Reflections on the state of developing virtual environments

Andrew Ray

Department of Information Technology Radford University aaray@radford.edu

***Abstract*— Virtual environments (VEs) demonstrate the immense potential computer technology can provide to society. VEs have been created for almost two decades, but standardized tools and procedures for their creation do not exist. Numerous efforts to create tools for creating VEs have come and gone, but there is little consensus among tool creators for establishing a common subset of standard features that developers can expect. Currently, developers use one of many Virtual Reality (VR) toolkits to create a VE. However, VR toolkits are problematic when it comes to interoperability between applications and other VR toolkits. This paper investigates why the development tools are in this state. A discussion on the history of VR toolkits and developer experiences is used to show what developers face when they create a VE. Next, Three Dimensional Interaction Technique (3DIT) toolkits are introduced to show a new way of developing some parts of VEs. Lastly, a vision for the future of VE development that may help improve the next generation of toolkits is presented.**

Index Terms — Virtual Reality, Software Engineering, 3D User Interfaces, Software Architecture

1. Introduction

VR toolkits exist to help developers create VEs. It was difficult for both non-VR researchers and VR Researchers to build VEs over a decade ago, and this fact is still true today (Bryson 96 and Pape 96). The state of the art in VR toolkits has its roots in the mid nineties when scientists and VR Researchers discovered three major facts: creating VEs is difficult, creating VEs that work on multiple hardware configurations is even more difficult, and developing interactions for VEs is a non-trivial task that must be performed in addition to the previous steps. The research community is still trying to find the best way to overcome these challenges. A short investigation of academic literature provides dozens if not hundreds of VR toolkits that try and solve these problems. The myriad number of tools illustrates the difficulty of VE creation. Publications provide a rich source of knowledge about the challenges facing VE creators. However, harvesting and reusing this knowledge has only happened on a publication level. When actual software artifacts are taken into consideration, there is virtually no software reuse present in the field of VR. Because of this, it can take months to years to develop a VR toolkit sufficient for a particular sites needs. Beyond this it can take months to build a VE with a VR toolkit. Even after all this work is done to create a VE, it is rarely reused at a software level between VR toolkits. The usual exception is copying and pasting small portions of code. Until actual software reuse becomes prevalent between both VEs and VR toolkits, the current trend of yet another VR toolkit will continue to happen. This paper presents my view of how to slow the

cycle of toolkit creation. I hope to show the steps that the VR community can take to improve the current state of VE development. I do this by illustrating the impact on developers when they use VR toolkits and compare and contrast this experience with related toolkits for three dimensional interaction techniques before introducing my view on the future of toolkit development.

2. A short history of VR toolkits

The CAVELIBS software was one of the first well-known VR toolkits (Pape 96). The MR toolkit is another example of an early VR toolkit (Shaw 93). The MR toolkit no longer exists, but CAVELIBS is still available for use today. There were enough deficiencies (cost, complexity, lack of flexibility) in the CAVELIBS and other early VR toolkits to warrant the creation of a multitude of similar toolkits. SVE, VRJuggler, DIVERSE, and Avocado are a few examples of VR toolkits that were released after the CAVELIBS toolkit was created (Kessler 2000, Cruz 2002, Kelso 2002, Tramberend 1999). Each of these toolkits has different advantages and disadvantages, and there is no "standard" VR toolkit that has been adopted by the VE development community. One of the major issues with these tools is that it takes a significant amount of development effort to create one. Sometimes it takes years to create a working toolkit. The range in publication dates from the CAVELIBS to these tools reflects the time necessary to advance beyond the general idea of VR toolkits. Additionally, keeping a VR toolkit maintained and current in an academic environment can be daunting. Two more issues are training time for developers to become proficient with a particular toolkit and the foci of each toolkit.

