Pointing Facilitation Techniques for 3D Object Selection in Virtual Environments

Doctoral Thesis

Fernando Argelaguet Sanz

Barcelona, May 2011

Tesi doctoral presentada per obtenir el grau del Doctor
amb Menció Europea per la Universitat Politècnica de Catalunya

Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemes Informàtics

Informàtica Gràfica
Programa de Doctorat de Software

Advisor : Carlos Andújar Gran

Chapter 6

Interacting with 2D GUIs embedded in VEs

Application control is one of the fundamental tasks a Virtual Reality application must ensure. It refers to the user task of issuing commands, requesting the system to accomplish a particular function or changing its internal state [12]. Given its flexibility and its ease of use, WIMP user interfaces are the facto standard for PC desktop based interaction.

By providing mechanisms to efficiently deploy and interact with 2D graphical user interfaces in virtual environments, we can provide increased flexibility and functionality to VR applications. However, the usage of 2D graphical user interfaces in virtual environments present two main issues.

First, existing GUI toolkits for VEs are still too simple; they allow only for a limited number of GUI components and often lack visual authoring tools. In contrast, existing GUI toolkits for 2D desktop environments are mature, include powerful authoring tools, have a wide range of widgets and are actively maintained.

Second, efficient selection and manipulation of 2D GUI elements (widgets) are required. A common requirement for graphical user interfaces is that they have to cover a relatively small field of the user's viewport to limit the occluded content. As the GUI is placed in 3D space, it can be placed away from the user or downscaled. Nevertheless, in both cases, it will require the user to select and manipulate small widgets, potentially decreasing performance, increasing error rates and reducing comfort.

In this chapter, we address both issues by presenting a new approach for fast development of application-control graphical user interfaces in virtual environments and two orthogonal approaches to improve performance and comfort when interacting with complex GUIs.

6.1 A cost-effective approach for embedding 2D GUIs in virtual environments

Rather than providing a new API for defining and managing the user interface components, we aim at extending current 2D toolkits so that their full range of widgets can be displayed and manipulated either as 2D shapes on the desktop or as textured 3D objects within the virtual world. The proposed approach allows 3D GUI developers to take advantage of the increasing number of components, layout managers and graphical design tools provided by 2D GUI toolkits. Resulting programs can run on platforms ranging from fully immersive systems to generic desktop workstations with little or no modification.

The basic components of the system are depicted in Figure 6.1. The *host toolkit* is a 2D GUI extensible toolkit such as Qt [26], whose widgets are accommodated to VE applications by the *extended toolkit*. A key feature of our approach is that the additional features are provided by only subclassing the *host toolkit*. The first consequence is that existing applications with 2D GUIs already designed with the *host toolkit* can be adapted to a VE environment with minimum effort and all the application's source code for GUI creation and behavior remain unmodified.

We now introduce some notation that will be used through-out the rest of the section. The word widget is used to refer to user interface objects such as windows, buttons and sliders that are used as the basic constituents of GUIs. A widget receives mouse, keyboard and other events from the environment, and paints a representation of itself on the output device. Widgets are arranged into a hierarchical structure. A widget that is not embedded in a parent widget is called a top-level widget. Usually, top-level widgets are windows with decoration (a frame and a title bar). Non-top-level widgets are child widgets. We also

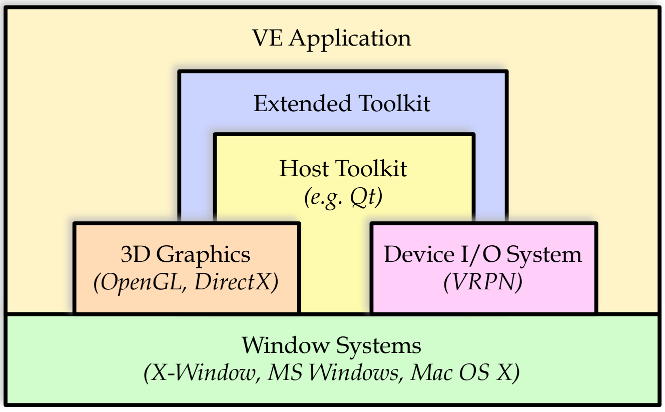


Figure 6.1: *System overview*

distinguish between widget objects provided by the host toolkit (called native widgets) and the objects that represent widgets as texture-mapped rectangles in 3D space (called virtual widgets).

Events of current 2D GUI toolkits can be roughly classified into two categories: application events, user events and synthetic events. *Application events* refer to GUI-related high-level events such as show, hide, close, resize and paint. On the other hand, *user events* are produced by simple user actions through input devices. According to the originating source, we also distinguish between *native events* originating from the window system or the host toolkit, and *synthetic events* initiated by the extension classes. For example, when the user presses a button on a 3D wand with the aim of selecting a menu item, a user event is created and then translated into a synthetic event (a mouse press event in this case) that is send to the native widget representing the menu item.

6.1.1 System components

The core of the extension toolkit consists on the components depicted in Figure 6.2. The main responsibilities of each component are described below.

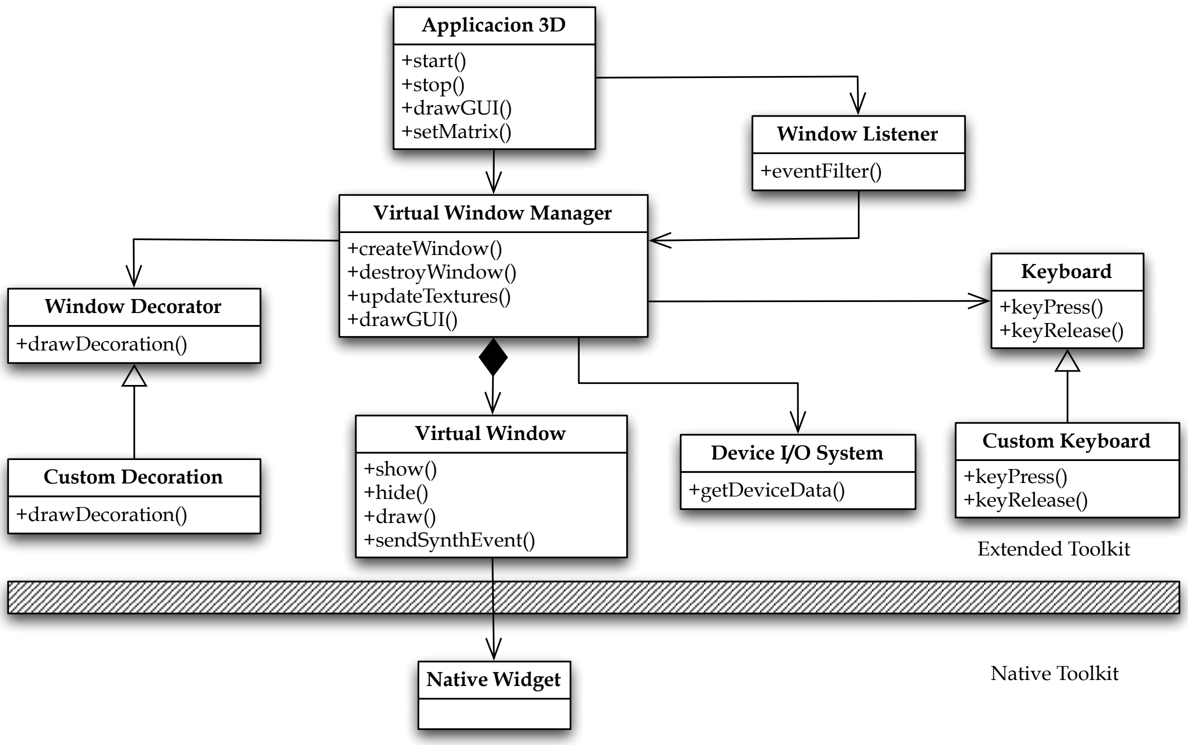


Figure 6.2: *UML Conceptual design of the extended toolkit. Only the most relevant components and operations are shown.*

The *Application 3D* provides a unified interface to the VE application, delegating client requests to appropriate subcomponents. This is the only component the application has to collaborate with, thus making the toolkit easier to use. The Application 3D manages the drawing of the GUI windows in 3D space as texture-mapped objects (forwarding the request to other components) and configuration options.

The *Virtual Window Manager* manages the behavior of virtual windows associated with top-level native widgets. This component handles basic window operations such as creation, destruction, show, hide and resize operations. This component also manages user events. This includes getting VR devices input data through the Device I/O System, managing interaction with the virtual window decoration through a decorator subclass, and translating user events into synthetic events that will be send to the appropriate virtual window.

Virtual Windows keep attributes concerning the 3D version of a top-level native widget such as rectangle size, texture size and transformation matrix. This component checks for intersection between the virtual window's plane and a selection primitive (e.g. a ray). In response to user events, Virtual Window objects send synthetic mouse events (to the child widget at a given 2D position) and keyboard events (to the child widget having the keyboard focus).

