteration in VEDI development (issue 2).

5.3.0.8 Issue 64 (T:T&C): It Is Difficult to Advance the State of the Art When Communities Are Fractured by the Tools They Use, Which Do Not Allow for Shared Implementations and the Growth of Communal Understanding. The use of in-house tools means communities form around spatially related groups, academic lineage, or arbitrary requirements, hindering the sharing of implementations and reproducibility of results.

5.3.0.9 Issue 65 (T:T&C): Expertise Is Toolkit Based and Not Generally Transferable Between Tools. In a first assignment of a 3DUI course, a simple VE was to be implemented using SVE (Kessler, Bowman, & Hodges, 2000) which took roughly 1 hour to implement by those familiar with SVE. In contrast, those without VEDI experience and those with VEDI experience with other tools, took between 10–20 hr. This was due to how tools provide access to functionality, assumptions held by other tools, and even naming schemes for function calls, events, and the scenegraph. This limits the ability for practitioners to switch to new tools that might be better for their needs but that would require time to ramp up to a level of expertise.

5.3.0.10 Issue 66 (T:T&C): Fractured Communities Mean Little Contribution, Documentation, Examples, and Tool Growth. Communities contribute bug patches, documentation, example code, new features, and new ideas. Because of fractured communities (issue 64), these benefits do not occur and this creates barriers to entry (Myers et al., 2000) that make it harder for others to enter the community.

5.3.0.11 Issue 67 (T:T&C): There Is Little Reward for Building Tools. Academics are not rewarded for building and maintaining tools. Businesses form niche markets, such as game development, and develop specialized functionality and closed tools for their own needs. Business partnerships have traditionally had little impact. As such, the resources available to advance the state of the art are limited.

6 Research Challenges

The following research challenges are proposed as having the greatest impact on improving VEDI and addressing its issues. These were synthesized by clustering possible solutions to the issues above into the following five challenges.

6.1 Natural Representations

Supporting the informal natural representations already used in design, and incorporating them in the implementation, is a means to improve understandability for the practitioners and their team, and to improve communal interaction. In the investigation, there were several structures found in practitioner language useful for capturing practitioners' streaming thoughts, abstractly representing familiar concepts and expressing flows of functionality. Language and practitioner artifacts are able to represent intent, presenting it at the proper level of abstraction for the need of collaborators, reusers, or the practitioners themselves. Related topics include natural programming, concept oriented design, and literate programming.

Themes impacted include T:MVS, T:DK, T:IP, and T:T&C.

6.2 Layered Abstractions

Dividing the representation and functionality into layers, similar to the seven layer OSI model, can create interoperability and work toward standardization, even with the widely varied requirements that exist for VEDI. No single tool has yet addressed all VEDI needs and it is doubtful any single tool ever will. Layers allow tools to narrow their focus, providing clearly defined functionality in a single layer and then communicating with other tools at clearly defined interfaces. In this way, barriers between the tools are removed, communities can form around their domain as opposed to their tools, practitioners can quickly incorporate new functionality, and the longevity of systems can be increased. One initial framework was proposed in Wingrave (2006).

Themes impacted include T:IP, T:MRR, and T:C&D.

6.3 Models of Systems

A model operates at a higher level of representation; affording easier iterative development, improving reuse, containing complexity, reducing chaos, and addressing many of the issues with callbacks and events. A model should also support the strengths of existing development and allow for round-trip engineering (Selic, 2003). Models, and user interface management systems, have many issues in their use (Myers et al., 2000). However, a model, natural and layered, holds promise for creating useful abstractions at a level higher than code.

Themes impacted include T:IP, T:MRR, T:CCD, and T:C&E.

6.4 Create an Archive

A community archive for content, usability claims, solutions to interaction issues, code, unit tests, UI research results, and algorithms (with assumptions listed), can reduce costs, time, and barriers to entry in future VEDI. Tremendous efforts have already created freely available tools and online archives of royalty-free content. Unfortunately, a VEDI archive is more than an organizational task or repeating the methods that created, for example, the comprehensive Perl archive network (CPAN). 3D applications on the web may motivate solutions. Lastly, an archive of VE systems could allow greater reproducibility and experimental validity of scientific studies.

Themes impacted include T:MVS, T:C, T:DK, T:MRR, T:RTO, and T:T&C.

6.5 Solve the Difficult Problems

Tasked with recreating many aspects of reality, VEDI is difficult but practitioners commonly create and recreate solutions; definitive solutions are required so that *practitioners can return to their main task*. Solutions will have to be flexible to the trade-offs inherent to VEs, settling for "good" answers by a choice between possible solutions. VE practitioners should solicit help from others in addressing these difficult problems, including researchers in graphics, hardware, machine learning, AI, software engineering, and the like.

Themes impacted include T:CCD, T:RTO, T:H, and T:T&C.