Because each toolkit has a few focuses, a majority of locations that have VR hardware usually create their own tools, or tweak existing toolkits for their usage. For example, if a VR toolkit is designed only to provide input for a specific range of joysticks and head trackers, and an application needs to use a motion platform, change must happen. Change is usually realized by the modification of a toolkit or the creation of a new one.

Using multiple toolkits would not pose a significant challenge except they are incompatible. VR toolkits share the same basic idea, but an application must be rewritten from scratch if it is to work with a different VR toolkit. The first wave of VR toolkits displayed this trait, and it is still true today.

3. Developer experience with VR toolkit development

This section will cover common experiences that developers have when developing with a VR toolkit. These experiences are not well known due to rarely being published. For example, a book whose purposes include explaining how to implement VEs, and it contains no practical information on VR toolkits or the methods tools use to create VEs (Staney 2002).

3.1 How to pick a VR toolkit

Developers have numerous options when it comes to choosing VR toolkits. The process of picking one is subjective, and depends on many factors. How critical is it that the toolkit support the hardware available at the site out of the box? What programming language are the developers familiar with? What graphical toolkits are best suited to create the types of applications that users want to experience? All of these options and more must be weighed carefully because they can have long lasting consequences.

One of these consequences is illustrated by the architectural decision of who is in charge.

3.2 Who is in charge, toolkit or developer?

One of critical differences in VE applications is whether the VR toolkit or the developer is in charge. With the VR toolkit in charge, developers are often restricted with what they can work on and what they can do; whereas, with the developer in charge, interoperability is much easier to accomplish. As stated earlier, there are a myriad of different libraries available to help developers. By restricting which libraries can be used, it can become quite difficult on the developer. From personal experience, interoperability is usually much lower among VR toolkits that are designed to be the center of the world. This seemingly minor point influences the design of VEs, and can greatly restrict developer productivity. A short foray into two different toolkits that espouse each of these two views illustrates these points.

3.2.1 Center of the world development

VRJuggler is a toolkit that uses the center of the world style of development. In order to develop a VE with VR Juggler a developer has to create a class that inherits from an interface for a specific graphical toolkit. Once this is done a series of functions must be implemented which are steps in the application loop. An application loop is usually structured in the following manner:

- Callback before rendering (usually used for gathering tracking data)
- Rendering callback
- Post-rendering callback (usually used for simulation)

Once these functions are implemented, then a simple driver program is created that initializes VRJuggler, creates an instance of the class created earlier, gives it to VRJuggler and then tells VRJuggler to run. VRJuggler then runs the application until the VE user stops the application. However, when multiple toolkits or existing applications are modified to work with VRJuggler, if more than one of them has a center of the world paradigm then integrating into an application becomes a nightmare for the developer. A multi-threaded application with one thread dedicated for each center of the world toolkit can mitigate this problem. However, highly skilled developers are usually required for this type of development. In order to simplify some of these issues, non-center of the world toolkits were developed.

3.2.2 Non-center of the world development

DIVERSE is an example of a non-center of the world toolkit. The purpose of non-center of the world tools is usually to accommodate the developer, not the creator of the toolkit. DIVERSE provides the same capabilities that VR Juggler does, but doesn't require developers to write an application using a specific framework. For example, DIVERSE is designed to take traditional scenegraph and OpenGL code and allow it to work with DIVERSE without having to rework the architecture of the application. All that is required is to take an existing application and tell DIVERSE what the rendering callback is (OpenGL), or add the root node of the sceneraph to the root node of the DIVERSE scenegraph. DIVERSE can then display the application in any physical environment that it supports. It also provides several navigations for users so developers do not have to custom develop navigations for each application.

The major difference between the styles of development being discussed lies in the fact that instead of handing off control of the application to DIVERSE, a developer needs to call an update function for DIVERSE. By

handing control over to the user, the DIVERSE architecture allows for multiple toolkits to be easily integrated in a single application. For example, an SVE application was written and was able to use DIVERSE's cluster functionality to provide synchronized input on a linux cluster. VRJuggler's architecture would make this difficult to achieve.