The *Window Listener* acts basically as an application-event filter. This component monitorizes the creation of new native widgets by the application and asks the Virtual Window Manager to create a new virtual window every time the application instantiates a top-level widget. Moreover, it also monitorizes application events related with a widget's life cycle: create, show, hide, paint, close and resize events originated from within the application. For each paint event (change on the image content) asks the virtual window manager to update the texture rectangle of the virtual window containing the native widget.

The rest of the components have simple responsibilities. The *Virtual Keyboard* defines an interface for a virtual keyboard that sends synthetic keyboard events to the application in response to user actions. The *Window Decorator* defines an interface for drawing the decoration of a virtual window. Decoration includes the frame, and buttons to iconify, deiconify and close the virtual window. Finally, the *Device I/O system* is used to read data from an extensible set of generic input devices.

Widget creation and placement

Each time a native window is created, the Virtual Window Manager receives a notification. If the widget is top-level, a virtual window is created along with a texture map capturing the contents of the widget's area. A rectangular portion of this texture is updated every time a widget of the native window changes its appearance.

An important issue is the initial window placement inside the 3D world. The placement strongly influences the user's ability and accuracy. A situation which arises in the use of multiple windows is the need to constantly arrange windows to get access to the particular window which houses the task or information needed at a given instant (window thrashing) [68]. According to the spatial reference, we can considerer windows that are world-referenced, object referenced, head-referenced and device-referenced [12]. Head-referenced and body referenced menus provide an appropriate spatial reference frame as they take the most of the user's proprioceptive sense, allowing users to accomplish some application control tasks without having to look at the menu. Since the extension toolkit knows everything about the widget hierarchy, any of these strategies can be plugged in.

Input handling

User events such as hand movements are captured by the Device I/O System and converted into synthetic events that are sent to native widgets. We will describe the main steps involved in this process with a concrete example. Suppose a CAVE user wearing a virtual wand (the wand has a six-DOF sensor, a two-DOF joystick and three buttons). Each time the user presses the left button a new user event is generated. This event contains a ray and a button state as parameters. This event is forwarded to the Extended toolkit which searches for the nearest virtual window containing the ray intersection. Finally, the intersected virtual window creates a synthetic event (a mouse event) and posts it to the native widget. The main advantage of this solution is that the management of the GUI elements and their behavior is delegated completely to the host toolkit, thus simplifying the migration of existing GUIs. Moreover, this approach allows the system to provide several interaction techniques for object selection such as ray-casting and arm-extension.

Host Toolkit Requirements

We now summarize the main features required for an extensible 2D GUI toolkit to be used as a host toolkit for the 3D extension:

- A hook to monitorize the creation and destruction of top-level widgets. In addition to this, the toolkit has to provide a mechanism to intercept widget-related events such as paint, show, hide, close and resize. For example, if a widget is repainted in response to a paint event, the application has to intercept the event to know that the associated texture requires an update. This feature is required by the Event Listener component.

- An operation to send a synthetic event (keyboard/mouse) to any widget (required by the Virtual Window component).
- A function to capture the image of a native widget into a bitmap. The image will be used to update the texture of the virtual window containing the widget (required by the Event Listener). If this function is supported also on widgets hidden by other windows, then the native GUI can share the desktop space with the OpenGL window where the VE application renders the virtual world.
- Functions to traverse the widget hierarchy (access to parent and children widgets) and operations to obtain the visible child widget at a given pixel position.

6.1.2 Prototype

A prototype system has been implemented above Qt, a cross-platform GUI development toolkit, and evaluated. Qt fulfills all the requirements listed in the previous section. Application events can be intercepted through adding a application event-filter through *QApplications::installEventFilter()*. The filter can either stop the event or forward it to other objects. Moreover, Qt allows multiple event filters to be installed, thus avoiding conflicts with application-defined event filters.

Creation of top-level widgets can be monitorized by listening to the *QEvent::Show* event which is send each time a window becomes visible. Changes in the widget's content are monitorized by listening to the *QEvent::Paint*. The QPixmap class provides two methods to grab the contents of a widget: *grabWindow()* grabs pixels directly off the screen, whereas *grabWidget()* asks the widget to paint itself by calling *paintEvent()* with output redirected to a bitmap. Although a bit slower, the later is more suitable because it works with hidden widgets.

Synthetic events (including predefined and user-defined events) can be send to any object through *QApplicationpost::Event()* and *QApplication::sendEvent()* methods. The former adds a synthetic event to the event queue with a given object as the receiver of the event; the later sends the event directly to the object. Finally, the toolkit provides operations for traversing the object hierarchy and for returning the visible child widget at a given pixel position in the widget's own coordinate system.

An example of collaboration is shown in Figure 6.3. (1) When a widget detects that it should repaint itself, it sends an event to the QApplication indicating which part of the widget should be repainted. (2) When appropriate, QApplication tells the widget it has to be painted. (3) This event is captured by the event filter and (4) produces a grab widget call to the QPixmap object, (5) which sends a repaint command to the QWidget. Finally, (6) the window listener asks the virtual window manager to update the texture.

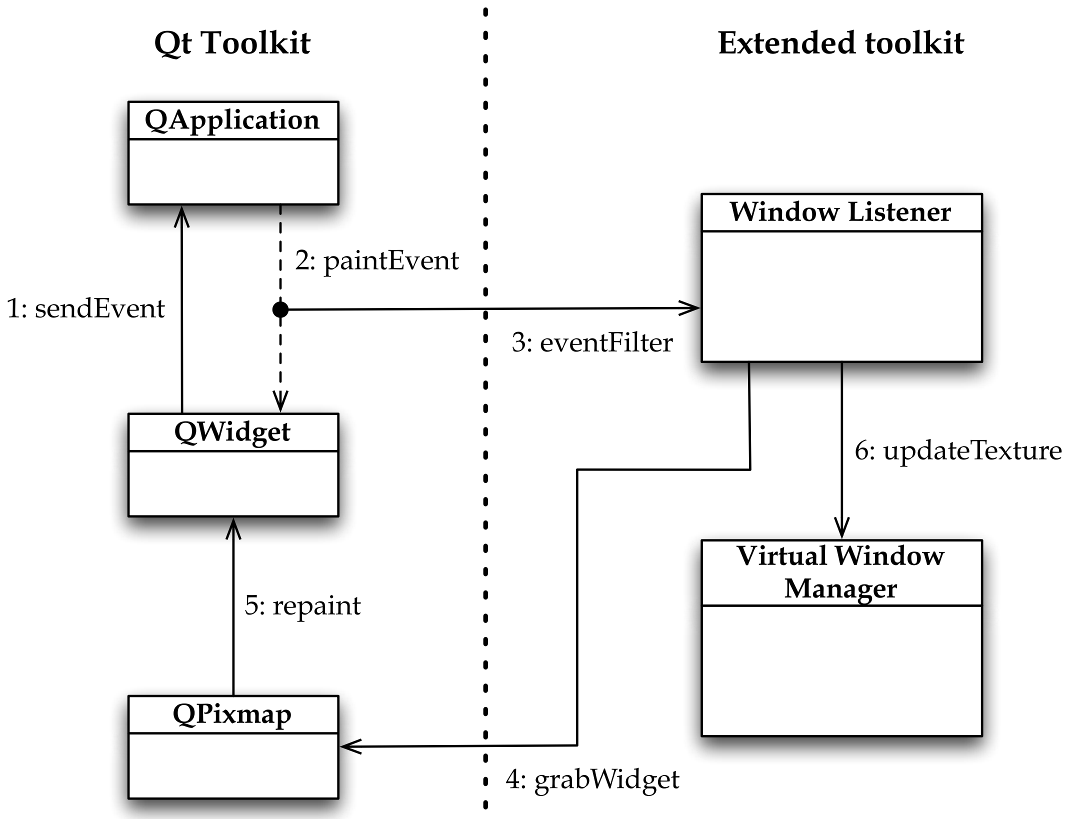


Figure 6.3: *Collaboration for processing a paint event between the Qt and our Extended toolkit.*

The prototype has been tested with two different applications: a scene viewer specialized for shipbuilding design (see Figure 6.4) and a volume rendering application (see Figure 6.5). Both applications had a Qt-based GUI and were extended to display stereoscopic images on either a CAVE or a stereo workbench. The source code modifications needed to accommodate the GUI to the CAVE where minimum. Moreover, it allowed the evaluation of the interaction techniques for selecting and manipulating 2D GUIs detailed in Sections 6.2 and 6.3.

The source code of the extended toolkit is available for download under the GNU GPL license at: http://www.lsi.upc.edu/~virtual/Qt3D.

