Available online at www.sciencedirect.com

SCIENCE DIRECT

ELSEVIER

Int. J. Human-Computer Studies 62 (2005) 73–103

International Journal of Human-Computer Studies

www.elsevier.com/locate/ijhcs

Efficient comparison of platform alternatives in interactive virtual reality applications

Pablo Figueroa, Walter F. Bischof, Pierre Boulanger*, H. James Hoover

Department of Computing Science, University of Alberta, 221 Athabasca Hall, Edmonton, Alberta, Canada T6G 2E8

Received 28 August 2003; received in revised form 6 August 2004; accepted 16 August 2004

Abstract

Virtual reality applications consist of an integrated combination of elements, such as hardware devices, interaction techniques, and content, in different modalities and qualities. Designers of virtual reality applications select combinations of such elements that allow users to accomplish their tasks, and it is feasible that more than one combination of such values will satisfy the user's needs. Unfortunately, current development environments, methodologies, and techniques in the field of virtual reality often preclude the exploration of the design alternatives, due to coverage or cost limitations. A limited number of options are covered by any given software development environment, and the development cost of new prototypes in such development platforms is too high to be considered as an evaluation tool. In this paper, we present a methodology for partial (i.e. hardware and interaction techniques alternatives) exploration of the design space of a virtual reality application, based on the creation of reusable components and a standard evaluation of alternatives. Since the cost of developing several versions of an application can be reduced by reusing elements from others, this method allows designers to evaluate the performance and user preferences of several implementations. As a proof of concept, we developed four versions of a simple matching application in different virtual reality platforms. Results of this study show how users react to each prototype

*Corresponding author. Tel.: +1 780 492 7418; fax: +1 780 492 1071.
E-mail addresses: pfiguero@acm.org (P. Figueroa), wfb@cs.ualberta.ca (W.F. Bischof), pierreb@cs.ualberta.ca (P. Boulanger), hoover@cs.ualberta.ca (H. James Hoover).

doi:10.1016/j.ijhcs.2004.08.004

and how the different solutions can be compared, no matter how different in technology they are.

Keywords: Virtual reality; Iterative development; User studies in virtual reality; Interaction techniques markup language; InTml

1. Introduction

Virtual reality is a broad field that defines a novel way to interact with information, different from the traditional graphical user interfaces based on the 20 year-old Windows, Icons, Menus, and Pointers (WIMP) paradigm. Virtual reality technology allow us to see geographical, molecular, or industrial design information in the same way that we use to understand real objects in the real world. This technology can also enhance our capabilities in the real world, so that our interaction in the virtual world is more efficient than the real world. Several industrial applications exist today, in areas such as car design, military, or oil exploration, and we expect many more to appear in the future.

Virtual reality applications are not limited to the traditional keyboard and mouse, and their interfaces can offer richer control and content to users. Currently, there are several input and output devices that may be used in virtual reality applications, and several interaction techniques that exploit device affordances and particular information characteristics. Each combination of devices and interaction techniques offers different features and limitations. Such variety creates problems for application designers in both the selection of hardware for a first prototype, and the hardware and software changes required to improve a successful application. Usually, the decision about which devices should be used in an application is taken in favor of the newer technology, the one that is most compatible with current applications, or the one that is easily available, without considering other options that might be favorable for the particular application and user needs. Further, it is often difficult to port an existing virtual reality application to a newer hardware platform. Common software practices create abstractions that do not fully exploit the device capabilities, so portability is achieved at the cost of minimal functionality in all environments.

The development of virtual reality applications should take into account alternatives in key design areas, regardless of technical limitations of the available software platform. Alternatives in devices, interaction techniques, content modality, and content quality should be taken into account, in order to better fulfill user requirements. At the same time, it should be possible to produce rapid prototypes, so designers can observe how users react to each alternative, and compare results and usability of different approaches.

One way to compare hardware platforms is to do an analytical analysis of device properties. However, little information is available on the functionality and efficiency of modern virtual reality devices, as compared to standard devices. This in turn makes the process of making decisions about devices more empirical. Another way is to run controlled experiments in which a particular interaction technique or a particular device is tested, with all other variables fixed. Such experiments can show the advantages of such a device or interaction technique and can give some guidelines about its use, but they still may leave doubts about its performance in a real application, when it is combined with other elements. Our approach for the selection of a hardware platform and interaction techniques for a particular virtual reality application is to test prototypes for several feasible options, and to compare performance and other important variables by analyzing a uniform cross-platform set of data. If development cost and time of new design alternatives can be reduced, several prototypes showing different alternatives could be developed, and empirical comparisons with real users could be the best way to choose a design alternative. Section 4.1 provides references to other ideas, and how they relate to ours.

This paper introduces our methodology for design exploration in virtual reality applications. Using InTml (Figueroa et al., 2002), an architectural language for virtual reality, we can describe an application as a set of interconnected components, which can be replaced to target different setups. Our methodology describes how to incrementally develop several prototypes of an application, using different hardware and software setups, how to reuse the design of components from previous prototypes, and how to compare them in terms of performance and usability. As a proof of concept, we have implemented a simple matching application in four different platforms, and we studied user performance and preferences. We decided to make these implementations as simple as possible, to accelerate the development time for the first prototype, to test the minimal capabilities of each platform, and to allow the creation of better prototypes with the experimental findings of the first one.

We discuss first the rationale for a new language in this domain, followed by our development methodology, and an example of how to compare different virtual reality prototypes.

2. A language for design exploration in virtual reality

In order to facilitate the development of several prototypes in heterogeneous virtual reality setups, we want to make changes in virtual reality applications easier, changes related to the following main technological issues: input and output devices, interaction techniques, and content quality. Table 1 shows the coverage of these key design issues by current toolkits, and the programming language that developers should know in order to use them.

Current toolkits for virtual reality development facilitate changes in some of these areas, but they do not cover all of them, and they usually require profound programming skills. For this reason, we have created the Interaction Techniques Markup Language (InTml). InTml describes devices, interaction techniques, and content as components of a virtual reality application, in an XML-based language. It allows developers to easily find the elements they want to replace and to make

Table 1
Extension mechanisms in toolkits for virtual reality

Toolkit	Lang.	Devices	Int. Techniques	Content
MRToolkit (Shaw et al., 1992)	C	▒	░	░
CAVELib (VRCO, 2003)	C	▒	░	░
VB2 (Gobbetti et al., 1993)	Eiffel	░	▒	░
Performer (SGI, 2003)	C + +	░	░	▓
Avocado (Tramberend, 1999)	C + +,	░	░	░
Lightning (Blach et al., 1998)	C + +,	▒	▒	░
WTK (Sense8, 2000)	C + +	▓	░	▓
MASSIVE-2 (Benford et al., 1997)	C + +	▒	░	░
Alice (University of Virginia, 1999)	Phyton	░	▓	▒
VRJuggler (Bierbaum et al., 2001)	C + +	▓	░	░
X3D (Web3D Consortium, 2003)	XML	░	▒	▓
CONTIGRA (Dachselt et al., 2002)	XML	░	▓	▓
HyNet (Massink et al., 1999)	Petri Nets	░	▓	▓
PMIW (Jacob et al., 1999)	Dataflows and State Machines	░	▓	░
VRPN (Russell et al., 2001)	C + +	▓	░	░
InTml	XML	▓	▓	▒

A darker gray means a better coverage than a lighter one, from the viewpoint of a developer.

changes without knowing the intrinsic details of each component, which are written in traditional programming languages. More details about InTml can be found in Appendix A.

3. Development of design alternatives

In order to assure uniformity of measurements, we used the following process for the development of comparable virtual reality prototypes. The process is facilitated by InTml because of its reuse capabilities among heterogeneous hardware and the easy identification of changes between platforms.

We define the process of design exploration as a collaboration between people in two different roles: designers and developers. Designers are responsible for the overall architecture of an application, while developers are in charge of the detailed implementation of tasks defined by designers. Fig. 1 shows the separation of tasks between these two roles and their interrelationships. We extend the development process proposed in (Smith and Duke, 2000).

From the point of view of the designer of virtual reality applications, an InTml application is both a set of modules that have to be implemented on top of a foundation framework, and certain rules of execution that have to be taken into account. The designer's work is divided between the definition of new filters, the reuse of previously defined ones, and the definition of applications. Designers collaborate with developers, whose main job in an InTml-based environment is to

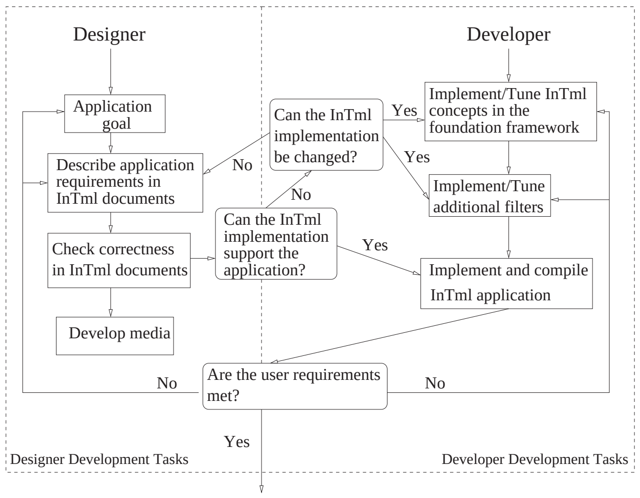

Designer
Developer
Application goal
Describe application requirements in InTml documents
Check correctness in InTml documents
Develop media
Can the InTml implementation be changed?
Can the InTml implementation support the application?
Implement/Tune InTml concepts in the foundation framework
Implement/Tune additional filters
Implement and compile InTml application
Are the user requirements met?
Yes
No
Designer Development Tasks
Developer Development Tasks

Fig. 1. Collaborative development process for a particular hardware and software platform.

develop the inner code for filters. Currently, InTml is compiled into Java3D for PC environments, and interpreted in a C++ and Performer based program for our CAVE-like environment.