3.2.3 Impact on the developer

Differences between center of the world and non-center of the world toolkits may seem miniscule, but when these decisions are multiplied by using the same mentality for an entire toolkit, the workload for developers changes dramatically. An example of this can be found in a sample application that loads a model and provides navigation for it. One version written with VRJuggler 2.2 has 487 Lines of Code (LOC). A comparable DIVERSE 3.0 version of this application is 17 LOC. A major portion of the LOC difference is because DIVERSE provides navigations and VRJuggler does not. VRJuggler is more of a virtual portable operating system for VE creation whereas DIVERSE is designed to allow developers to quickly build applications for desktop or immersive systems while providing the same capabilities that VR Juggler provides. A non-center of the world paradigm allows for more flexibility and the ability to reuse existing components compared to a toolkit that is the center of the world. Beyond non-center of the world toolkits there is an even simpler method for creating simpler VEs by focusing on higher level abstractions instead of low level development.

3.3 User centric tools

Several early researchers discovered early on that developing VEs is difficult. Instead of building better traditional VR toolkits, they created tools that allowed non-developers to create VEs. In order to accomplish this, several pieces of VR toolkit functionality were sacrificed for simplicity. Alice is the premier example of these tools; however, commercial tools such as VirTools also exist (Conway 2000 and Virtools 2008).

The major focus of these tools is to strip away the complexities of traditional VR toolkit development. For example, take object manipulation. A typical VR toolkit has the developer use matrix multiplication to change the position of an object. A user centric tool would use a higher level abstraction such as object.move(forward, 1). The disadvantage of such abstractions is that it limits the flexibility that developers have. However, the VEs created with limited flexibility can be high quality, and prove that the VR community should look at what experience these toolkits provide the end user.

3.4 What happens when existing VR toolkits are not sufficient for developers

One of more disheartening moments in virtual environment creation is the realization that you will have to step back and create the tools necessary for developing a particular application. As noted earlier, it is a non-trivial task to build a VR toolkit, and it is a choice that few developers are eager to make. However, there are good reasons for creating a new toolkit. An example from my career is used to illustrate one instance where creating a VR toolkit was necessary and what it produced.

3.4.1 Rationale for creation of DIVERSE

When the hardware for a CAVE system was acquired at Virginia Tech, there was a natural need for software to power the hardware. One of the initial research applications planned for creation was a crane ship simulator. The simulator necessitated the use of a motion platform inside of a CAVE. For the mid 1990s, this

was novel and commercial software such as CAVELIBS and VEGA could not provide the functionality needed to develop this application. It turned out that the hardware independent promises of CAVELIBS had their limits. A standalone application or a VR toolkit that would allow the usage of a motion platform in a VE had to be built. Several researchers worked together to build a Device Independent tool that was meant to build Virtual Environments that are Reconfigurable, Scalable, and Extensible (DIVERSE). DIVERSE was meant to do much more than just provide motion platform support. It was designed to make VE creation much simpler than with existing VR toolkits. One unique feature of this VR toolkit is that it started with input before addressing graphics.

3.4.2 Design decisions

There are three design decisions that were made for DIVERSE. The first decision was to work by default, secondly stay out of the developer's way, and lastly be easy to use (Kelso 2002). Many VR toolkits of this era required extensive configuration before they could be used to produce working application; used a center of the world paradigm that disrupted the developers workflow; and were not designed to allow parts of one VE to be reused in another.

The design decisions led to a software architecture that has a highly modular design, augments instead of replacing existing work, and allows programs to work anywhere (Keslo 2002). Most VR toolkits provide the capability to work anywhere because that is the classic definition of a VR toolkit. However, most VR toolkits focus on replacing existing libraries with one of their own. Reuse was one of the main concerns both for the creation of the VR toolkit, and for VEs created with the toolkit. The result of these decisions lead to a unique toolkit structure.