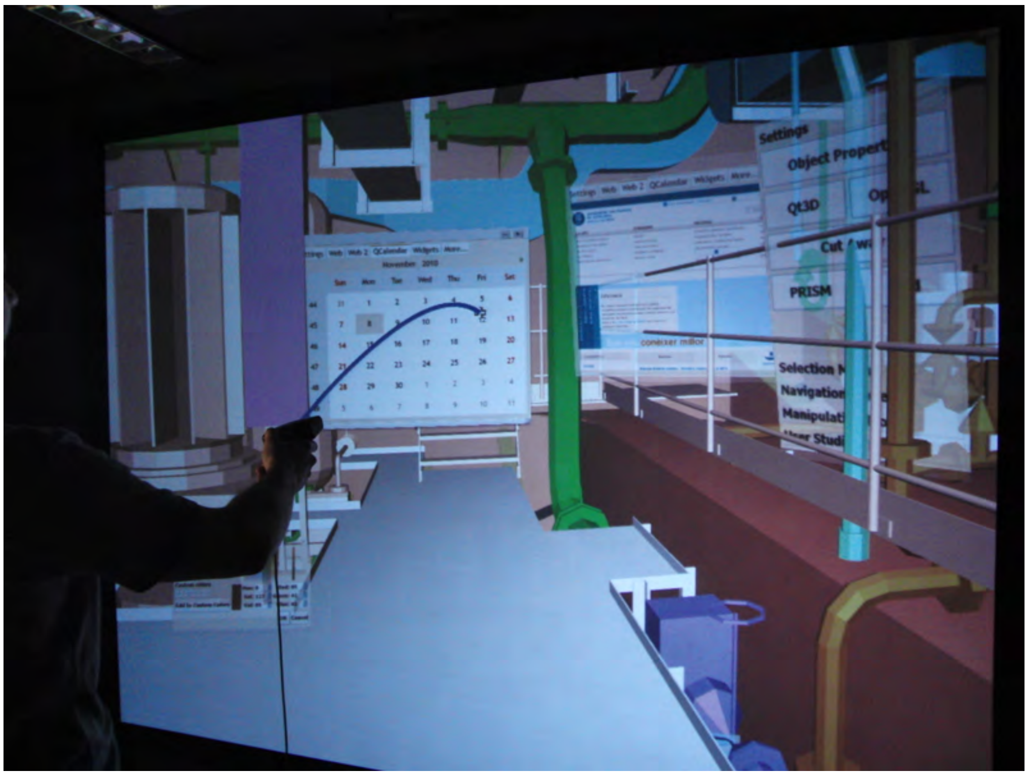


Figure 6.4: *Scene viewer specialized for shipbuilding using the developed GUI toolkit.*

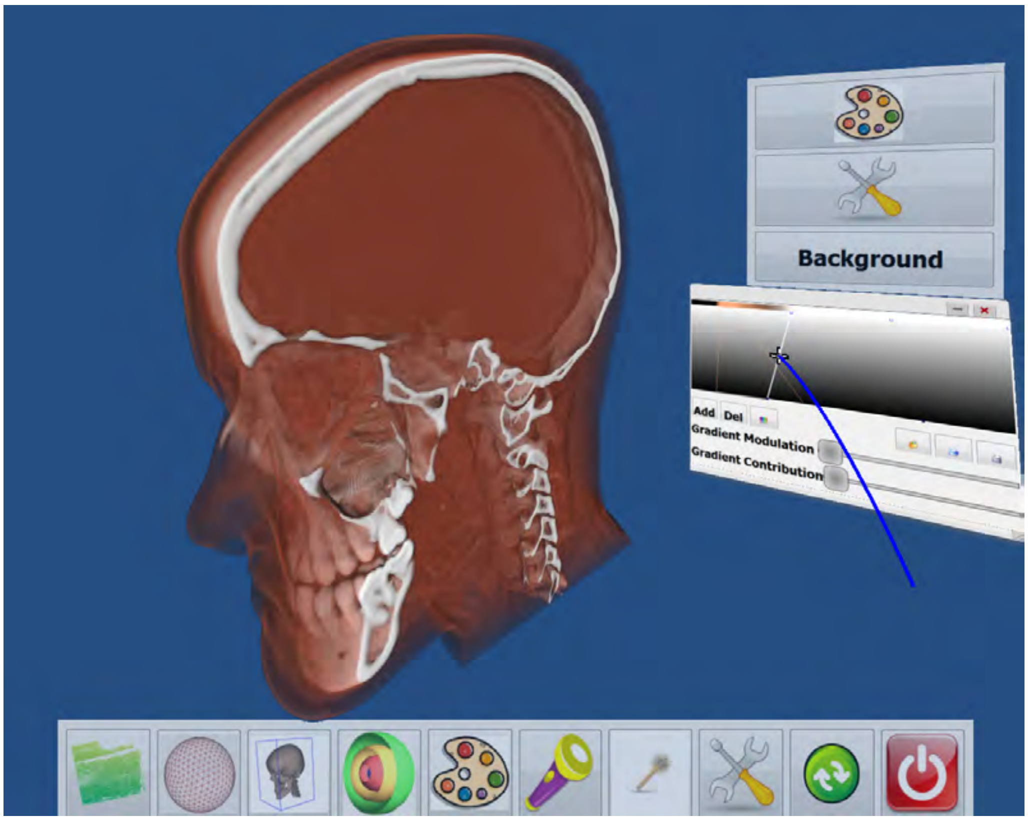


Figure 6.5: *Volume rendering application using the developed GUI toolkit.*

6.1.3 Discussion

Our approach has several advantages over previously-reported systems. Applications adopting our approach can provide an interface optimized for each platform (e.g. a desktop interface can be used on a desktop system, and an immersive interface can be used in an immersive system), without needing to modify the application. Moreover, importing 2D interfaces into the VE has some considerable advantages:

- Transparent use of all the functionalities provided by the host toolkit, enabling the use of a large number of widgets.
- Delegation of functional interface handling to an independent component.
- Users can work with the graphical user interface they are used to.
- Accommodation of different 3D selection techniques.

From the point of view of software development, the advantages of our approach are:

- An important part of the UI can be developed and tested in a desktop workstation.
- Access to UI graphical design tools (e.g. QtDesigner).
- Fast porting of existing applications to VEs with minimum changes.
- Transparent support to cluster-based systems and collaboration.

Our approach has some additional advantages over VNC-based methods for immersing 2D GUIs. The system is not framebuffer-oriented but widget-oriented. That implies that the VE application knows everything about the immersed GUI, not only framebuffer updates. That allows for much more flexibility for placing and sizing the widgets: host widgets can be automatically resized so that width and height are appropriate for fast texture conversion (power of two), widgets can be independently resized and moved from within the VE application, pop-up menus and pop-up lists can be true pop-up components, i.e. they can be drawn at a certain offset from the parent widget; the look-and-feel is completely customizable from within the VE application. Moreover, our system does not have any network nor image encoding overhead. For example, when running on a cluster, texture updates are not broadcasted to all clients because the native widgets and the virtual windows are local to each process.

6.2 Anisomorphic raycasting interaction

Two-dimensional windows in 3D environments can include small, nearby buttons which can be difficult to select and manipulate using standard 3D interaction techniques. For example, raycasting does not perform well when selecting small or distant objects. Small rotations of the wrist sweep out large arcs at the end of the selection ray. Therefore hand trembling and tracking errors are amplified with increasing distance, thus requiring a higher level of angular accuracy.

Accurate selection is also compromised by hand instability, amplified by the absence of constraints on the hand movements. Selections of small widgets require a considerable effort to stabilize the selection tool.

6.2.1 Friction Surfaces

We have conceived the Friction Surfaces metaphor to facilitate the interaction with external 2D applications being accessed from immersive VEs. The main goal is to provide accurate selection and manipulation of 2D GUIs that have not been particularly designed for VEs. To this end, we use a modified raycasting technique which adapts the CD ratio according to the size and position of the virtual window.

Friction surfaces uses two distinct modes: one which scales hand rotations when accuracy is needed (scaled mode) and one which provides direct, isomorphic interaction (normal mode). The scaled mode is activated automatically whenever the selection ray enters a virtual window. The mode is set back to normal mode when the selection ray leaves the active window.

We start by introducing some notation that will be used in the rest of the section. The *Device Coordinate System* (DCS) is an orthonormal frame centered at the position of the 6-DOF sensor attached to the user's hand. We assume the DCS is oriented as depicted in Figure 6.6a, with the negative Z axis defining the user's hand pointing direction. This pointing direction will be referred to as the *device ray*. The *Zero orientation* is an orthonormal reference basis is defined from DCS and the window's center when the scaled mode is activated and then remains unchanged until the mode is deactivated.

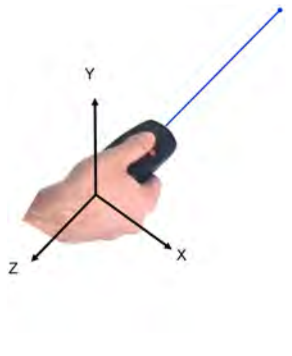
During the normal mode, the ray does not intersect any window, the CD ratio is always one; the device mapping is isomorphic. When the ray intersects a window, the mode switches to scaled mode and CD ratio is computed as follows. First, we compute the zero orientation reference frame $(\vec{x}, \vec{y}, \vec{z})$ given the virtual window and the DCS (see Figure 6.6b). Let $\vec{z}$ be a unit vector in the direction of the segment joining the window center and the device position at activation time. Let $\vec{u}$ the unit vector defined by the vertical orientation of the window. We compute $\vec{x} = \frac{\vec{u}\times\vec{z}}{\|\vec{u}\times\vec{z}\|}$ and $\vec{y} = \vec{z} \times \vec{x}$.