Designers are in charge of the definition of application goals based on user requirements. They express such requirements in an InTml application, which can be validated against general semantic rules and also against basic usability rules (Stanney et al., 2003). At this point, developers can check the application and see if it is supported by the current implementation and libraries. If the current implementation does not cover all requirements for the new application, two tasks can be pursued: developers might change the InTml library[1] to accommodate to the application, or designers might change the application to fit the platform. Finally, the application in execution is validated against the requirements. Iteration over the previous tasks allows for an evolutionary development of the application. Application content, such as geometric models for objects, special graphic effects, sound, or haptics is designed with third-party tools. It is necessary that all created media types can be understood by the foundation framework in which the InTml application will run.

The exploration of new design alternatives is shown in Fig. 2. Collaboration refers to the relationships between designer and developer tasks in Fig. 1. New alternatives can propose more efficient interaction techniques, use of different hardware, changes in content quality, or a combination of such issues. In general, hardware, content quality, and interaction techniques are interrelated, so changes in one will trigger changes in the others.

[1]Changes to the framework might also be required, but this type of change is very rare once the prototype and the library matures.

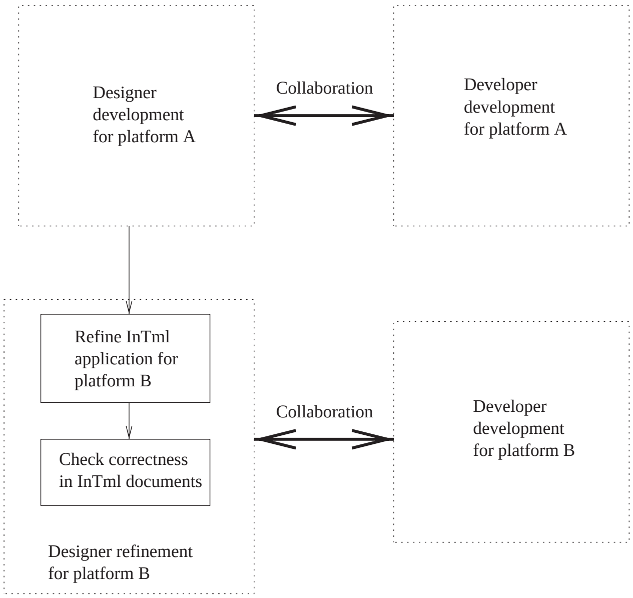

Designer development for platform A
Collaboration
Developer development for platform A
Refine InTml application for platform B
Check correctness in InTml documents
Designer refinement for platform B
Collaboration
Developer development for platform B

Fig. 2. Adapting a virtual reality application to a new hardware and software platform.

After a first version of an application in platform A has been developed, a refinement process can be led by designers in order to accommodate the tasks to a new platform. The adaptation process includes the following changes:

- Change of devices to the ones available in platform B.
- Creation of adapters in order to simulate the type of information that was received from devices in platform A. For example, if we move from a tracker-based system to a mouse and keyboard based system, an adapter can be created to simulate the tracker output by mouse and keyboard events.
- Replacement of object's behavior, interaction techniques, and widgets, for the ones that are more suitable to platform B. For example, if the selection technique in platform A is based on collision with a virtual hand, we can decide to change this in platform B to selection by intersection with a ray. This replacement might propagate changes to the entire application.
- Addition or removal of tasks, behaviors, interaction techniques, and widgets that are important in a platform. For example, the coordination of movement of the image and the head in a HMD based environment should be removed in other platforms.

Developers repeat the development process in platform B, reusing parts or designs from platform A when possible, and interact with designers until the user requirements have been met. In this way the application keeps the same functionality in both platforms, while also taking into account the particular advantages of each

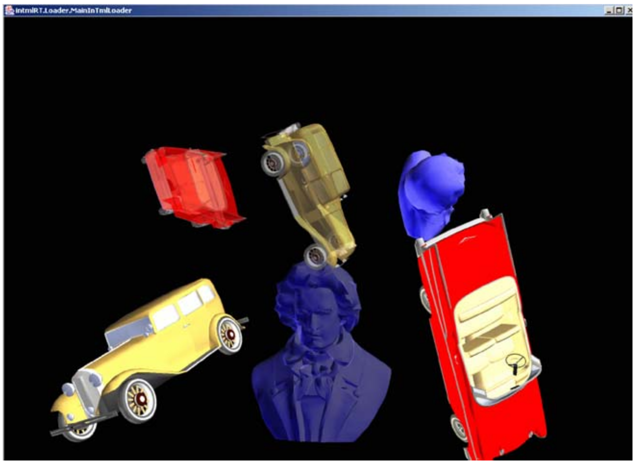


Fig. 3. Screen image of the Matching Test.

one of them. More iterations of the entire process can create evolved versions of the application with improved functionality in each platform.

This methodology allow non-programmers—called designers—to collaborate with programmers—called developers—in the creation of new virtual reality applications. Designers can specify virtual reality applications in InTml without deep understanding of the underlying implementation, and in programming languages such as C++ or Java. Applications in InTml become the blueprint for developers, so communication between designers and developers is aided by a structured language, with a clear semantics. Previous methodologies for virtual reality Development found in the literature, such as (Tanriverdi and Jacob, 2001), do not take into account the multidisciplinary nature of such work, neither the necessity for higher level virtual reality languages, in order to allow non-programmers to develop applications and to communicate their ideas to expert programmers.

4. Comparing design alternatives

As a proof of concept, we developed an application that allows users to move and rotate an object from an initial position to a target position. In our current implementation, we used three objects: a red car, a yellow car, and a blue model of Beethoven's face, as shown in Fig. 3.[2] Each object had a copy that defined the target position and rotation. In the application, the user selected an object (not its copy), grabbed it, moved it, and rotated it, until it matched its corresponding copy. Once an object and its copy were close enough, they disappeared. Objects could be moved in any order, until all of them had been matched, at which time the application ended.

[2]Models are from the public repository of Viewpoint Corporation (Viewpoint, 2003), and from the Java 3D's distribution.

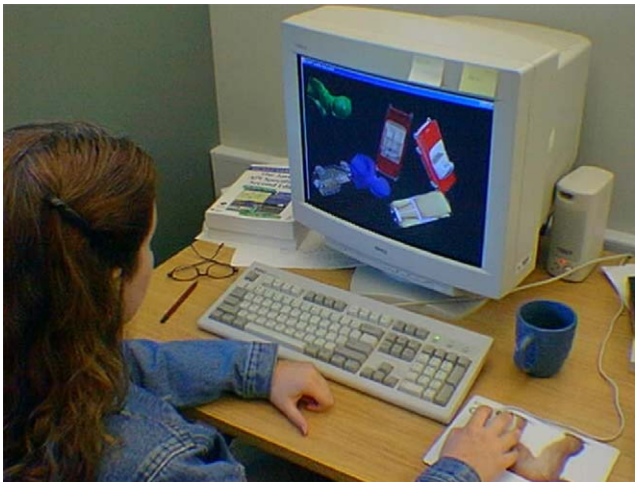


Fig. 4. Application running on a PC environment.

We kept constant the linear and angular distances in the initial position of objects, and we minimized occlusion problems by showing objects and their copies in different areas.

Our design space included the following hardware platforms that were available in our lab:

- The PC system used a standard PC interface (Fig. 4). Selection was made by ray casting. Object moving was done with three adjacent keys (z,x,c) that selected a 2D plane (XY, XZ, YZ) and the left mouse button for dragging. Rotations were achieved by dragging the right mouse button, and were limited to the X and Y axises.
- The SMART Board (SB) based system used a front projected SMART Board[3] (Fig. 5). Movements and rotations worked as in the PC version, but were activated by four pens over a touch-sensitive screen. Selection was available as a screen touch with no pen, but it was rarely used.
- The head mounted display system (HMDJ) used a low resolution head mounted display from I-O Glasses with a basic 3DOF tracker, and a standard joystick with four buttons (Fig. 6). We drew a pointer to represent the joystick position, that could move in a plane parallel to the viewport. Selection was done as in the PC version, but taking into account the joystick and the tracker movement. Moving in one of the three 2D planes was done by dragging the joystick and pressing a button. The fourth button was used for rotations, as in the PC version.
- The Space Mouse based system (3DD) used four keys on the keyboard, a Space Mouse from Logitech, and a standard display, as seen in Fig. 7.[4] The Space Mouse controlled the 3D position and orientation of a pointer on the screen. Selection

[3]SMART Board is a trademark of SMART Technologies Inc.

[4]The cameras in the picture are part of a different experiment.

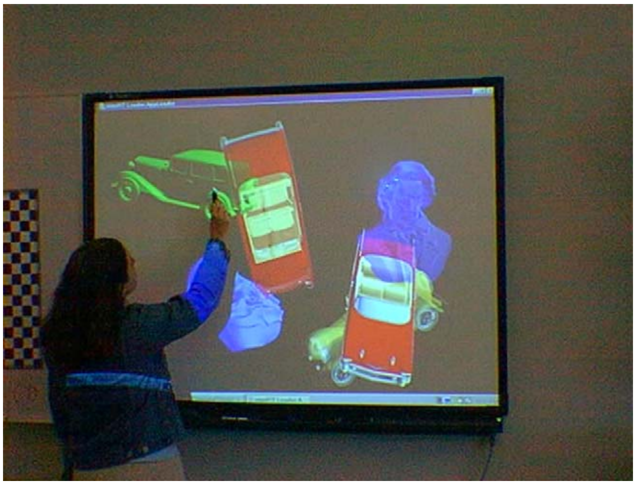


Fig. 5. Application running on a SMART Board.

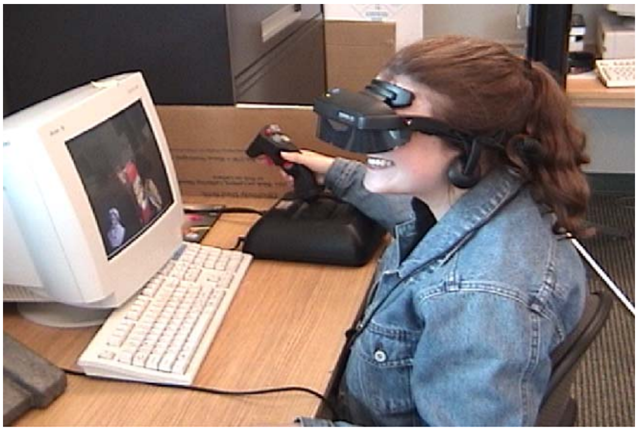


Fig. 6. Application running on a HMD plus Joystick.

was done by colliding the pointer with an object. There was a key for grabbing an object, one for rotating objects, and one for moving objects.