3.4.3 Structure of DIVERSE

Most VR toolkits are monolithic in nature. They are one big lumbering tool that must be used in its entirety. SVE and VRJuggler exemplify this type of structure. DIVERSE is a modular toolkit. It contains two major parts, a utility layer called DTK and a graphics layer. The utility layer provides plugins that are loadable at runtime, a framework for applications, shared memory, device drivers, and navigations. On top of this, optional layers for specific graphical toolkits have been developed. DPF, the first graphical layer was dependent on a proprietary scenegraph and is no longer used. The second graphical layer, DGL, provides the capability to display OpenGL and other toolkits that produce OpenGL in VEs (Ray 2008). The power of this structure became evident when DIVERSE needed to be upgraded to work on a cluster. Because of the dual layering and modular nature of DIVERSE, clustering plugins were created using the non-graphical layer, and worked with either graphical layer. This upgrade required no code changes to any layer of DIVERSE and over ninety percent of the applications written with DIVERSE needed no changes to work on a cluster. It also allowed for the SVE application to use only the non-graphical portion of DIVERSE. Although these are novel architecture decisions compared to other VR toolkits, their purpose was solely to make an efficient working system.

3.4.4 Maintenance of DIVERSE

DIVERSE has been a reasonably successful VR Toolkit that has been adopted by at least a dozen sites outside of where it was created. However, it faced significant challenges in staying updated and current. There was little support for making, creating, and

updating the tool itself. The need was for interactive VEs, not for tools. Although DIVERSE was created to support the VE creation efforts at Virginia Tech, funding cuts and differing interests essentially put DIVERSE into a purely maintenance mode where the authors of it have kept it up and running but have not been able to advance it due to other demands on their time. VR Juggler is in a similar situation. Academia is not well suited to the long term maintenance and improvement of technology. Only recently has VR taken off in the main-stream and commercial companies have stared offering non-niche based support for VR developers (Parkin 2014).

4. Beyond VR toolkits

Many 3DIT developers are aware of the issues that exist with most VR toolkits. This is because the problems with VR toolkits have exacerbated development difficulty of 3DITs. An analysis of the proceedings from the first symposium on Three Dimensional User Interfaces (3DUIs) reveals several interesting facts. Out of the 25 papers, only five of the papers used or created tools that are available to the public. Only one of these papers exclusively used freely available software *not* developed by their research group. Nine of the papers didn't even mention the software tools used in creating the interfaces or techniques. Six created their own tools and did not make them available. An additional four used tools that their group created, but were not available for public review (IEEE 2006). These statistics have remain unchanged over the past several years because a review of 3DUI applications found the same exact results six years later (Tukaa 2012).

3DIT developers have noticed this and have started to act. They have created a new direction for 3DIT toolkits that identify strengths of VR toolkits and do not replicate them. Their main focus is to address the weakness of VR toolkits. It is possible that these tools will revolutionize VR toolkit development if their approach and implementation is polished. However, adoption has many difficulties. There is a strong sense of “not invented here” within the 3DUI community and it will take time and effort to provide some semblance of a common starting ground for 3DUI applications (Tukka 2012).

4.1 History of 3DIT specific toolkits

3DIT tools first focus on what steps are necessary to realize an interaction technique, not on how those steps are implemented. Secondly, these tools tend to use a descriptive language (XML) instead of an object-oriented language (C++). Generally steps are realized by reusable components. How these components are created and the degree of reusability is still not well understood. In fact, even though toolkits exist to facilitate interaction, most 3DUI application developers still implement well known basic tasks such as (Tukka 2012).

One toolkit that uses this approach is InTml (Figueora 2002). It is primarily focused on describing input to different black boxes, and how those boxes are connected together. Its weaknesses lie in that these black boxes must be created with a traditional VR toolkit, and there is little support for creating them. The second toolkit is NiMMiT (Vanacken 2006). NiMMiT builds on InTml’s work but is allows multi-modal input. It also offers a graphical editor for connecting black boxes instead of requiring developers to work with XML. However, it has weaknesses similar to InTmL.