7 Conclusions

We believe that VE design and implementation remains an active topic of research. Many issues remain to be addressed and solving them will improve VE applications and remove the potential stagnation for related fields. We identified 67 unique issues related to the design and implementation of VEs, clustered them into 11 themes, and have proposed five research challenges for VEDI improvement. It is thus our conclusion that there is no single issue afflicting VEDI and thus no single solution; that is (as in software engineering), there is no silver bullet (Brooks, 1987). As such, solutions will require multiple researchers with varied approaches.

We hope this reflection will stimulate research by identifying VEDI issues so they can be analyzed, targeted, and addressed in an organized manner. In the current state, VE design and implementation is advancing very slowly but we have hope that this will change. Lastly, we recognize that there are potentially other issues that we have not examined in this paper and as such, this should be considered a living work, adding and removing issues over time.

References

3DUI Listserv. (n.d.). Retrieved from http://people.cs.vt.edu/~bowman/3dui.org/3DUI_mailing_list.html .

Alice.org. (n.d.). Retrieved from Alice: An Educational Software that teaches students computer programming in a 3D environment: http://www.alice.org .

Blackwell, A., & Green, T. (2000). A cognitive dimensions questionnaire optimized for users. *Workshop on the Psychology of Programming Interest Group,* 137–154.

Bowman, D., & McMahon, R. (2007). Virtual reality: How much immersion is enough? *IEEE Computer, 40*(7), 36–43.

Bowman, D., Gabbard, J., & Hix, D. (2002). Survey of usability evaluation in virtual environments: Classification and comparison of methods. *Presence: Teleoperators and Virtual Environments, 11*(4), 404–424.

Bowman, D., Johnson, D., & Hodges, L. (2001). Testbed evaluation of virtual environment interaction techniques. *Presence: Teleoperators and Virtual Environments, 10*(1), 75–95.

Bowman, D., Kruijff, E., LaViola, J., & Poupyrev, I. (2004). *3D user interfaces: Theory and practice*. Reading, MA: Addison-Wesley.

Brooks, F. (1987). No silver bullet: Essence and accidents of software engineering. *IEEE Computer, 20*(4), 10–19.

Burns, E., Razzaque, S., Panter, A., Whitton, M., McCallus, M., & Brooks, F. (2005). The hand is slower than the eye: A quantitative exploration of visual dominance over proprioception. *IEEE Virtual Reality,* 3–10.

Carroll, J., & Rosson, M. B. (1992). Getting around the task artifact cycle: How to make claims and design by scenario. *ACM Transactions on Information Systems, 10*(2), 181–212.

Conway, M., Audia, S., Burnette, R., Cosgrove, D., Christiansen, K., Deline, R., et al. (2000). Alice: Lessons learned from building a 3D system for novices. *CHI,* 486–493.

Davis, A. (1993). *Software requirements: Objects, functions and states* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.

Ellis, S., Young, M., Adelstein, B., & Ehrlich, S. (1999). Discrimination of changes in latency during voluntary hand movement of virtual objects. *Proceedings of Human Factors and Ergonomics Society,* 1182–1186.

Green, M., & Jacob, R. (1991). SIGGRAPH '90 workshop report: Software architectures and metaphors for non-WIMP user interfaces. *Computer Graphics, 25*(3), 229–235.

Green, T. (1989). Cognitive dimensions of notations. In A. Sutcliffe & L. Macauley (Eds.), *People and Computers V* (pp. 443–460).

Harel, D., & Politi, M. (1998). *Modeling reactive systems with statecharts: The STATEMATE approach.* New York: McGraw-Hill.

Herndon, K., Van Dam, A., & Gleicher, M. (1994). Workshop on the challenges of 3D interaction. *SIGCHI Bulletin, 26*(4), 1–9.

Hutchins, E. (1987). *Metaphors for interface design.* Workshop on Multimodal Dialogues Including Voice. Institute for Cognitive Science, University of California San Diego.

Jacob, R., Deligiannidis, L., & Morrison, S. (1999). A software model and specification language for non-WIMP user interfaces. *ACM Transactions on Computer-Human Interaction, 6*(1), 1–46.

Jacob, R., Girouard, A., Hirschfield, L., Horn, M., Shaer, O., Solovey, E., et al. (2008). Reality-based interaction: A framework for post-WIMP interfaces. *CHI,* 201–210.

Kessler, D., Bowman, D., & Hodges, L. (2000). The simple virtual environment library: An extensible framework for building VE applications. *Presence: Teleoperators and Virtual Environments, 9*(2), 187–208.

Krueger, M. (1996). Virtual (reality + intelligence). In B. Gorayska & J. Mey (Eds.), *Cognitive technology: In search of the humane interface*. Amsterdam: North-Holland.

LaViola, J. (2000). A discussion of cybersickness in virtual environments. *SIGCHI Bulletin, 2*(2), 47–56.

Lehman, M. (1980). Programs, life cycles and the laws of software evolution. *Proceedings of the IEEE, 15*(3), 1060–1076.

Mankoff, J., Hudson, S., & Abowd, G. (2000). Interaction techniques for ambiguity resolution in recognition-based interfaces. *User Interface Software and Technology,* 11–20.