The implementation of these versions reused eight tasks: object loading and drawing, viewpoint positioning, creation of object copies, random positioning of objects, selection feedback, object matching, overall control of the application, and log facilities. Elements such as input events, selection techniques, and grabbing techniques, had to be changed for each environment in order to use the particular affordances of the available devices.

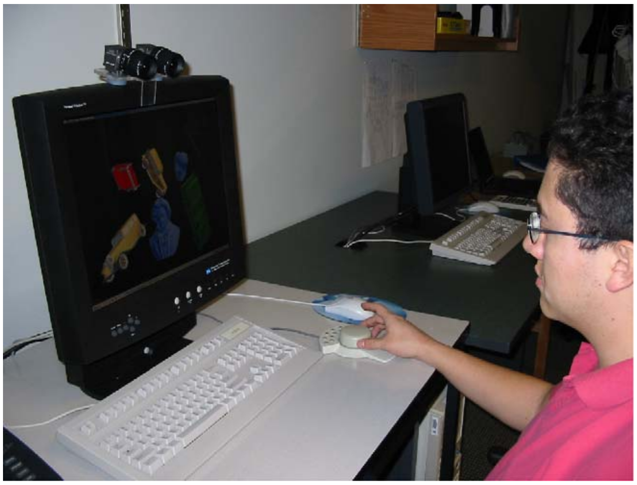


Fig. 7. Application running on a PC plus 3D Mouse.

One developer was involved part time in the creation of these four prototypes, and they were finalized in a consecutive fashion, in the following order: PC, SB, HMDJ, and 3DD. Changes to each platform affected the reusable components, so the first prototypes were constantly improved. Although we do not have information about time spent in each task during the development of our example application, we can extract some information from the CVS repository used by developers. In general, the development of the four versions took 6 months for two developers, with some interruptions during that time.

One way to describe the amount of work per application version is to see changes in files registered in our CVS-based repository (addition and removal of lines). Although this information is limited since the initial file size is not recorded in the CVS reporting tool and some classes are more important than others in the application, such changes give a good idea of the amount of effort dedicated per hardware platform. Fig. 8 shows changes in number of lines of code over time, in files grouped by hardware platforms (*All* refers to files that were used in all versions). At each point in time we take the number of lines added plus the number of lines deleted in all files related to a particular version.

We notice that general files dominate changes over time, which means that most of the effort in development is shared among several versions of the application. Peaks show some deadlines for deliverables in each platform, starting with the PC and SB platforms, followed by HMDJ and 3DD. We also notice that the later versions required fewer changes (or fewer time) to complete than the previous ones. This indicates that the cost of producing new versions is reduced over time.

The last peaks in the *All* curve are related to the log mechanism of the applications. All versions log information about the user's experience in a centralized database, and such functionality required several changes over time.

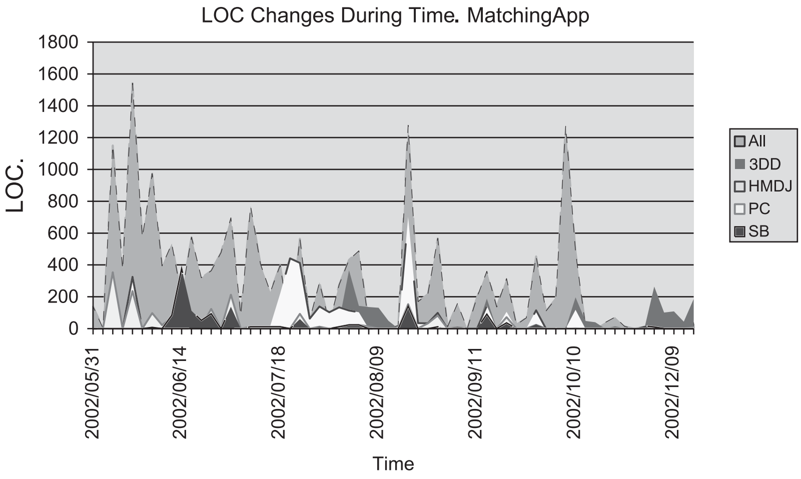


Fig. 8. Changes in lines of code per hardware platform.

4.1. *Comparison metrics*

Many user studies in virtual reality have concentrated on the systematic analysis of interaction techniques, devices, and content characteristics. Some have studied variations of a particular application, and some have concentrated on the reactions that people have to this technology. Our goal is to define metrics and methods to compare all these factors at once, for a particular application and user community. The following paragraphs give more detail about these studies, and about the metrics and supported data we use in our study.

Several studies have compared interaction techniques, such as comparisons of manipulation techniques (Zhai and Milgram, 1993; Poupyrev et al., 1997), grabbing techniques (Bowman and Hodges, 1997a), and travel techniques (Bowman et al., 1998). Such comparisons usually create environments with parameters relevant to the particular set of interaction techniques in the study, and they typically use a fixed set of devices. Other studies have compared a smaller subset of interaction techniques in order to show advantages and disadvantages in more detail, such as a virtual hand versus a virtual pointer (Poupyrev et al., 1998), *HOMER* versus *Vodoo Dolls* (Pierce and Pausch, 2002), or speed-coupled flying with orbiting versus others (Tan et al., 2001).

Several papers have introduced new devices, such as (Sharlin et al., 2000) and (Hinckley et al., 1994), some with detailed user studies about their benefits and characteristics (e.g. (Ware and Balakrishnan, 1994; Mason et al., 2001; Myers et al., 2002)), and some with comparison with other devices in a particular task (Hinckley et al., 1997). Other papers have been devoted to the study of small changes in devices for an application, such as a joystick versus a stylus (Bowman and Hodges, 1997b), a 3D-ball versus a tracker (Hinckley et al., 1997), two hand control versus one hand (Balakrishnan and Kurtenbach, 1999), or eye movements versus a pointer

(Tanriverdi and Jacob, 2000). Although these studies provide good design guidelines for new developments in the area, they are of limited use in real applications where a device is not used in isolation.

Some papers have shown studies of several implementations of interesting virtual reality applications in areas such as 3D model design (Foskey et al., 2002) or non-linear drama (Craven et al., 2001), but they have considered only a limited number of implementations, and their analysis techniques have been very subjective.

There are also some studies on the effect of virtual reality environments on humans, e.g. a study of object rotation (Ware and Rose, 1999), a study of role of kinesthetic reference frames in two handed input performance (Balakrishnan and Hinckley, 1999), or a study on nausea effects of navigation techniques (Howarth and Finch, 1999). Again, very useful guidelines can be extracted from these studies, but they have to be validated against real users in any new application.

However, in a real application, all interaction techniques, devices, and people act together, and findings for each isolated factor are affected by the relationship with the others. Hence it is difficult to directly apply previous studies since they do not show the effect of factor combinations. Previous results are very important as guidelines, but a set of previously tested devices, and interaction techniques have to be tested together in order to understand their relationships. Kjeldskov (Kjeldskov, 2001) presents an example of correlation between interaction techniques and display devices and we attempt to show another example of this type of studies

We use objective and subjective metrics to compare several versions of an application. There are many ways to capture subjective information from the user, but we decided to use questionnaires, because they can be filled on-line and they are easily processed. There are many examples of questionnaires for user evaluation in virtual reality applications, e.g. (Slater et al., 1998; Witmer and Singer, 1998), and more general ones applicable to any interface (Chin et al., 1988). Despite known disadvantages, e.g. a lack of correlation with reality (Usoh et al., 2000), we consider them useful for comparing different implementations of a virtual reality application.

We collected the following standard set of events for all platforms:

- SEL and DES: Selection and de-selection of objects.
- MOV_XY, MOV_XZ, MOV_YZ: Change in the plane of movement, applicable in all platforms except 3DD.
- RI and RE: Subject starts or stops an object rotation.
- TI and TE: Subject starts or stops an object movement.
- INTML_EXEC_TIME: Time for the execution of tasks in InTml, in milliseconds.
- MATCH_SIGNAL: Event received when an object and its copy are close enough so they are considered a match.
- END_SIGNAL and ABORT_SIGNAL: Events received when the application finishes or its aborted.
- MOV: An object has moved.
- ROT: An object has been rotated.

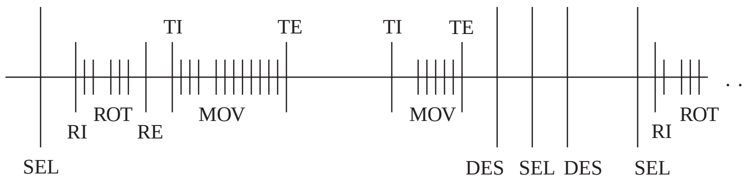


Fig. 9. Order of events throughout time.

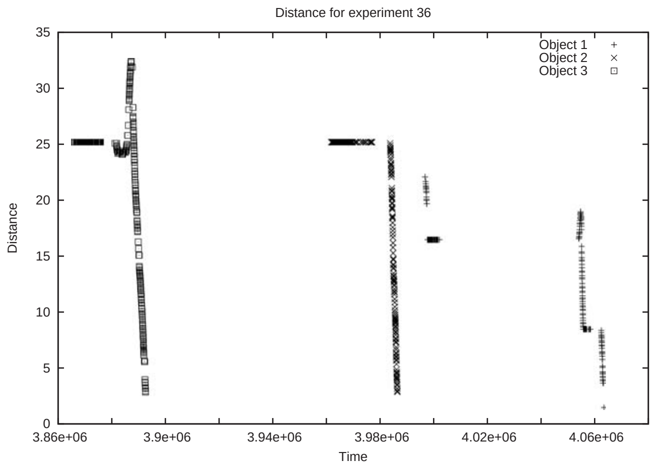
Some events cannot be generalized, such as the MOV_* group, since such events do not exist in the 3DD platform. Others behave differently in different platforms, such as the RI/RE/TI/TE: In the 3DD platform, both groups are simultaneous to the SEL–DES events, since once an object is grabbed, it can be rotated and translated at the same time.[5] Still these events allow the development of a set of uniform metrics and an objective comparison of different implementations. We define the following metrics, based on the raw events described above: time for object matching, number of control events per object, distance error, orientation error, and preparation time. Details of such metrics are described below.