A third toolkit that uses this approach is CHASM (Wingrave 2008). CHASM represents a break from previous tools

because it does not focus solely on describing an interaction technique. It is a tool that structures the development of the black boxes through code generation. An important aspect of this work is that it investigates the granularity of the black boxes and the relationship behavior between them. A major finding from this work is that by using black boxes to represent a step in implementation, developers can have a linear increase in black boxes but will get an exponential increase in possibilities for creating techniques. Its problems are similar to the previous two tools, and CHASM was not designed for allowing techniques to be reused among multiple VR toolkits, even though it is theoretically possible.

The last toolkit this papers covers in this general area is the Interaction Framework For Innovation (IFFI) (Ray 2007). IFFI includes elements of the previous approaches, but focuses on reusability of 3DITs and VEs. The previous toolkits are designed for describing and creating 3DITs. While description and creation is an important problem, these approaches are short sighted in one critical area. Until reusability of VEs and 3DITs is possible across multiple VR toolkits, there will be no major change in the field of VR (Tukka 2012). IFFI is the first step towards toolkits designed to interoperate with multiple VR toolkits that allow both 3DITs and VEs to be portable. It is the first toolkit that demonstrated multiple 3DITs working on different VR toolkits without modification.

However, significant challenges still exist. The research community remains divided on exactly what the correct direction to take is. Many significant questions still exist in this are and require further research. For example, is standardization the answer? Is a multipurpose toolkit really that useful for developers? Are toolkits that provide reusability the answer? Also, how would a toolkit that allows for interoperability and reusability to gain the critical momentum necessary for mass adoption? How can a toolkit quantify how easy it is to adopt and apply to individual projects? Academia is not well suited for these pursuits and little momentum other than publishing of potential solutions to the problem has been accomplished.

4.2 Characteristics of 3DIT specific toolkits

Compared to VR toolkits, this new direction does not have similar breadth and depth of experience. However, 3DIT toolkits have ground breaking features and approaches compared to traditional VR toolkits. The major new features are component based architectures, GUI editors, code generation, descriptive languages, and reuse. Component based architectures mean that higher-level software reuse beyond copying and pasting is possible. GUI editors allow developers to create toolkits outside of traditional programming languages. Code generation means that portions of the application are created for you, and help create the structure necessary for an application. Descriptive languages allow the behavior of an application to be stated outside of a low-level implementation language. Reuse means that portions of techniques can finally be reused outside of existing techniques.

5. One possible future of VE development

It is clear that the methods used to create VEs are fraught with problems. The sheer number of components necessary to create a VE illustrates these problems. When developers have to deal with structuring complicated interaction behavior into a simple update callback, difficulties arise. VR toolkits provide a wealth of research about implementing VEs, but they do not

offer solutions to this problem. Additionally there has been little success in harnessing this research beyond sharing publications. Only a few VR toolkits have ever successfully been made available to the public. One research area that can harness the power of most VR toolkits is 3DIT specific toolkits. A different approach to VE and behavior specification is provided by these toolkits. Even though VR toolkits have been around longer, they have yet to change their traditional method of operation.

Because considerable effort is spent on VR toolkit creation, and each one is significantly different, each institution usually has its own. Reusing work from multiple toolkits is next to impossible because of this decision. Any future direction of VE creation must work with any VR toolkit. If interoperability is not chosen, small fragmented research results will continue. The 3DIT toolkits studied show that this is starting to become an accepted viewpoint. Researchers are concerned about the components of interaction and how they interact not about the complexity of a particular implementation. There will be little change in the field until the field focuses on making VR toolkits a commodity option. IFFI and other toolkits show that reusability among toolkits is feasible.