Then, we compute the range of directions the device ray can travel before leaving the active window. Let P_i be the i-th vertex of the virtual window. Let θ_i be the azimuthal angle (longitude) of P_i in the XZ-plane, measured from the negative Z-axis of the zero orientation, with $0 \leq \theta_i < 2\pi$. Likewise, let ϕ_i be the zenith angle (latitude) from the XZ-plane, with $-\frac{\pi}{2} \leq \phi_i \leq \frac{\pi}{2}$.

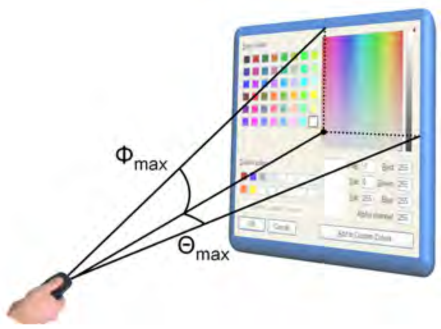
These spherical coordinates can be computed using Equations 6.1, where (x_i, y_i, z_i) are the P_i coordinates relative to the zero orientation and where the inverse tangent must be suitably defined to take the correct quadrant into account:

$$(\theta_i, \phi_i) = \left(\tan^{-1}\left(\frac{-x_i}{-z_i}\right), \sin^{-1}\left(\frac{y_i}{\sqrt{x_i^2 + y_i^2 + z_i^2}}\right)\right) \tag{6.1}$$

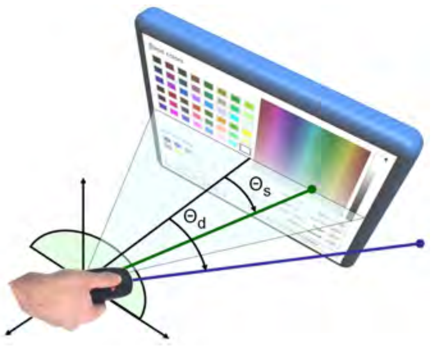
Once computed the angles for the four window's corners, we obtain the maximum rotational angles in both directions as $\theta_{max} = \max_i \left\{|\theta_i|\right\}$ and $\phi_{max} = \max_i \left\{|\phi_i|\right\}$. Since we want to use isotropic scale on both directions, we just use the maximum of both. Therefore, the CD ratio r is computed as shown in Equation 6.2, where ψ is a user-defined constant. ψ determines the range of directions of the input device that approximately map onto a selection ray within the virtual window. If the device ray exceeds that angle the selection ray will fall outside the active window. In our implementation we employed $\psi = \pi/4$, thus providing the user with a 90 degrees arc for interaction with the active virtual window. If the window is sufficiently close to the user ($r < 1$), we do not enable the scale mode and keep the CD ratio



(a)



(b)



(c)

Figure 6.6: *Elements involved in the computation of the CD ratio during the scaled mode: (a) Device coordinate system and device ray, (b) spherical coordinates used at activation time to fix the CD ratio , (c) spherical coordinates used for computing the selection ray direction during scaled mode.*

equal to one.

$$r = \max\left(1, \frac{\psi}{\max(\theta_{max}, \phi_{max})}\right) \tag{6.2}$$

As the deactivation causes the selection ray to coincide again with the device ray, the selection ray might enter another window. We found this situation to be rare in practice as most windows are likely to be placed in front of the user. Nevertheless, unintentional activation can be easily avoided by waiting a short period of time (e.g. half a second) before the scaled mode is activated over the new window.

Computation of the selection ray

In scaled mode, the selection ray is computed using the CD ratio defined at activation. Let θ_d and ϕ_d be the spherical coordinates of an arbitrary point of the device ray (distinct from its origin) with respect to the zero orientation basis. We first compute the spherical coordinates θ_s and ϕ_s of a target point T by applying the CD ratio, $\theta_s = \theta_d/r$, $\phi_s = \phi_d/r$ (see Figure 6.6c).

The new coordinates of T require a simple conversion back to cartesian coordinates (see Equation 6.3, $\rho > 0$ is an arbitrary value). The resulting selection ray passes through the current device position and point T.

$$\begin{aligned} x &= -\rho \sin(\theta_s)\cos(\phi_s) \\ y &= \rho \sin(\phi_s) \\ z &= -\rho \cos(\theta_s)\cos(\phi_s) \end{aligned} \tag{6.3}$$

Feedback

Two distinct options were considered to provide visual feedback. The first option consisted in drawing both the device ray and the selection ray, using different visual attributes (such as color and thickness). However, this option appears to be quite distracting so we have opted for a single bent ray providing feedback of both the device ray and the selection ray.

Similar to IntentSelect [32] and the Flexible Pointer [90], we draw a curved line segment using a Bézier spline (see Figure 6.7). The curve originates at the user's hand and ends at the intersection P of the selection ray with the virtual window's plane. These two points define the first and last control points of the Bézier curve. The second control point is computed on the device ray so that the tangent direction at the origin is that of the device ray. Finally, the third control point is the point on the device ray closest to P.

When the selection ray leaves the virtual window, the scaled mode is deactivated and the displayed ray instantly goes straight. The users' feeling when scaled mode is active is that a flexible ray gets curved as it is moved over a virtual high friction surface defined by the window. Visual feedback is completed by drawing a cross-shaped cursor in the intersection point P.

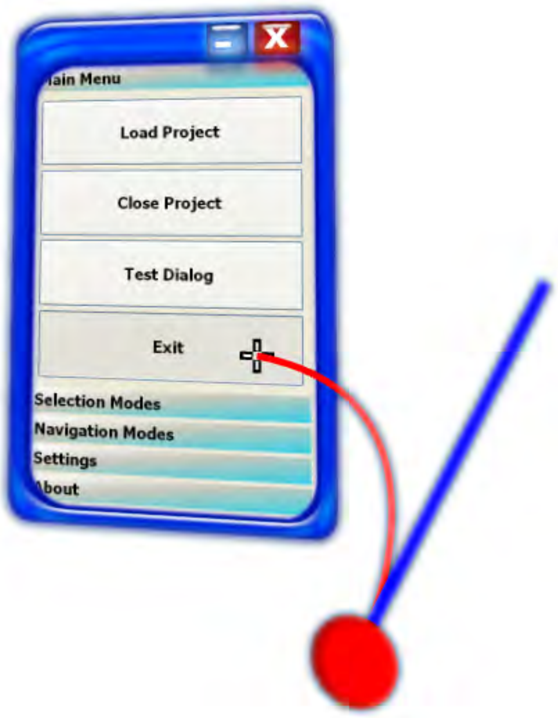


Figure 6.7: *The red ray corresponds to the feedback ray and the blue ray to the real device ray. The selection ray is not displayed but its defined by the hand's position and the intersection point between the feedback ray and the virtual window.*

6.2.2 Friction Surfaces evaluation

We conducted a usability evaluation to measure the effectiveness of the anisomorphic raycasting manipulation compared with classic isomorphic raycasting and raycasting in combination with PRISM [41]. The evaluation test was designed to evaluate the task performance in terms of time-to-complete a given task and the maximum accuracy achieved in a fixed period of time.

Design and procedure

The test dialogs used in the experiments are shown in Figure 6.8. The first two dialogs are designed to measure task performance when selecting small/middle size objects. The first dialog contains different kinds of buttons whereas the second dialog includes basically combo boxes and selection lists. The third dialog is also designed to measure speed but putting the emphasis on manipulation rather than on selection. Finally, the fourth dialog is designed to measure the accuracy during object manipulation. In all cases the label attached to each widget indicates the requested task.

A repeated-measures, within-subject design was used. The dependent variable was the selection technique: raycasting (RC), friction surfaces (FS) and PRISM. Users performed the task once for each technique condition. The order of the conditions was randomized for each user to avoid ordering effects.

The dependent variables were selection time, error rates, the path length described by the cursor over the virtual window, and the amount of directional changes (changes of direction of the hand orientation greater than 90 degrees).