The events generated by an application are temporally organized as shown in Fig. 9. Objects are manipulated in selection intervals, the intervals that start with a SEL and end with a DES. We define the time for object matching as the sum of all selection intervals dedicated to a particular object, intervals that might be intermixed and sparse, but are always disjoint. The number of control events per object is the sum of all control events, MOV_XY, MOV_XZ, MOV_YZ, RI, RE, TI, TE, inside the selection intervals of an object.

The distance error (DE) is defined in terms of the MOV events. Every time such an event is generated for a particular object, we save the distance to its copy, and we consider this distance fixed during the period of time given by the MOV event and the next one, or the end of the selection interval. The distance error can vary substantially, as is illustrated in Fig. 10. Ideally, the user's interaction will steadily decrease the distance between an object and its copy. However, errors in the user's interpretation or in the use of the interface can create a more erratic behavior, as illustrated in the right panel of Fig. 10. We take the number of times that the distance increases as a measure of the distance error in an experiment. This measure can be computed in several ways, depending on how fine-grain detail changes are considered, and how important the errors are. Here, we use every third point, and we record any increase in distance. All other measure techniques we used gave similar results. The orientation error (OE) is defined in a similar way, except that every time a ROT event is received, we consider the angle between the quaternions that describe the current orientation of an object and its copy.

If there are no objects selected, we assume that the user is preparing an interaction with the system. The preparation time related to an object is the sum of all preparation intervals that precede selection intervals for this object.

[5]We also use these groups to mark the tasks' dis-entanglement in the 3DD environment.



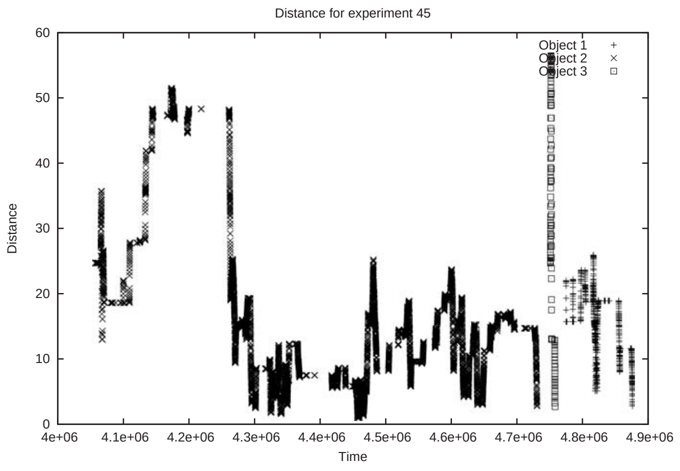


Fig. 10. Distance error over time.

4.2. Method and apparatus

We conducted an experiment in which participants used one of four versions of an application running on different hardware platforms. 42 participants volunteered for the experiment, eight women and 34 men between 21 and 30 years. They were divided randomly in four groups, one for each hardware setup. Three participants did not finish the experiment, and we did not consider their data in the following analysis, so there were 10 participants for all experimental conditions except one. Participants read a general introduction to the experiment and filled a form with general information about themselves, inspired by the QUIS evaluation (Chin et al., 1988; Shneiderman, 1998) and shown in Appendix B. Users were classified according to two factors: immersive tendency and previous 3D experience. We considered that

Table 2
Participants' previous experiences

Platform	Immersive tend.	3D experience
PC	6.05 ± 0.47	90%
SB	5.5 ± 0.37	80%
HMDJ	5.97 ± 0.38	90%
3DD	5.7 ± 0.51	89%

Table 3
Average InTml and Java3D frame rates (frames/s)

Platform	InTml frame rate		Java3D frame rate	
	Mean	SE	Mean	SE
PC	55.2	2.1	52.4	3.3
SB	18.9	3.2	57.4	0.5
HMDJ	32.2	1.8	36.0	2.6
3DD	64.4	8.9	31.9	3.7

participants have had previous experience with 3D information if they selected any of the following options in the first questionnaire (Appendix B): Game consoles, HMDs, Tracking devices, Stereo glasses, Passive Stereo Displays, 3D Mouse, CAD, or 3D Video Games. Table 2 shows the averages of answers related to section "Immersive Tendency" in the first questionnaire (Appendix B), and previous experience as stated above.

We guided the participants while they were matching the first object, and they worked autonomously with the other two objects. Data collected during the interaction with the first object is not included in the following analysis.

After the experiment was completed, performance data was collected and participants were asked to fill a second form, a short version of the one designed by Witmer and Singer (Witmer and Singer, 1998) (see Appendix C) about their experiences and personal opinions. Participants took about half an hour to complete the whole experiment.

4.3. Results

We discuss here the general results of the experiment, the event-related results, and the subjective results of the questionnaire evaluation. In the following sections, data of the general and event-related results were analysed using analysis of variance (ANOVA) designs, either one-way designs or mixed (repeated-measures) designs. For testing specific comparisons, Tukey–Kramer tests were used with a significance level $\alpha = 0.05$.

Table 4
Average time per experiment

Platform	Time (s)	SE
PC	714	147
SB	645	62
HMDJ	456	55
3DD	585	56

4.3.1. General results

Table 3 shows means and standard errors[6] of the InTml frame rate, derived from the average of execution time of the InTml code, and of the Java3D frame rates, derived from the mean of the difference between start times of InTml frames.

For the InTml frame rate, a one-way ANOVA with platform as factor showed a significant effect
($F(3,35) = 20.1$, $p < 0.001$), and Tukey–Kramer tests showed that platforms PC and 3DD were different from platforms SB and HMDJ. For the Java3D frame rate, the ANOVA also showed a significant effect ($F(3,35) = 9.9$, $p < 0.001$), and Tukey–Kramer tests showed that platforms PC and SB were different from platforms HMDJ and 3DD.

The InTml frame rates for the PC and 3DD environments are enough to keep the Java3D frame rates up to speed. The implementation of the SB and HMDJ environments should, however, be faster in order to get at least one update per Java3D frame.

Table 4 shows the average experiment duration for each platform. Although an ANOVA with platform as a factor showed no significant effect ($F(3,35) = 1.50$, $p > 0.1$), data show that average duration tends to be longest on the PC platform, and tends to be shortest on the HMDJ platform, despite the lower frame rate of the HMDJ platform.

4.3.2. Event-related results

Most users started the task with the Beethoven face, matched the yellow car second, and matched the red car last. Table 5 shows the time for matching the second and third object for each platform. A mixed ANOVA with platform as between-subjects factor and object (second or third) as within-subjects factor showed neither an effect of platform ($F(3,35) = 1.67, p > 0.1$) nor of object ($F(3,35) = 1.76, p > 0.1$). We notice, however, that there is a tendency for matching times to decrease between the second and third object matching.

Further, there is a consistent trend in matching time for different platforms. Ordered from faster to slower matching time, we have the following order of platforms: 3DD, SB, HMDJ, PC. It is interesting to notice the good performance of

[6]The standard error is defined as the standard deviation divided by the square root of the number of samples.

Table 5
Average time of matching the second and third object

Platform	Second object		Third object	
	MEAN (s)	SE	MEAN (s)	SE
PC	262.6	103.0	134.8	29.4
SB	119.3	18.9	100.8	23.8
HMDJ	129.2	36.3	128.2	25.4
3DD	106.8	27.2	98.1	22.1

Table 6
Average number of control events per platform

Platform	Second object		Third object	
	MEAN	SE	MEAN	SE
PC	940.9	549.8	619.3	186.6
SB	63.3	11.8	62.6	10.5
HMDJ	26.5	3.8	26.9	2.9
3DD	4.2	2.5	3.8	1.8

the SB platform, despite the slower hardware that it uses. The PC platform performs slower than the others, despite the fact that users were already familiar with the devices used on this platform.

Table 6 shows the number of control events per platform. A mixed ANOVA with platform as between-subjects factor and object as within-subjects factor showed a significant effect of platform ($F(3,35) = 4.63, p < 0.01$). Tukey–Kramer tests showed that the number of control events for the PC platform was significantly different from all other platforms, but the other three platforms did not differ from each other.

3DD tends to have the lowest number of control events, mainly due to the absence of MOV_* events. Further, despite the uniformity of interaction techniques in the SB and PC platforms, SB has significantly fewer control events than PC.

Table 7 shows distance errors, defined by the number of times the user moves in the wrong direction (see Section 4.1). A mixed ANOVA with platform as between-subjects factor and object as within-subjects factor showed a significant effect of platform ($F(3,35) = 4.26, p < 0.05$), and no other effect was significant. Tukey–Kramer showed that distance errors for platform SB were significantly different from platform 3DD.

Table 8 shows orientation errors, defined as the number of times the user takes the wrong direction for rotation. A mixed ANOVA with platform as between-subjects factor and object as within-subjects factor showed a significant effect of platform

Table 7
Average distance error for each platform

Platform	Second object		Third object	
	MEAN	SE	MEAN	SE
PC	20.7	11.3	10.1	2.6
SB	3.1	1.0	3.7	0.9
HMDJ	5.3	0.9	5.1	1.4
3DD	13.4	3.2	21.0	2.4

Table 8
Average orientation error, per platform

Platform	Second object		Third object	
	MEAN	SE	MEAN	SE
PC	76.4	25.9	52.5	15.1
SB	15.7	3.4	15.9	3.1
HMDJ	28.2	14.6	36.9	6.1
3DD	13.1	4.3	23.1	4.7

Table 9
Average preparation time per platform

Platform	Second object		Third object	
	MEAN (s)	SE	MEAN (s)	SE
PC	16.4	3.7	11.8	2.1
SB	104.9	13.9	92.4	17.8
HMDJ	22.0	5.2	29.6	6.0
3DD	94.7	35.4	68.5	25.9

($F(3,35) = 5.33$ $p<0.01$), and no other effect was significant. Tukey–Kramer tests showed that this effect was due to the differences PC-SB and PC-3DD.

Finally, Table 9 shows the preparation time per platform. A mixed ANOVA with platform as between-subjects factor and object as within-subjects factor showed a significant effect of platform ($F(3,35) = 17.55$, $p<0.001$), and no other effect was significant. Tukey–Kramer tests showed platforms PC and HMDJ were significantly different from platforms SB and 3DD.