Beyond VR toolkits reuse, a move towards reuse-centered development must take place. Component based architectures should become the norm, not the exception for VR toolkits. Beyond simple components, methods of component interaction must be researched and made available to others. Developers will be able to move away from low-level component development when this is possible. A new wave of research will be possible once this is accomplished. Specifically, true rapid prototyping of 3DITs may become possible. The cost of developing VE components will finally start to decrease. Large VE development costs have hampered the field of VR for too long. Component based architectures provide one way to help reduce VE creation cost when combined with the interoperability requirement listed earlier.

As a precondition for reuse, availability must also be present. Unless we move beyond sharing results solely by publications, the field will not change. Publications serve a vital role and should not be discarded. However, one purpose of publications is reproducible results. Replicating VE work from a publication may not capture subtle nuances in the implementation, and can directly influence the results of a future study. Also, replicating some VE work may take years to accomplish if it involves creating a VR toolkit. A long time frame for replication is understandable if physical work has to be undertaken, but VEs are realized in software. The cost of distributing software is negligible compared to its creation. Convention, commercialization, and intellectual property concerns are factors hindering this adoption of this practice. As a community, we should demand that software and publications be released simultaneously. Released software will not work everywhere without modification in the foreseeable future, but it will provide the foundation for allowing this to happen. In the short-term, it will provide immensely important artifacts that will help others in the field. Imagine incorporating a newly published 3DIT into a testbed without having to spend several months implementing it from scratch. Greatly reduced development time will allow researchers to focus on what is important, not on implementation details.

Description based approaches must be used in combination with lower-level implementation efforts. Alice and the 3DIT toolkits have demonstrated that this is a better method for influencing the behavior of VEs. The method for implementing behaviors is still not ready for standardization, but language used to describe them is. It may be beneficial to the community if a standardization effort similar to X3D is undertaken (Brutzman 2007). A standardized method of describing behavior will allow the coupling between behavior and implementation to decrease. Researchers and developers can focus on what is important, not on painstaking implementation details. When researchers from other disciplines can adopt VEs without the need for massive technical help, considerable new research may be generated.

Lastly, focus on the developer and end user must be at the heart of any future direction. Expecting developers to be fluent in computer graphics, parallel computation, interfacing with hardware, software engineering, and interface design will only ensure that VEs remain a small niche in society. Becoming a specialist in any of these fields is difficult. Obtaining competency in all of them cements VR's niche status. Developers and researchers must wait for a large engineering based approach to the field before new directions can be explored. Until mundane systems level work focused on reducing complexity is accomplished, developing VEs will remain difficult. One possible direction for this effort is given by 3DIT toolkits. Specifics of the actual effort will be debated over the coming years. Until this effort materializes, the benefits of focusing on the developer and end-user will remain elusive. Alice and traditional reuse provide a powerful reminder for what is possible and what the VR community should focus on.

6. Conclusion

I have presented a short history of VR toolkits, what developers experience when using a VR toolkit and contrasted this with the experience that 3DIT specific toolkits provide. Lastly, I discussed what direction the future of VE development could take.

7. References

Bowman, D., Kruijff, E., LaViola, J., and Poupyrev, I. An Introduction to 3D User Interface Design. *Presence: Teleoperators and Virtual Environments*, vol. 10, no. 1, 2001, pp. 96-108.

Bowman, D., Chen, J., Wingrave, C., Lucas, J., Ray, A., Polys, N., Li, Q., Haciahmetoglu, Y., Kim, J., Kim, S., Boehringer, R., and Ni, T. New Directions in 3D User Interfaces. *International Journal of Virtual Reality, vol. 5, no. 2*, 2006, pp. 3-14.

Brutzman D., Daly L. X3D: Extensible 3D Graphics for Web Authors. Morgan Kauffman Publishers. 2007.

Bryson, S. 1996. Virtual reality in scientific visualization. *Commun. ACM 39, 5 (May. 1996)*, 62-71.