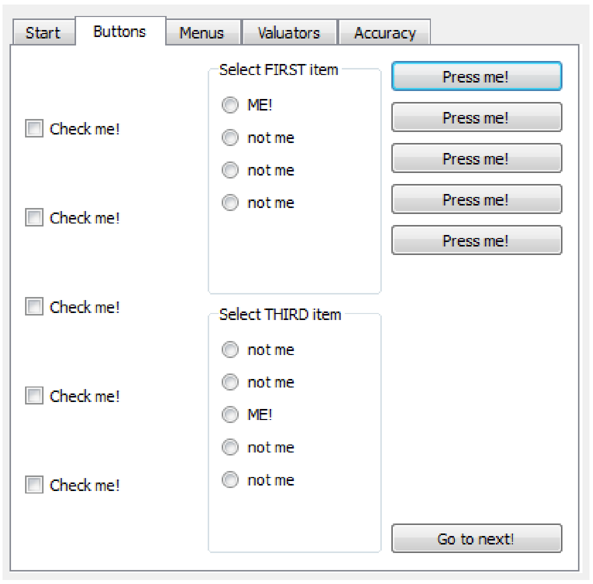
For the first three dialogs, users were requested to complete the involved tasks as quickly as possible. For the fourth dialog, users were asked to manipulate several sliders to get a certain value as accurately as possible, but giving only five seconds of time for each slider, starting from the first click on it. After that time, the slider was disabled and the user was forced to proceed with the next slider.

Apparatus and virtual setup

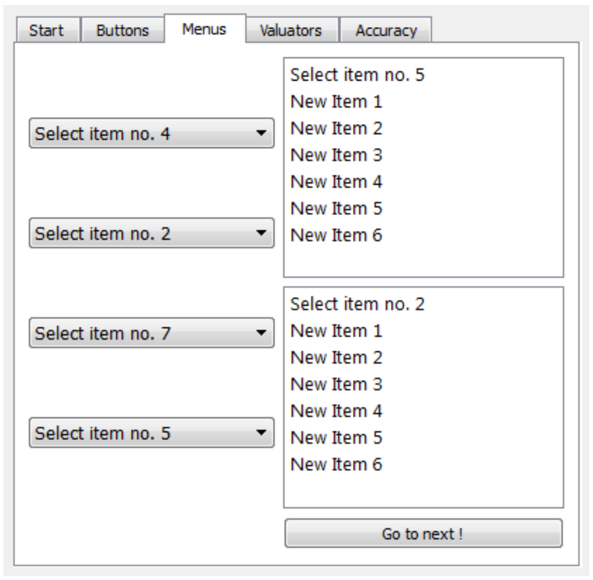
All the experiments were conducted on a four-sided CAVE with a 6-DOF hand-held device and a Polhemus Fastrak tracking system with 2 receivers providing 60 updates/s with 4 ms latency. The virtual window used in the experiments was initially placed at 1.5 m from the CAVE center, covering about 20 degrees of the user's field-of-view.

Participants

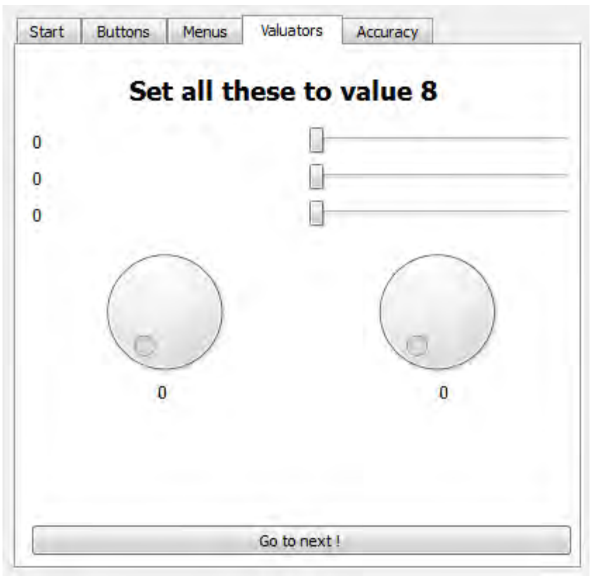
Seventeen users (undergraduate and graduate students) participated in the study, aged 22-42, 14 male and 3 female. Most participants (9) had no experience with VE applications; 5 had some experience and 3 were experienced users.



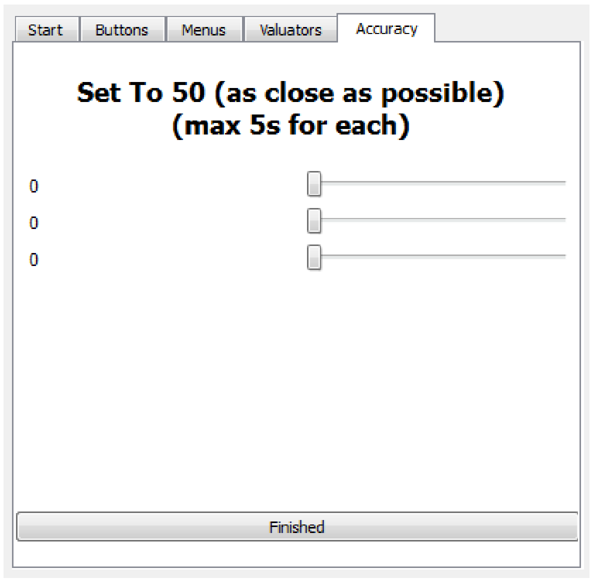
(a)



(b)



(c)



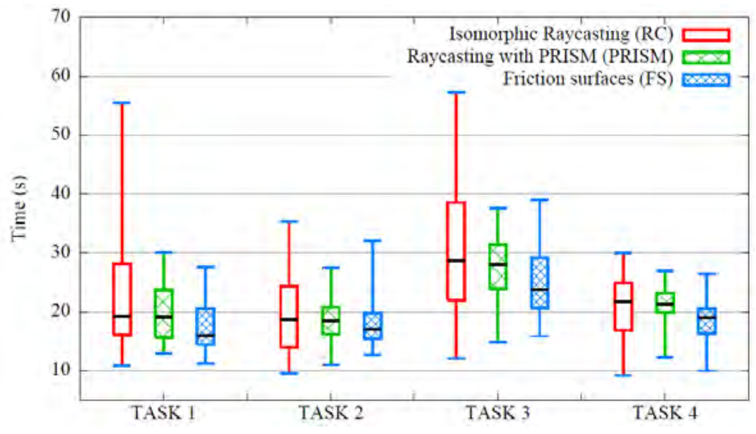
(d)

Figure 6.8: *The test dialogs used in the user evaluation.*

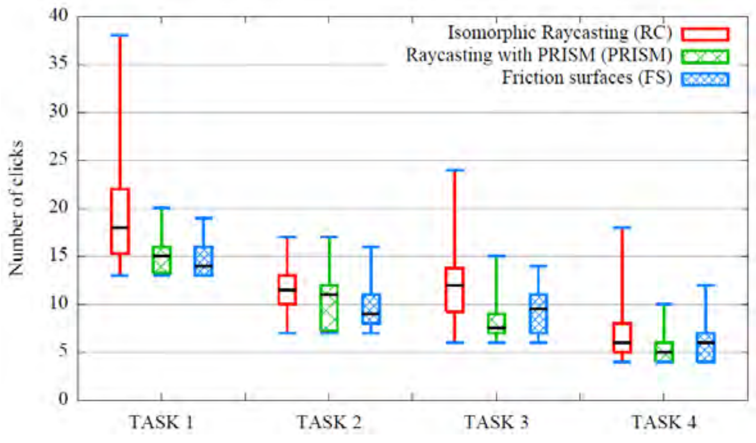
Results

Figure 6.9a shows the completion time for each task. Tasks 1-4 correspond to the dialogs (a)-(d) shown in Figure 6.8. The one-way ANOVA of completion time versus interaction technique showed significant differences for task 1 ($p < 0.05; F = 4.15$) and for task 3 ($p < 0.05; F = 3.7$). Tukey pair-wise HSD tests showed only significant differences between RC and FS, ($p < 0.01$) for both tasks, users with FS performed significantly faster than users with RC.

Figure 6.9b shows the button clicks for each task. Note that in tasks emphasizing selection (tasks 1 and 2), users made less mistakes on average with FS, whereas PRISM yield better results in tasks emphasizing on manipulation (tasks 3 and 4). The one-way ANOVA showed significant differences in clicks for tasks 1 ($p < 0.001; F = 12.14$), task 3 ($p < 0.001; F = 13.62$) and task 4 ($p < 0.001; F = 5.53$). Tukey pair-wise HSD tests showed that PRISM



(a)



(b)

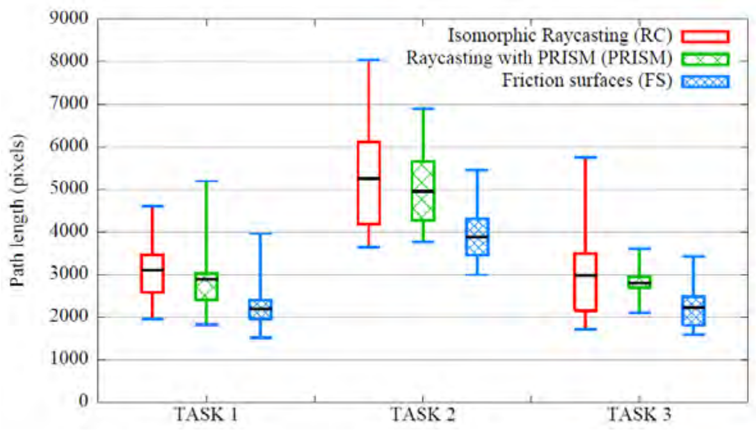
Figure 6.9: *(a) Time to complete the tasks involved for each dialog. (b) Number of button clicks performed during each task.*

was always significantly better than RC ($p < 0.01$), and FS was better than RC in tasks 1 and 3 ($p < 0.001$). Regarding the number of mistakes, it should be noted that PRISM incorporates a noise filter. Any motion below a given velocity is considered tracking error or inadvertent drift and the controlled ray is not moved. Note that our current implementation of Friction Surfaces does not have such a filter.