Table 10 shows a way to summarize previous data for all platforms, according to each one of the above measures. For each object in each platform we assign a

Table 10
Summary of previous measurement ranked per platform

Platform	Matching time	Control events	Distance errors	Orientation errors	Prep. time	Average ranking
PC	4	4	3.5	4	1	3.3
SB	2	3	1	1.5	4	2.3
HMDJ	3	2	2	3	2.5	2.5
3DD	1	1	3.5	1.5	2.5	1.9

Table 11
Averages of user answers in the surveys

Platform	User reactions	Screen	Learning	System capabilities	Sense of presence
PC	5.40 ± 0.36	8.33 ± 0.33	5.12 ± 0.50	7.49 ± 0.29	5.96 ± 0.38
SB	5.58 ± 0.51	7.73 ± 0.31	5.84 ± 0.24	5.36 ± 0.57	5.75 ± 0.43
HMDJ	6.27 ± 0.50	6.97 ± 0.41	5.52 ± 0.51	7.25 ± 0.43	6.41 ± 0.23
3DD	6.41 ± 0.65	8.12 ± 0.23	6.30 ± 0.94	7.08 ± 0.60	5.82 ± 0.51

number between 1 and 4 according to their relative preference, 1 for the best option and 4 for the worst. In case of contradictory results between the second and third matched object, we assign the mean value. On average, the HMDJ, SB, and 3DD platforms outperform the PC. Again, it is interesting to note the good performance of the SB platform despite the slower hardware it has.

4.3.3. *Subjective results*

Table 11 shows average responses for each platform and for each group of questions regarding user reactions, screen, learning, system capabilities, and sense of presence. (See Appendix C for all questions). One-way ANOVAs showed a significant effect of platform for the screen questions ($F(3,34) = 1.07$, $p < 0.05$) and for the system questions ($F(3,35) = 4.19$, $p < 0.05$), but not for the other questions. For the screen questions, Tukey–Kramer test showed a significant difference between platforms PC and HMDJ, and for the system questions, the differences PC−SB and SB−HMDJ were significant.

Some users complained about the difficulty of the interaction techniques in the PC and SB environments, especially for rotations. Users in the HMDJ platform asked to map rotations in the Z-axis to the joystick's twist capability. Users tended to like the screen appearance of the PC best, and that of the HMDJ least. Users complained about the screen size in the HMDJ environment, and about the visual feedback for selectable objects, specially in the 3DD environment[7]. With respect to learning, users

[7]In the 3DD environment, users checked whether an object was selectable by moving the pointer to the object's position, which is a slow operation.

did not find any interface particularly intuitive, and they tended to dislike the PC interface most. Subjects in HMDJ and PC tended to like more the system capabilities than subjects in SB and 3DD environments. Finally, presence ratings were in general very low, with a slight, but not significant, advantage for the HMDJ platform. Users complained about the interaction techniques, the front projection system in the SB, the mapping of the joystick in the HMDJ, and the jittering of the tracker in the HMDJ.

From the statistically significant results of this subjective data we can conclude that users have a slight preference for the PC environment, followed by the HMDJ. An analysis similar to that in Table 10, with all the information from the users, revealed that users have a slight preference for the SB environment. It is also interesting to note that despite the fact that the PC and HMDJ environments were comparable from the user's point of view, the HMDJ had a better objective ranking. For this reason, both objective and subjective metrics should be taken into account in the development of new prototypes.

5. Discussion

Experiments with the four prototypes show some interesting results. Frame rates were low in SB, as can be expected with the slower machine it uses. HMDJ also had low InTml and Java3D frame rates, possibly caused by device drivers for the orientation tracker or display. Users in the HMDJ platform completed the task faster despite its low frame rate, which makes this platform very promising in terms of user performance. Matching times for the 3DD and SB were comparable and better than the ones in PC and HMDJ. Users took probably more time to prepare in these environments, and they were more efficient when they were interacting. This conclusion is also supported by the number of control events generated in the PC platform compared to the others: PC users generated many more control events than users of the other platforms. We believe that users were more comfortable and familiar with the PC platform, so they allowed themselves more interaction mistakes than in the other environments.

Users produced larger distance errors in the PC and 3DD platforms than in the other two. We believe that they took more time to interact and move the objects in these two environments, and consequently the distance errors were bigger. There are more orientation errors than distance errors, suggesting that users spent most of the time rotating objects, instead of moving them.

Users spent more time preparing in the SB environment than in the other ones, suggesting that the characteristics of this environment, bigger display and slower machine, affected their performance. However, increased preparation time resulted in lower number of errors and shorter matching times.

From the questionnaires we see that users reacted more critically to the 3DD environment than to the others. Our believe based on their comments is that the selection technique was the cause of such reaction. Screen quality tended to be better on the PC than in the HMDJ, as it can be expected due to the lower resolution of the later, but it was comparable to the one in the SB. It is interesting that screen rates between SB and PC environments were similar, despite the size difference between them. Maybe resolution and projection occlusion were factors against the SB implementation.

Users in 3DD complained about lack of visual feedback, despite the fact that all platforms used the same approach: the copy of an object was not selectable, only the original, so any attempt to select a copy did not give any feedback. We believe this complaint is justified in terms of the slower process of selecting an object in the 3DD platform, since users have to move a pointer in 3D until it collides with an object. This can take more time than simply moving a 2D pointer in the display plane and using ray casting for selection, so the slower the process the more noticeable were problems with feedback. Finally, users tended to rate the HMDJ better in terms of presence, maybe because vision with the HMD was limited to the virtual world, so distractions were avoided.

We can use this information in several ways. The first one, our main purpose stated at the beginning of this paper, is to decide the best implementation for a particular task. In this case, we can give weights to the different qualitative and quantitative results we collected in order to decide which prototype fulfills our conditions. By following this method we are inclined to prefer our SB implementation. A second use, now evident, is the roadmap of improvements that results from the comparison information we gathered. If we want to keep different interfaces for this application we can now concentrate on improving frame rate counts of the SB and HMDJ implementations, the interaction technique for orientation and moving in the PC platform, or the orientation techniques in the HMDJ, among others. This information gives us a clear goal in each platform—improve in comparison with the others—and a list of findings in which we can concentrate on in future iterations of the methodology described in Section 3.

6. Conclusions

We have described a methodology for development of design alternatives in virtual reality applications, and we have developed in Java four versions of a simple application as a proof-of-concept. We have shown how several implementations of a 3D application can be developed and compared using objective and subjective data. Our results show that the environment that users prefer may not necessarily be the one in which they perform best. The results also show that implementations on slow machines can compete with faster ones, at least at a prototype level for the interaction techniques and object's behavior. In summary, the development of several prototypes of an application, instrumented by standard metrics among them, can help understanding the best implementation for a given population.

We plan to further test and refine our current methodology by running similar studies with different applications. We also plan to add more platforms and programming paradigms (i.e. descriptive programming instead of procedural programming) to this study, to do more studies about controlled improvements of this application, to improve our metrics for the development effort of different prototypes (LOC changes is rather limiting for this purpose), and to compare specially tuned implementations in these platforms.

Acknowledgments

Special thanks to Robyn Taylor, our research assistant in the development of this application; and to Gerald Häubl from the School of Business at the University of Alberta, for the discussions about this and other experiments. We thank the anonymous reviewers for their extremely helpful comments and suggestions.

Appendix A. An introduction to InTml

InTml considers a virtual reality application as a data flow of interconnected filters. Filters are the building blocks which describe the standard connections for any of the following entities: input or output devices, interaction techniques, object behavior, animations, geometric objects, or other media objects. Details about the gathering of information from devices or about object behavior code are described elsewhere through the use of programming languages. Further, geometry or other media types related to content are developed in any of the available tools for that purpose, such as Maya (Alias Wavefront, 2003), 3D Max (Discreet, 2003), or Blender (Blender.org, 2003). InTml is a language for integrating all elements in a virtual reality application, not for specifying details, enabling designers to concentrate on the architecture of the application and interaction issues without dealing with too much complexity. Whereas dataflow-based languages such as VRML focus on the description of geometry and animation, InTml focuses on application specific behavior, object behavior, gathering of input device information, and on integration.

A *filter* represents any device, interaction technique, behavior, or content in a virtual reality application. A filter's interface is defined in terms of input ports, the type of events it can receive, and output ports, the type of information it produces. Some input ports can be considered parameters, which receive information only once, at application startup. A filter can have an internal state, which is important for predicting the filter behavior, but it is described at the architectural level, due its low-level nature. Fig. 11 shows a way to represent a filter, `SelectByTouching`, with input ports on the left of a box and output ports on the right. In this particular example, the output port is a selected object from the scene, and the input ports are the object used as hand representation, the current position and orientation of such an object, the scene of objects to pick from, and the events that inform about added or deleted objects from the scene. The input ports for the hand representation and the scene can be considered parameters of this filter.

A filter computation is divided into three stages, data collection, processing, and output propagation. In data collection, all information generated in a certain time

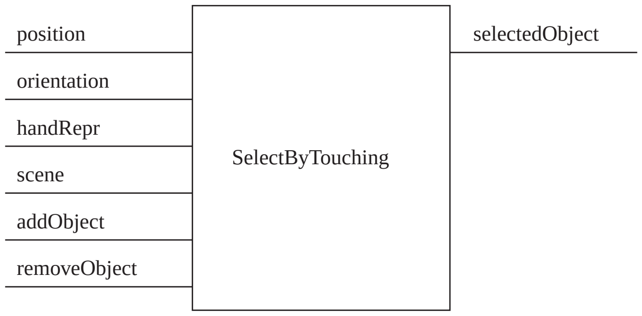


Fig. 11. Graphical representation of the SelectByTouching filter.