Mankoff, J., Hudson, S., & Abowd, G. (2007). OOPS: A toolkit supporting mediation techniques for resolving ambiguity in recognition-based interfaces. *Computers and Graphics Special Issue on Calligraphic Interfaces, 24*(6), 819–834.

Meehan, M., Razzaque, S., Whitton, M., & Brooks, M. (2003). Effects of latency on presence in stressful virtual environments. *IEEE Virtual Reality,* 141–148.

Mori, M. (1970). Bukimi no tani (The uncanny valley). *Energy, 7*(4), 33–35.

Myers, B. (1991). Separating application code from toolkits: Eliminating the spaghetti of call-backs. *User Interface Software and Technologies,* 211–220.

Myers, B. (1993). Why are human-computer interfaces difficult to design and implement? *Technical Report CMU-CS-93-183.* Carnegie Mellon University, Pittsburgh, PA.

Myers, B., Hudson, S., & Pausch, R. (2000). Past, present and future of user interface software tools. *ACM Transactions on Computer-Human Interaction, 7*(1), 3–28.

Park, K., & Kenyon, R. (1999). Effects of network characteristics on human performance in a collaborative virtual environment. *IEEE Virtual Reality,* 104–111.

Pausch, R., Burnette, T., Brockway, D., & Weiblen, M. (1995). Navigation and locomotion in virtual worlds via flight into hand-held miniatures. *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques,* 399–400.

Pausch, R., Conway, M., & Deline, R. (1992). Lessons learned from SUIT, the simple user interface toolkit. *ACM Transactions on Information Systems, 10*(4), 320–344.

Payne, C., Allgood, C., Chewar, C., Holbrook, C., & McCrickard, D. (2003). Generalizing interface design knowledge: Lessons learned from developing a claims library. *Information Reuse and Integration,* 362–369.

Pierce, J., & Pausch, R. (2006). Generating 3D interaction techniques by identifying and breaking assumptions. *IEEE Virtual Reality,* 15–21.

Polys, N., & Ray, A. (2006). Supporting mixed reality interfaces through X3D specification. *Workshop on Specification of Mixed Reality User Interfaces: Approaches, Languages, Standardization in IEEE Virtual Reality,* 29–32.

Razzaque, S., Swapp, D., Slater, M., Whitton, M., & Steed, A. (2002). Redirected walking in place. *Eurographics Workshop on Virtual Reality,* 123–130.

Schön, D. (1983). *The reflective practitioner*. New York: Basic Books.

Selic, B. (2003). The pragmatics of model-driven development. *IEEE Software, 20*(5), 19–25.

Shaer, O., Jacob, R., Green, M., & Luyten, K. (2008). User interface description language for next generation user interfaces. *CHI Workshop.*

Shaw, C., Green, M., Liang, J., & Sun, Y. (1993). Decoupled simulation in virtual reality with the MR toolkit. *Transactions on Information Systems, 11*(3), 287–317.

Steed, A. (2008). Some useful abstractions for re-usable virtual environment platform. *SEARIS Workshop at IEEE Virtual Reality.*

Stytz, M. (1996). Distributed virtual environments. *Computer Graphics and Applications, 16*(3), 19–31.

Sutcliffe, A. (2000). On the effective use and reuse of HCI knowledge. *ACM Transactions on Computer-Human Interaction, 7*(2), 197–221.

Weinberg, G. (1998). *The psychology of computer programming* (silver ed.). New York: Dorset House.

Wingrave, C. (2001). *Nuance-oriented virtual environments* (Unpublished masters thesis). Virginia Tech, Blacksburg, VA.

Wingrave, C. (2006). Understanding the limits and benefits of a 3D interface specification. *IEEE VR Workshop on Specification of Mixed Reality User Interfaces,* 12–15.

Wingrave, C. (2008). *Concept-oriented design in chasm: Conversational domain language inspired 3D user interface de-*

sign and development (Unpublished doctoral dissertation). Virginia Tech, Blacksburg, VA.

Wingrave, C. (2009). 3DUI flavors beyond vanilla. *SEARIS Workshop in IEEE Virtual Reality*. Retrieved 3/25/10 from http://www.searis.net/index.php5/Workshop2009_Program .

Wingrave, C., & Bowman, D. (2005). Baseline factors for raycasting selection. Proceedings on CD of *Virtual Reality International*.

Wingrave, C., & Bowman, D. (2008). Tiered developer-centric representations for 3D interfaces: Concept-oriented design in Chasm. *IEEE Virtual Reality,* 193–200.

Wingrave, C., Haciahmetoglu, Y., & Bowman, D. (2006). Overcoming world in miniature limitations by a scaled and scrolling WIM. *Symposium of 3D User Interfaces,* 11–16.

Wingrave, C., LaViola, J., & Bowman, D. (2009). A natural, tiered and executable UIDL for 3D user interfaces based on concept-oriented design. *ACM Transactions on Computer-Human Interaction, 16*(4), 1–36.

Wingrave, C., Tintner, R., Walker, B., Bowman, D., & Hodges, L. (2005). Exploring individual differences in raybased selection: Strategies and traits. *IEEE Virtual Reality,* 163–170.