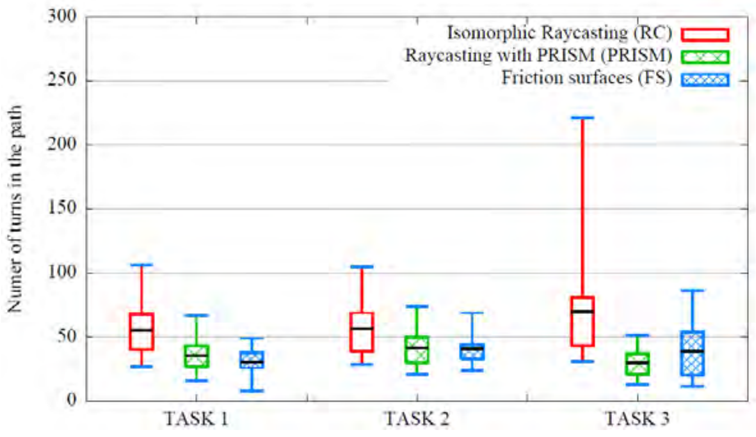
The length of the paths traced by the cursor, in native window pixel units, are shown in Figure 6.10a. The one-way ANOVA showed significant differences ($p < 0.001; F = 28.4582$). Tukey pair-wise tests confirmed that FS lead to cursor paths significantly shorter than both RC ($p < 0.01$) and PRISM ($p < 0.01$). Figure 6.10b also shows the number of times the user had to rectify their movement. Both PRISM and FS were found to produce less turns than RC ($p < 0.01$).

Figure 6.10c shows the results of the accuracy test (Task 4). The plot shows the average deviation (in slider units) from the target value when the user had only five seconds to adjust it (see Figure 6.8d). Each slider had increasing ranges and thus increasing levels of difficulty. Both PRISM and FS performed much better than RC, with nonsignificant differences between PRISM and FS.

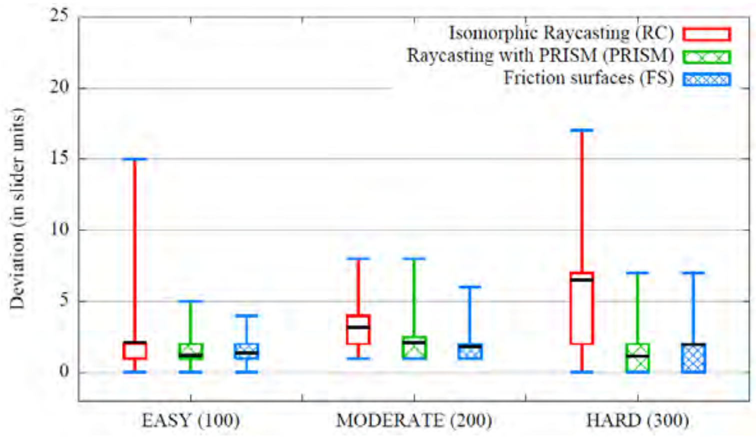
Regarding the path traced by the 3D cursor over the virtual window, Figure 6.11 shows the paths described by users who achieved times on the first, second and third quartile values in Task 1. Color temperature represents the speed of the trace. Note that the lack of accuracy of isomorphic ray-casting forced the users to perform many attempts before the right button was selected. This is reflected by the loops around the small targets in Figure 6.11a. This contrasts with the smoother paths produced by PRISM and FS (Figure 6.11b and 6.11c).



(a)

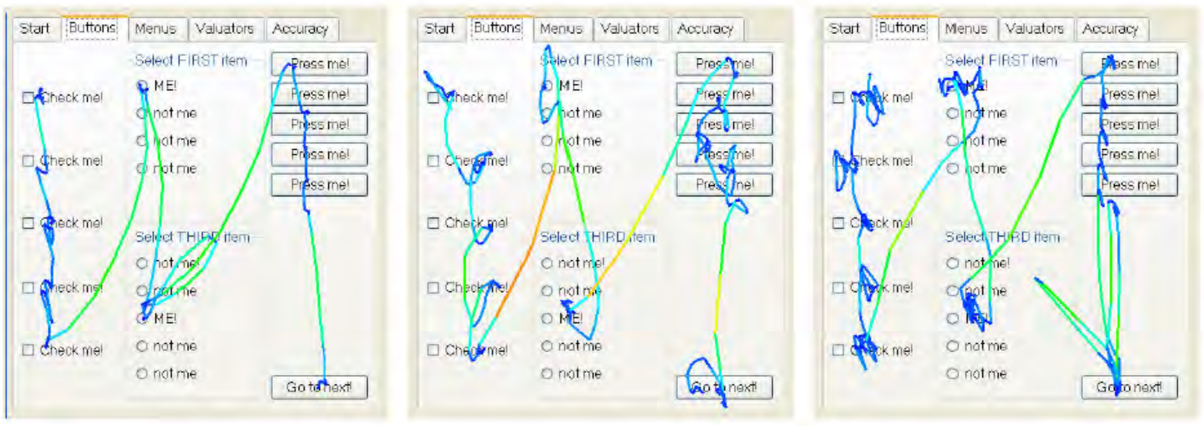


(b)

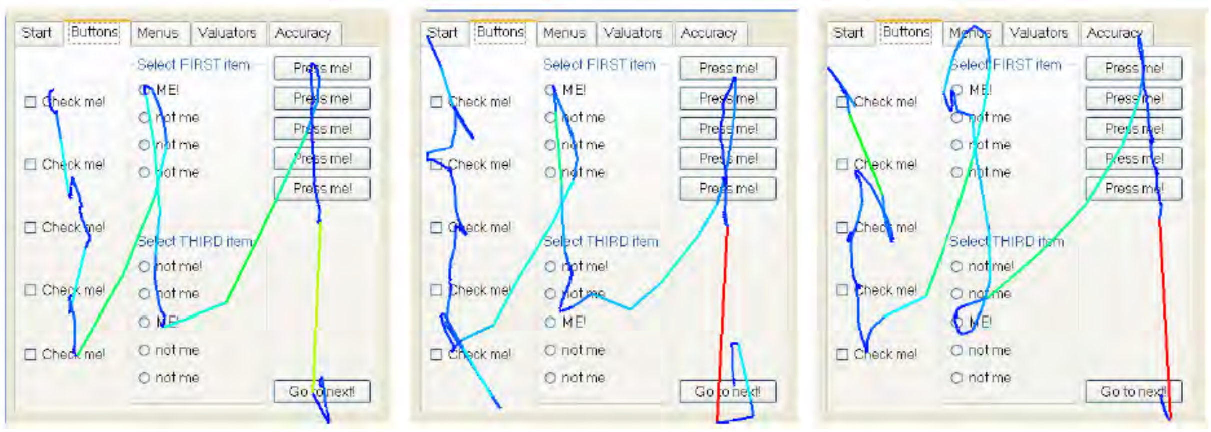


(c)

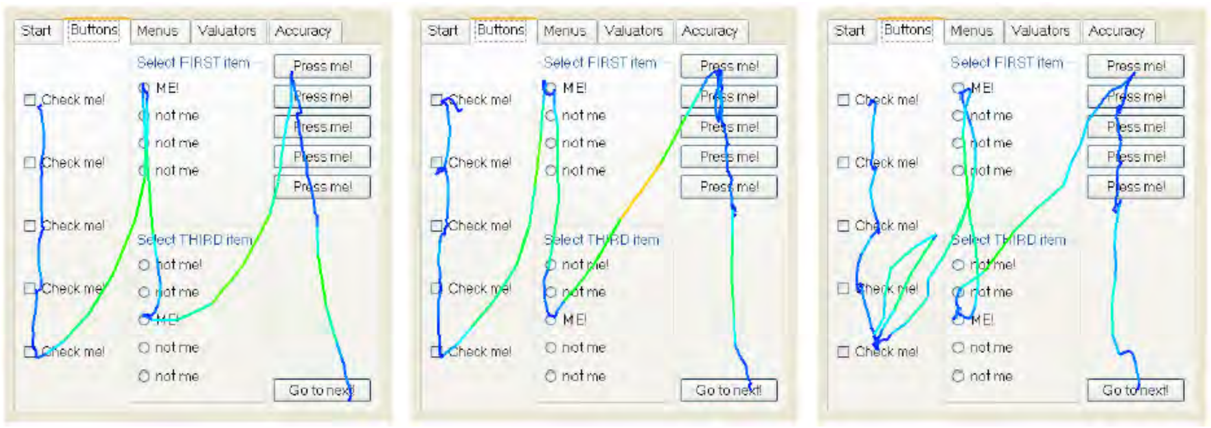
Figure 6.10: *(a) Length of the path traced by the cursor. (b) Number of turns in the path traced by the cursor. (c) Deviation from the target value on the accuracy test. The integer value between parentheses is the slider's range.*



(a)



(b)



(c)

Figure 6.11: *Path traced by the 3D cursor over the virtual window with RC (top), PRISM (middle) and FS (bottom). Paths correspond to users who achieved times on the first, second and third quartile values. Note that checkboxes can be toggled by clicking on their label.*

Survey

After the experiments, subjects were requested to rate each interaction technique using a 7-point Likert scale. All users preferred either PRISM or FS against RC, with nonsignificant differences between PRISM and FS. Nine users preferred PRISM with average rate of 5.5; eight users preferred FS with average rate of 5.4. RC was given an average rate of 3.