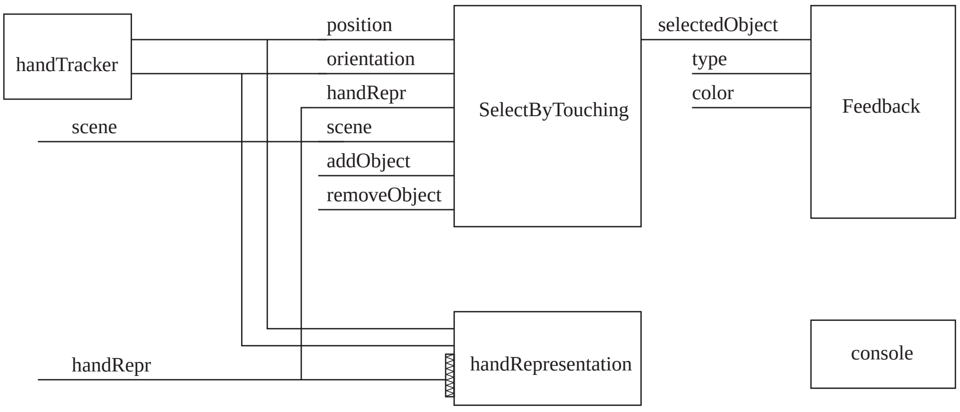


Fig. 12. Graphical representation of an InTml simple application. Touching objects with a virtual hand.

frame is collected. Some filters can do some information preprocessing, before the actual filter processing begins. A filter executes in the processing stage, given its internal state and the processed input information. Output information is generated, but not propagated. Finally, information is propagated to all interested filters in the output propagation stage.

An *application* is a set of interconnected filters, that meet certain user requirements. Fig. 12 shows a simple application, which allows a user to move a virtual hand with a tracker and to touch virtual objects. In this example, a device (`handTracker`) gives position and orientation information to a selection technique (`SelectByTouching`) and to an object holder (`handRepresentation`), which is a filter that associates one or more objects to a set of desired changes, in this case, movements and rotations. Object holders are graphically identified by the special decoration in one of their input ports, the one that receives new objects to be hold, i.e. `handRepr` in this case. The actual object representing the user's hand (`handRepr`) is given to `SelectByTouching` for collision detection, and to `handRepresentation` for changing the object. Once a collision is detected, the

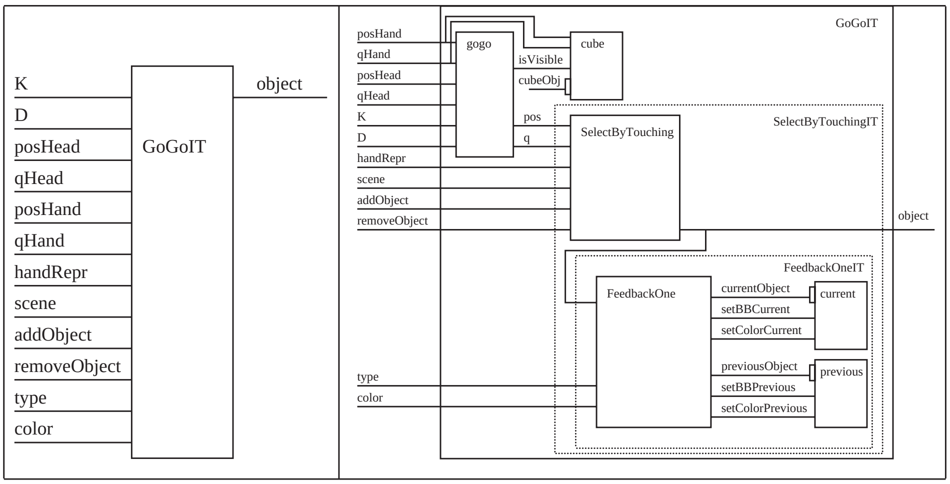


Fig. 13. The Go-Go interaction technique. General and detailed views.

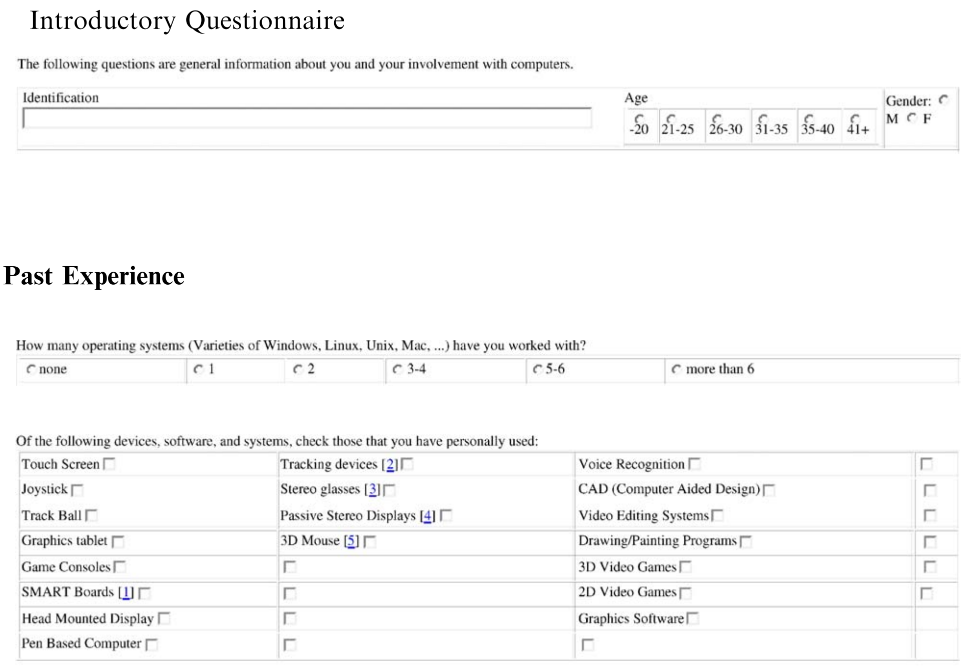
collided object is passed to `Feedback`, which changes the color of the object. Filters and applications are independent of any particular software framework and hardware, so the designer is not limited by platform-specific elements, and the developer is free to reorganize the implementation in order to improve the performance of the application in a particular platform. Filter ports can be left unconnected, which means that they will not change in a particular application. For example, ports such as `type` and `color` in the `Feedback` filter are not connected, so the filter will not change such values and it will use some internal values by default.

The *objects* represent identifiable pieces of content in the virtual environment, elements that can be seen, heard, or touched by the user. An *object holder* is a special type of filter that allows indirect references to objects. An *input device* is a filter with output ports that send events of a certain type to the dataflow. An *output device* is a placeholder that describes where the output of the application will be shown—it is internally related to the objects, but the details are hidden from the designer. Output devices are implicitly connected to all objects in the application, since they should be rendered, but such connections are not shown in an application diagram. For example, the `console device` in Fig. 12 does not have connections, which means that it does not require parameters from the environment. It will render all objects in the scene. Information in the dataflow is considered immutable, and so are the states of objects in a particular time frame, hence all filters see the same state of an object inside a computation frame.

In order to reduce the complexity of an application, subsets of interconnected filters can be encapsulated in a *composite filter*. A composite filter represents a complex behavior in an application, that might be treated as a unit and reused in new applications. Composite filters can be used to encapsulate all necessary details of an interaction technique. For example, Fig. 13 shows two views of the Go–Go interaction technique (Poupyrev et al., 1996), an interaction technique to lengthen the user's virtual arm for reaching distant objects: A simple view for designers with just its interface, and a detailed view for developers with its internal structure.[8]

An implementation of an InTml application executes four main tasks in every frame: device polling, data flow execution, object modification, and rendering. Device polling reads information from all devices connected to the hardware platform during a certain period of time. Data flow execution propagates the events to all filters in the application, but simply queues all object changes, so that all filters in execution see the same scene graph state. Object modification executes all requested changes, and rendering renders the new scene on each of the available output devices. These tasks can be parallelized or pipelined, so it is possible to get the best performance for each platform with only one application description. An InTml application describes the first two tasks, device polling and data flow execution, whereas the other two, object modification, and rendering, are hidden from the designer. Again, since the rendering step is implicit for the designer, media objects do not have to be connected to output devices, as Fig. 12 shows.

Appendix B. User information questionnaire



Introductory Questionnaire

The following questions are general information about you and your involvement with computers.

Identification	Age	Gender:
	○ -20 ○ 21-25 ○ 26-30 ○ 31-35 ○ 35-40 ○ 41+	○ M ○ F

Past Experience

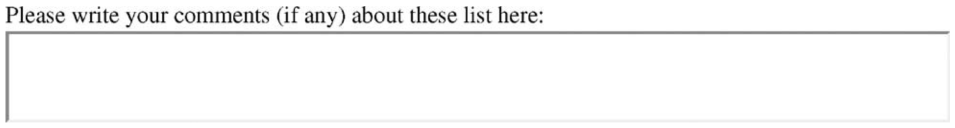
How many operating systems (Varieties of Windows, Linux, Unix, Mac, ...) have you worked with?

○ none	○ 1	○ 2	○ 3-4	○ 5-6	○ more than 6

Of the following devices, software, and systems, check those that you have personally used:

Touch Screen ☐	Tracking devices [2] ☐	Voice Recognition ☐	☐
Joystick ☐	Stereo glasses [3] ☐	CAD (Computer Aided Design) ☐	☐
Track Ball ☐	Passive Stereo Displays [4] ☐	Video Editing Systems ☐	☐
Graphics tablet ☐	3D Mouse [5] ☐	Drawing/Painting Programs ☐	☐
Game Consoles ☐	☐	3D Video Games ☐	☐
SMART Boards [1] ☐	☐	2D Video Games ☐	☐
Head Mounted Display ☐	☐	Graphics Software ☐	
Pen Based Computer ☐	☐	☐	

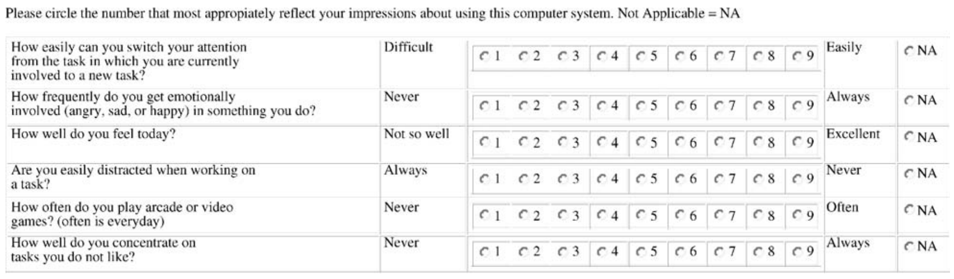
[8] In Fig. 13, `K` and `D` refer to two parameters of the Go–Go Technique, and `q` refers to orientation data.