Conway, M., Audia, S., Burnette, T., Cosgrove, D., and Christiansen, K. 2000. Alice: lessons learned from building a 3D system for novices. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (The Hague, The Netherlands, April 01 - 06, 2000).* CHI '00. ACM, New York, NY, 486-493.

Cruz-Neira, C., Sandin, D. J., and DeFanti, T. A. 1993. Surround-screen projection-based virtual reality: the design and implementation of the CAVE. *In Proceedings of the 20th Annual Conference on Computer Graphics and interactive Techniques (Anaheim, CA, August 02 - 06, 0093). SIGGRAPH '93. ACM, New York, NY*, 135-142.

Cruz-Neira, C. Bierbaum A., Hartling P., Just C., and Meinert. K. VR Juggler – An Open Source Platform for Virtual Reality Applications. *In 40th AIAA Aerospace Sciences Meeting and Exhibit 2002, Reno, Nevada*, January 2002.

Figueroa, P.; Green, M.; Hoover, H. J. InTml: A Description Language for VR Applications. *Web3D'02*, February 24-28, 2002. Tempe, Arizona, USA.

IEEE. TOC for the 2006 3DUI symposium. On the web at
http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=34551&isYear=2006

Kelso J., Arsenault, L.E., Satterfield S.G., Kriz, R.D. DIVERSE: A Framework for Building Extensible and Reconfigurable Device Independent Virtual Environments, *Proceedings of IEEE Virtual Reality* 2002.

Kessler, G., Bowman, D., and Hodges, L. The Simple Virtual Environment Library: An Extensible Framework for Building VE Applications. *Presence: Teleoperators and Virtual Environments*, vol. 9, no. 2, 2000, pp. 187-208.

Olsen. E. Cluster Juggler – PC Cluster Virtual Reality. (Masters Thesis. Iowa State 2002).

Pape. D. "A Hardware-Independent Virtual Reality Development System." *IEEE Computer Graphics and Applications, Vol 16.4, July 1996*, pp. 44-47.

Parkin, Simon. "Oculus Rift." *Technology Review* 117, no. 3 (2014): 50-52.

Ray, A. and Bowman, D. A. 2007. Towards a system for reusable 3D interaction techniques. *In Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*. S. N. Spencer, Ed. VRST '07. ACM, New York, NY, 187-190.

Ray A. (2008). The Interaction Framework For Innovation: A method to create Reusable Three Dimensional Interaction Techniques. (PhD Dissertation Virginia Tech. 2008)

Shaw C., Green. M, Liang O., Sun, Y. Decoupled simulation in virtual reality with the mr toolkit. *ACM Transactions on Information Systems. Vol. 11. 1993.* pp. 287-317.

Silicon Graphics Inc. Software Products. Performer homepage, 2008. On the web: http://www.sgi.com/products/software/performer

Stanney, Kay M. Handbook of Virtual Environments: Design, Implementation, and Applications. *Lawrence Erlbaum Associates, 2002. ISBN 080583270X, 9780805832709*

Tuukka M. Takala, Paivi Rauhamaa, Tapio Takala. Survey of 3DUI applications and development challenges. In: *IEEE Symposium on 3D User Interfaces (3DUI)*, 2012.

Tramberend, H. "Avocado: A Distributed Virtual Reality Framework." *IEEE Virtual Reality 1999*:14-21

Vanacken D., Boeck J.D., Raymaekers C., and Coninx K. 2006. *NiMMiT: a Notation for Modelling Multimodal Interaction Techniques*. *Proceedings of International Conference on Computer Graphics Theory and Applications*, 2006, pp. 224-231.

Virtools. Homepage. 2008. On the web: http://www.virtools.com

Wingrave, C.A. Bowman, D.A. Tiered Developer-Centric Representations for 3D Interfaces: Concept-Oriented Design in Chasm. *Virtual Reality Conference*, 2008. IEEE Volume , Issue , 8-12 March 2008 Page(s):193 - 200