We also asked subjects about what they found most difficult and what was the easiest for them. The results were similar from all users. All of them agreed on having less problems on selecting buttons and manipulating sliders with friction surfaces.

The most difficult task was to achieve a certain value on the sliders, because the free movement of the wand on their hand makes it difficult to maintain the value when the finger is moved to press or release the button. This problem was noticeably alleviated with PRISM and with our technique. Most users complained about the effort required for selecting small buttons with normal ray-casting because of the considerable effort to stabilize the wrist.

On the other hand, a few users pointed out that friction surfaces was a bit unnatural compared with isomorphic raycasting, although their performance was better using an anisomorphic mapping.

A limitation of our technique is that, for certain orientations, the curvature of the selection ray and its intersection with the virtual window is hard to perceive (when the viewpoint approaches the plane defined by the four control points of the curved ray). However, users did not find this to be a problem as the cross-shaped cursor showing the intersection of the selection ray with the virtual window was clearly visible.

Discussion

Using an anisomorphic mapping between the user's hand orientation and the selection ray orientation, we are able to scale down hand rotations enabling accurate selection and manipulation of small GUI objects.

The user evaluation showed that PRISM and Friction Surfaces perform significantly better than classic ray-casting, with little performance differences between FS and PRISM. PRISM seems to perform better than Friction Surfaces when extreme accuracy is required (e.g. adjusting a slider with pixel accuracy) whereas Friction Surfaces is particularly suitable for fast selection of small targets. Both cannot be achieved with classical raycasting as they require the user a great effort to stabilize the pointing device before each selection, the

Heisenberg effect will further hinder precise selections.

A key feature of Friction Surfaces is that maintains both directional and nulling compliances [99] as it simulates approximately the interaction with a large spherical window (see Figure 6.12). Note that PRISM does not preserve nulling compliance and requires offset recovery techniques to reduce the accumulation of an offset value representing the angular difference between the hand and the ray being manipulated. Besides these aspects, an important difference is that Friction Surfaces does not force users to slow down their movements to gain precision. Our approach uses a larger range of movements for controlling the ray in a more reduced region.

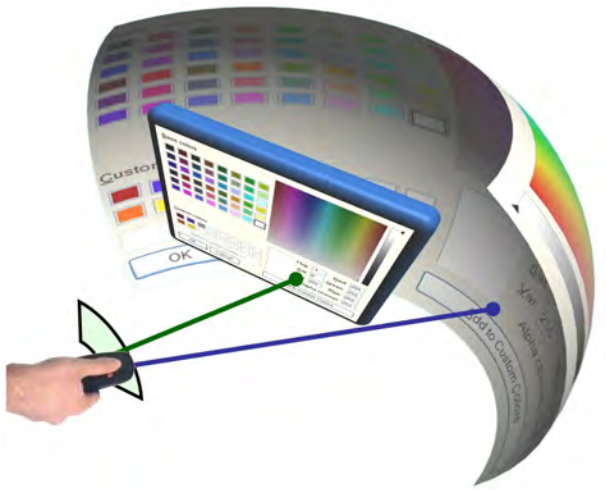


Figure 6.12: *Manipulation with friction surfaces simulates approximately the interaction with a large spherical window. Note that rendering this simulated window would be unusable as it would occlude most of the scene.*

6.3 Decoupling motor space and visual space

In a typical desktop HCI setup, the motor space is decoupled from the display space (e.g. the movement of a mouse on a horizontal plane is transformed to the movement of a cursor on a vertical screen). We propose a *Virtual Pad* metaphor exploiting how a similar decouple is beneficial for the interaction with 2D virtual windows embedded in a VE. The decouple is accomplished through a virtual pad which receives user actions (motor space) and maps them into cursor movements on the active virtual window (visual space).

By decoupling the motor and the visual space, virtual windows can be manipulated within a user-defined working volume (the virtual pad), whose location and size is completely independent from the application's visual representation. In addition, the user is able to adjust the control space (virtual pad dimensions) allowing to seamlessly balance speed and accuracy without affecting the visual representation of the application's GUI.

The Virtual Pad metaphor has been conceived to facilitate the interaction with external 2D applications being accessed from immersive VEs that have not been particularly designed for VEs while maximizing user's comfort.

6.3.1 The Virtual Pad

The virtual pad (see Figure 6.13) is a rectangular region that defines the control space, i.e., the region the user has to touch or point to when interacting with a virtual window. This tool establishes the link between the working space and the visual space.

In our implementation the virtual pad is rendered as a wireframe rectangle providing the user with a reference frame with minimal visual obtrusion, the user is supposed to be looking at the virtual pad only from time to time or through peripheral vision.

The virtual pad is user-adjustable, it allows the user to change its size and location. Allowing to seamlessly balance speed and accuracy without affecting the visual representation of the application's GUI, through increasing or decreasing its area.

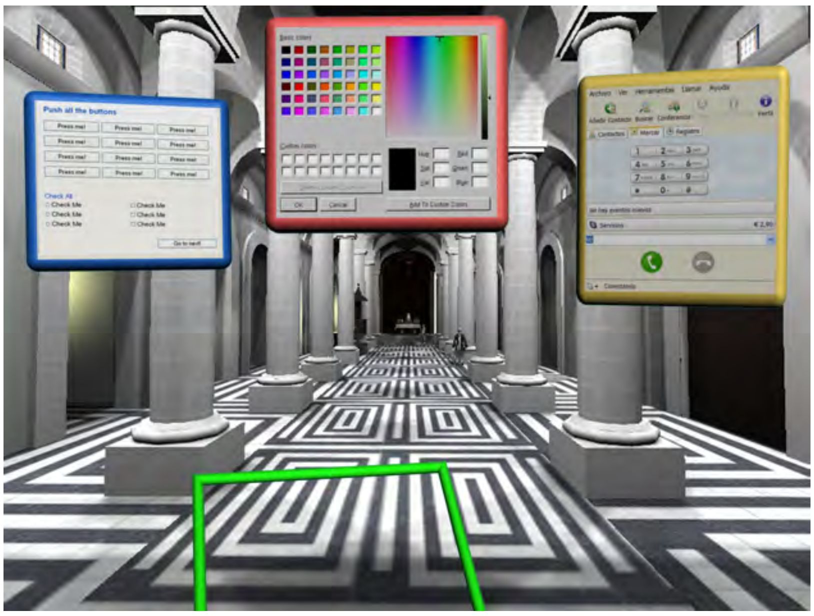


Figure 6.13: *Virtual windows on VE. The green rectangle represents the frame of the virtual pad.*

Motor-to-visual space mapping

When using our virtual pad metaphor in combination with pointing techniques like ray-casting, the mapping from control space to visual space is straightforward:

- Compute the intersection of the ray/selection volume with the virtual pad's plane.
- Compute the new cursor location in the virtual window by transforming linearly the pad's coordinates into window coordinates.

In contrast, when using our virtual pad metaphor in combination with the virtual hand technique, a projection method must be adopted. Due to the absence of constraints on the hand movements, the user hand will move close to but not exactly over the virtual pad. Therefore, the hand position must be projected into the virtual pad before mapping its position to a new cursor position. Several projections can be adopted. Note that the projection type actually defines the shape of the working space. We have considered two different projections:

- *Parallel projection*: the normal vector of the virtual pad is used to project the hand position into the virtual pad. The resulting working volume is a rectangular prism.
- *Viewpoint projection*: the hand position is projected to the virtual pad from the user's viewpoint. Note that this is equivalent to using the ray-casting variant where the ray's direction is defined by the segment joining the user's viewpoint with the hand position. The resulting working volume is a pyramidal frustum.

Activation/deactivation

The proposed metaphor has been conceived to manipulate 2D GUIs (or in general, constraint 2D manipulation) inside 3D worlds. Therefore, in a practical situation, this approach has to co-exist with the selection and manipulation techniques used for interacting with the rest of 3D objects. The activation of the virtual pad should therefore be in accordance with these techniques. In our implementation, raycasting is used to both select 3D objects and activate the external application the user wants to interact with.