Please write your comments (if any) about these list here:

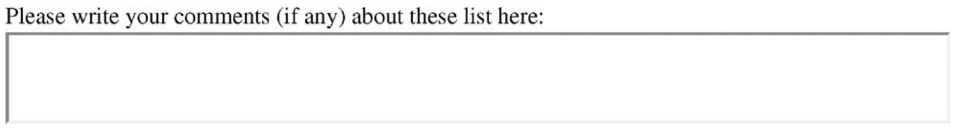
Description

[1] A SMART Board is a device produced by SMART Technologies for computer based presentations.
[2] A Tracking device allow the computer to know the position and orientation of the user's hand or head.
[3] A stereo glasses device allows an user to see a special image in 3D.
[4] A Passive stereo display allow users to see images in 3D without wearing special glasses.
[5] A 3D mouse allow an user to move a pointer in 3D.

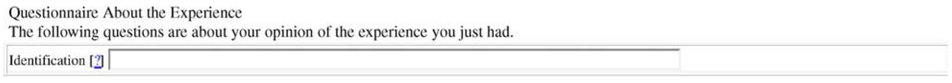
Immersive Tendency


Please circle the number that most appropiately reflect your impressions about using this computer system. Not Applicable = NA

How easily can you switch your attention from the task in which you are currently involved to a new task?	Difficult	○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9	Easily	○ NA
How frequently do you get emotionally involved (angry, sad, or happy) in something you do?	Never	○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9	Always	○ NA
How well do you feel today?	Not so well	○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9	Excellent	○ NA
Are you easily distracted when working on a task?	Always	○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9	Never	○ NA
How often do you play arcade or video games? (often is everyday)	Never	○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9	Often	○ NA
How well do you concentrate on tasks you do not like?	Never	○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9	Always	○ NA


Please write your comments (if any) about these list here:

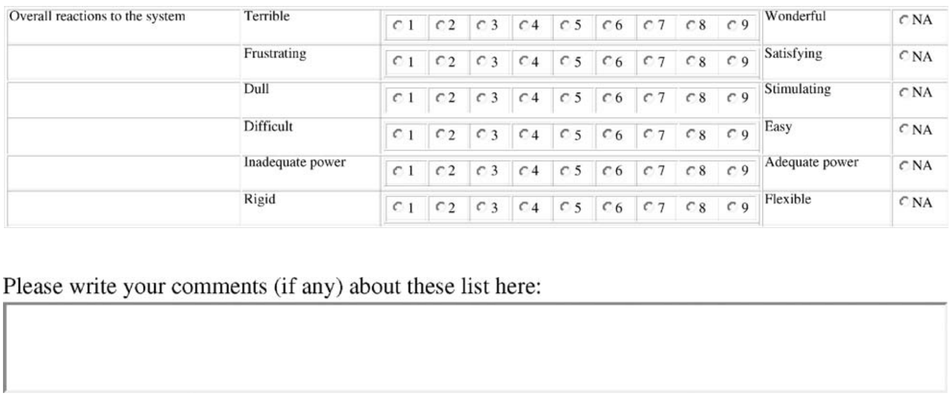
Appendix C. User experience questionnaire


Questionnaire About the Experience
The following questions are about your opinion of the experience you just had.

Identification [?]

Please circle the number that most appropriately reflect your impressions about using this computer system. Not Applicable = NA

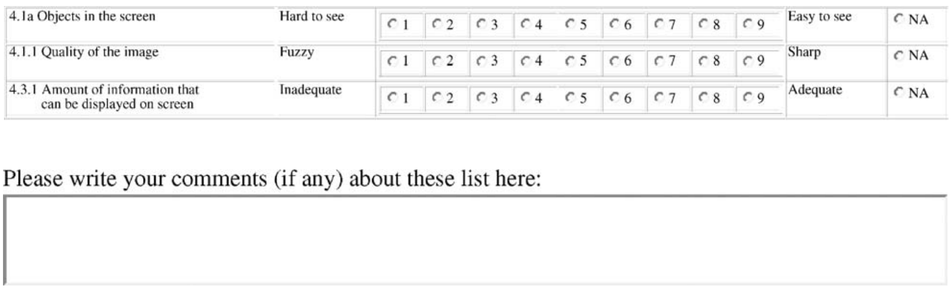
Overall User Reactions



Overall reactions to the system	Terrible	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Wonderful	○ NA
	Frustrating	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Satisfying	○ NA
	Dull	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Stimulating	○ NA
	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy	○ NA
	Inadequate power	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Adequate power	○ NA
	Rigid	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Flexible	○ NA

Please write your comments (if any) about these list here:

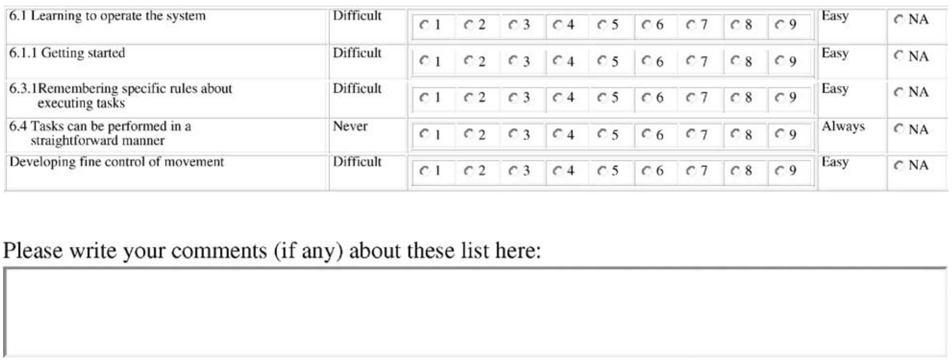
Screen



4.1a Objects in the screen	Hard to see	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy to see	○ NA
4.1.1 Quality of the image	Fuzzy	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Sharp	○ NA
4.3.1 Amount of information that can be displayed on screen	Inadequate	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Adequate	○ NA

Please write your comments (if any) about these list here:

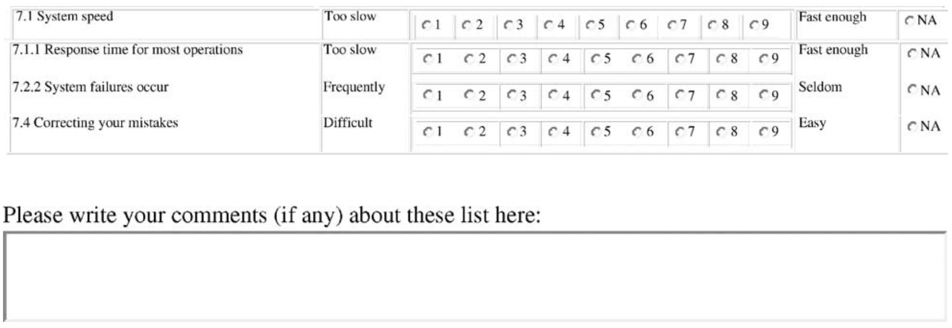
Learning



6.1 Learning to operate the system	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy	○ NA
6.1.1 Getting started	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy	○ NA
6.3.1Remembering specific rules about executing tasks	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy	○ NA
6.4 Tasks can be performed in a straightforward manner	Never	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Always	○ NA
Developing fine control of movement	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy	○ NA

Please write your comments (if any) about these list here:

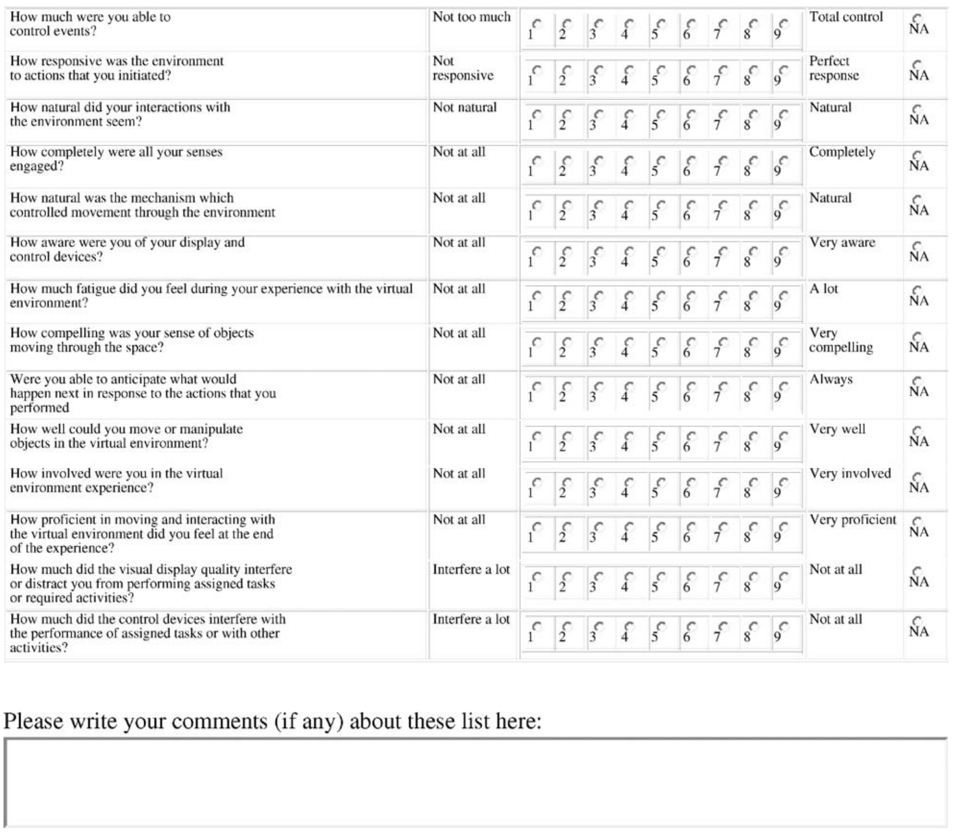
System Capabilities



7.1 System speed	Too slow	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Fast enough	○ NA
7.1.1 Response time for most operations	Too slow	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Fast enough	○ NA
7.2.2 System failures occur	Frequently	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Seldom	○ NA
7.4 Correcting your mistakes	Difficult	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Easy	○ NA

Please write your comments (if any) about these list here:

Presence Items



How much were you able to control events?	Not too much	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Total control	○ NA
How responsive was the environment to actions that you initiated?	Not responsive	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Perfect response	○ NA
How natural did your interactions with the environment seem?	Not natural	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Natural	○ NA
How completely were all your senses engaged?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Completely	○ NA
How natural was the mechanism which controlled movement through the environment	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Natural	○ NA
How aware were you of your display and control devices?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Very aware	○ NA
How much fatigue did you feel during your experience with the virtual environment?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	A lot	○ NA
How compelling was your sense of objects moving through the space?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Very compelling	○ NA
Were you able to anticipate what would happen next in response to the actions that you performed	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Always	○ NA
How well could you move or manipulate objects in the virtual environment?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Very well	○ NA
How involved were you in the virtual environment experience?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Very involved	○ NA
How proficient in moving and interacting with the virtual environment did you feel at the end of the experience?	Not at all	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Very proficient	○ NA
How much did the visual display quality interfere or distract you from performing assigned tasks or required activities?	Interfere a lot	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Not at all	○ NA
How much did the control devices interfere with the performance of assigned tasks or with other activities?	Interfere a lot	○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7	○ 8	○ 9	Not at all	○ NA

Please write your comments (if any) about these list here:

References

Alias Wavefront, 2003. Maya, http://www.aliaswavefront.com/en/products/maya/index.shtml.

Balakrishnan, R., Hinckley, K., 1999. The role of kinesthetic reference frames in two-handed input performance. In: Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology. ACM Press, New York, pp. 171–178.

Balakrishnan, R., Kurtenbach, G., 1999. Exploring bimanual camera control and object manipulation in 3d graphics interfaces. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press, New York, pp. 56–62.

Benford, S., Greenhalgh, C., Lloyd, D., 1997. Crowded collaborative virtual environments. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press, New York, pp. 59–66.

Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., Cruz-Neira, C., 2001. VR juggler: a virtual platform for virtual reality application development. In: Proceedings of IEEE Virtual Reality, pp. 89–96.

Blach, R., Landauer, J., Rosch, A., Simon, A., 1998. A highly flexible virtual reality system. Future Generation Computer Systems 14 (3–4), 167–178.

Blender.org, 2003. Blender, http://www.blender.org/.

Bowman, D.A., Hodges, L.F., 1997a. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In: Proceedings of the 1997 Symposium on Interactive 3D Graphics. ACM Press, New York, pp. 35–ff.

Bowman, D.A., Hodges, L.F., 1997b. Toolsets for the development of highly interactive and information-rich environments. The International Journal of Virtual Reality 3 (2), 12–20.

Bowman, D.A., Koller, D., Hodges, L.F., 1998. A methodology for the evaluation of travel techniques for immersive virtual environments. Virtual Reality, pp. 120–131.

Chin, J.P., Diehl, V.A., Norman, L.K., 1988. Development of an instrument measuring user satisfaction of the human-computer interface. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press, New York, pp. 213–218.

Craven, M., Taylor, I., Drozd, A., Purbrick, J., Greenhalgh, C., Benford, S., Fraser, M., Bowers, J., Jää-Aro, K.-M., Lintermann, B., Hoch, M., 2001. Exploiting interactivity, influence, space and time to explore nonlinear drama in virtual worlds. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press, New York, pp. 30–37.

Dachselt, R., Hinz, M., Meiner, K., 2002. Contigra: an xml-based architecture for component-oriented 3d applications. In: Proceeding of the Seventh International Conference on 3D Web Technology. ACM Press, New York, pp. 155–163.

Discreet 2003. 3D Max, http://www.discreet.com/products/3dsmax/.

Figueroa, P., Green, M., Hoover, H.J., 2002. InTml: a description language for VR applications. In: Web3D 2002 Symposium Proceedings, pp. 53–58.

Foskey, M., Otaduy, M., Lin, M., 2002. Artnova: touch-enabled 3d model design. In: Virtual Reality, 2002. Proceedings. IEEE, New York, pp. 119–126.

Gobbetti, E., Balaguer, J.-F., Thalmann, D., 1993. Vb2: an architecture for interaction in synthetic worlds. In: Proceedings of the Sixth Annual ACM Symposium on User Interface Software and Technology. ACM Press, New York, pp. 167–178.

Hinckley, K., Pausch, R., Goble, J.C., Kassell, N.F., 1994. Passive real-world interface props for neurosurgical visualization. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press, New York, pp. 452–458.

Hinckley, K., Tullio, J., Pausch, R., Proffitt, D., Kassell, N., 1997. Usability analysis of 3d rotation techniques. In: Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology. ACM Press, New York, pp. 1–10.

Howarth, P., Finch, M., 1999. The nauseogenicity of two methods of navigating within virtual interfaces. Applied Ergonimics 30, 39–45.

Jacob, R.J.K., Deligiannidis, L., Morrison, S., 1999. A software model and specification language for non-wimp user interfaces. Transactions on Computer Human Interaction 6 (l), l–46.

Kjeldskov, J., 2001. Combining interaction techniques and display types for virtual reality. In: Proceedings of OzCHI 2001, Edith Cowan University Press.

Mason, A.H., Walji, M.A., Lee, E.J., MacKenzie, C.L., 2001. Reaching movements to augmented and graphic objects in virtual environments. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press, New York, pp. 426–433.

Massink, M., Duke, D.J., Smith, S., 1999. Towards hybrid interface specification for virtual environments. In: Duke, D.J., Puerta, A., eds.), Design, Specification and Verification of Interactive Systems '99. Wein. Springer, Berlin, pp. 30–51.

Myers, B.A., Bhatnagar, R., Nichols, J., Peck, C.H., Kong, D., Miller, R., Long, A.C., 2002. Interacting at a distance: measuring the performance of laser pointers and other devices. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press, New York, pp. 33–40.

Pierce, J.S., Pausch, R., 2002. Comparing voodoo dolls and HOMER: exploring the importance of feedback in virtual environments. In: Proceedings of the CHI Conference. ACM, New York, pp. 105–112.

Poupyrev, I., Billinghurst, M., Weghorst, S., Ichikawa, T., 1996. The go-go interaction technique: non-linear mapping for direct manipulation in VR. In: Proceedings of the Ninth Annual ACM Symposium on User Interface Software and Technology. ACM Press, New York, pp. 79–80.

Poupyrev, I., Weghorst, S., Billinghurst, M., Ichikawa, T., 1997. A framework and testbed for studying manipulation techniques for immersive VR. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology. ACM Press, New York, pp. 21–28.

Poupyrev, I., Weghorst, S., Billinghurst, M., Ichikawa, T., 1998. Egocentric object manipulation in virtual environments: empirical evaluation of interaction techniques. In: Eurographics. Blackwell Publishers, Oxford.

Russell, M., Taylor, L., Hudson, T.C., Seeger, A., Weber, H., Juliano, J., Helser, A.T., 2001. VRPN: a device-independent, network-transparent VR peripheral system. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology. ACM Press, New York, pp. 55–61.

Sense8, 2000. Virtual reality development tools. The sense8 product line, http://www.sense8.com/products/index.html.

SGI, 2003. Iris performer home page, http://www.sgi.com/software/performer.

Sharlin, E., Figueroa, P., Green, M., Watson, B., 2000. A wireless, inexpensive optical tracker for the cave. In: Virtual Reality. IEEE, New York.

Shaw, C., Liang, J., Green, M., Sun, Y., 1992. The decoupled simulation model for virtual reality systems. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press, New York, pp. 321–328.

Shneiderman, B., 1998. Designing the User Interface: Strategies for Effective Human–Computer Interaction, Third ed. Addison-Wesley, Reading, MA.

Slater, M., Steed, A., McCarthy, J., Maringelli, F., 1998. The influence of body movement on subjective presence in virtual environments. Human Factors 40 (3), 469–478.

Smith, S.P., Duke, D.J., 2000. Binding virtual environments to toolkit capabilities. Computer Graphics Forum 19 (3).

Stanney, K.M., Mollaghasemi, M., Reeves, L., Breaux, R., Graeber, D.A., 2003. Usability engineering of virtual environments (VEs): identifying multiple criteria that drive effective VE system design. International Journal of Human Computer Studies 58 (4), 447–481.

Tan, D.S., Robertson, G.G., Czerwinski, M., 2001. Exploring 3D navigation: combining speed-coupled flying with orbiting. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press, New York, pp. 418–425.

Tanriverdi, V., Jacob, R.J.K., 2000. Interacting with eye movements in virtual environments. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press, New York, pp. 265–272.

Tanriverdi, V., Jacob, R.J.K., 2001. Vrid: a design model and methodology for developing virtual reality interfaces. In: ACM (Ed.), Proceedings of the ACM Symposium of Virtual Reality Software and Technology. ACM Press, New York, pp. 175–182.

Tramberend, H., 1999. Avocado: a distributed virtual reality framework. In: Virtual Reality, 1999. Proceedings. IEEE, New York, pp. 14–21.

University, C.M., Virginia, U., 1999. Alice: easy interactive 3D graphics, http://www.alice.org.

Usoh, M., Catena, E., Arman, S., Slater, M., 2000. Using presence questionnaires in reality. Presence: Teleoperators and Virtual Environments 9 (5), 497–503.

Viewpoint, 2003. Viewpoint Corporation, http://www.viewpoint.com.

VRCO, 2003. Cavelib library, http://www.vrco.com/products/cavelib/cavelib.html.

Ware, C., Balakrishnan, R., 1994. Reaching for objects in VR displays: lag and frame rate. ACM Transactions on Computer-Human Interaction (TOCHI) 1 (4), 331–356.

Ware, C., Rose, J., 1999. Rotating virtual objects with real handles. ACM Transactions on Computer-Human Interaction (TOCHI) 6 (2), 162–180.

Web3D Consortium, 2003. Extensible 3D (X3D^{TM}) Graphics. Home page, http://www.web3d.org/x3d.html.

Witmer, B., Singer, M.J., 1998. Measuring presence in virtual environments: A presence questionnaire. Presence: Teleoperators and Virtual Environments 7 (3), 225–241.

Zhai, S., Milgram, P., 1993. Human performance evaluation of manipulation schemes in virtual environments. In: Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS), pp. 155–161.