Activation of a virtual window causes the virtual pad to be adjusted so that its aspect ratio matches the aspect ratio of the virtual window, as described above. Once the window becomes active all the actions over the virtual pad are mapped to it.

Several activation strategies can be adopted. If we want the user not to leave the virtual pad every time he wants to interact with another window, we need to provide mechanisms to choose the active window automatically. For selecting the active window we can use hints given by window events and changes in the window hierarchy. In our implementation a window becomes active automatically when (a) it is shown as the result of a user action (e.g. the user clicks a menu to open a file dialog) or (b) its child window is closed or hidden (e.g. after closing a modal dialog, the focus is assigned to the window where the action was initiated).

This behavior is particularly convenient with pop-up menus and pop-up windows. Manual activation of any window is also provided by simply clicking over the chosen window, although this forces the user to move the ray outside the virtual pad.

As discussed above, when there is an active window all the actions over the virtual pad are mapped to it. When no window is active, the virtual pad has to provide convenient feedback of this fact. We have decided to let the virtual pad to behave as a virtual desktop when no window is active (see Figure 6.14b). In this case the pad shows an icon for every application window (including maximized, minimized and hidden windows). We also use a button trigger to manually deactivate the active window and enable the virtual desktop behavior of the pad. This allows the user to operate different windows with minimum effort and without leaving the virtual pad.

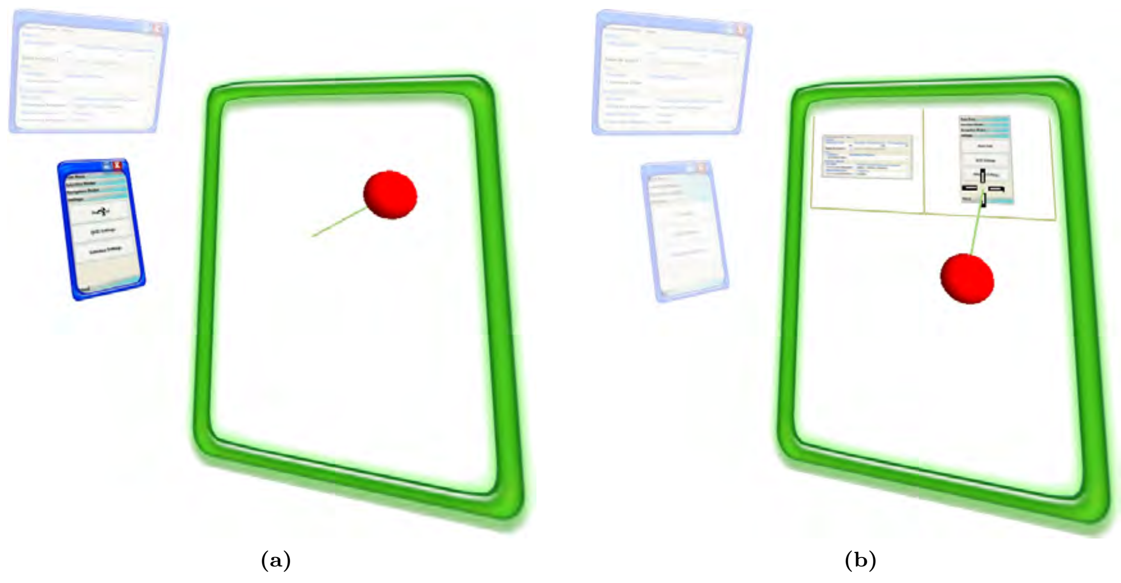

(a) (b)

Figure 6.14: *(a) Interaction using the Virtual Pad. (b) If no window is selected, the Virtual Pad works as a window selector.*

6.3.2 Virtual Pad evaluation

Since the virtual pad can be freely adjusted to suit user preferences, we can expect a higher degree of comfort on positions reducing fatigue in wrist, arm, shoulder and neck, plus the additional benefit of dynamic adjustment of the speed/accuracy tradeoff.

Therefore we have designed an experiment to measure the price we have to pay in user performance to achieve this higher degree of comfort and flexibility. More precisely, we want to measure the impact in performance of the decoupling between control and visual spaces, choosing ray-casting as the interaction technique and 2D GUI manipulation as the reference task.

When motor and visual spaces are not superimposed, target acquisition requires a different cognitive strategy to use vision to control the cursor rather than a more direct visuomotor mechanism [107]. Although this decoupling is ubiquitous in desktop HCI setups, very few studies have analyzed its impact in user performance.

Design and procedure

A repeated-measures, within-subjects design was used. The dependent variable was the interaction technique used, (a) Direct Manipulation of the virtual window (Direct, see Figure 6.15a), (b) interaction through a Virtual Pad (VPad, see Figure 6.15b) with exactly the same size of the virtual window but located in a more comfortable position and (c) interaction through a 50% enlarged pad (VPad+) in a coplanar position with the VPad condition (see Figure 6.15c). The three conditions were randomized for each user. Before each ex-

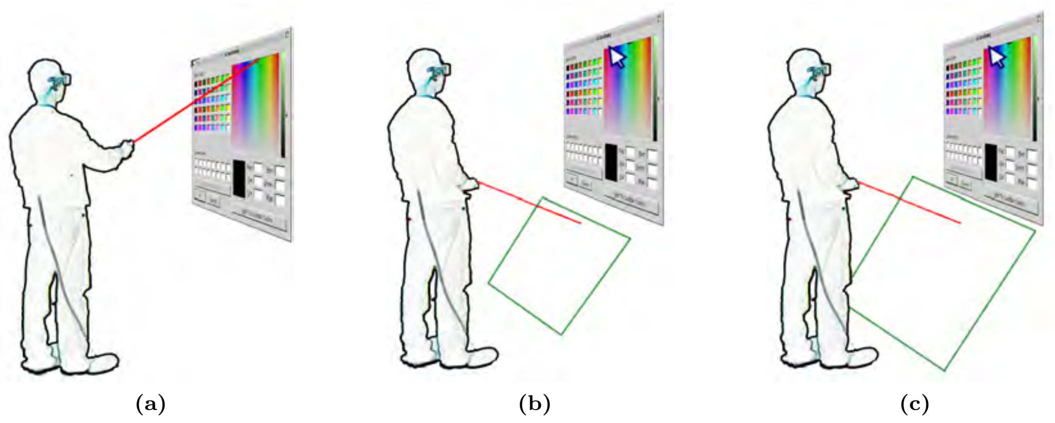
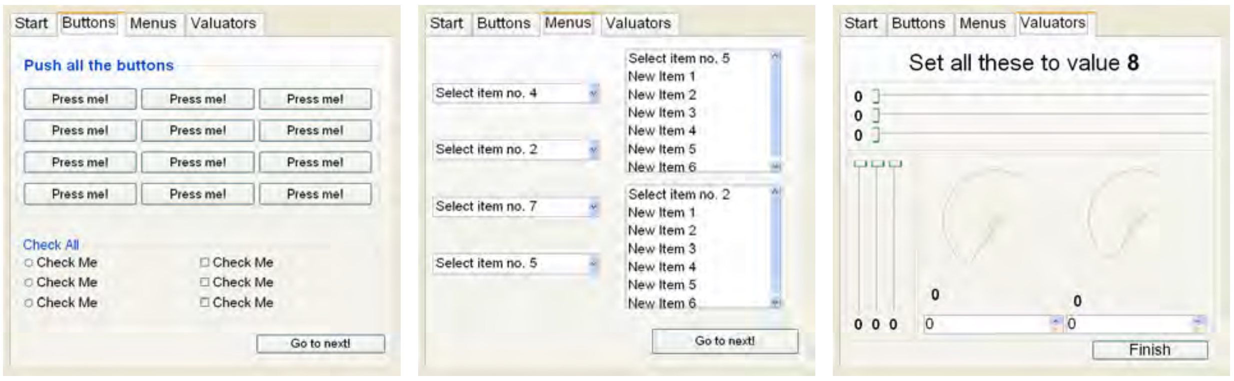


Figure 6.15: *Scenarios used in our experiment: (a) direct manipulation of a window through raycasting, (b) manipulation through a virtual pad rotated 45 degrees with respect to the window and (c) manipulation through an 50% scaled virtual pad.*



(a) (b) (c)

Figure 6.16: *The test dialogs used for the evaluation of the Virtual Pad metaphor*

periment users were provided with a short training session (5 min) which required them to complete a few practice trials.

As a dependent variables we recorded the time-to-complete the task and the number of clicks. Also, at the end of the experiment, users were provided with a short questionnaire.

The dialogs used in the experiments are shown in Figure 6.16. The first two dialogs are designed to measure task performance on selecting small/middle size objects. The first dialog contains different kinds of buttons whereas the second dialog includes combo boxes and selection lists. The third dialog is designed also to measure speed but putting the emphasis on manipulation rather than on selection. In all cases the label attached to each widget indicates the requested task, so users can be more focused at purely interaction tasks. For all dialogs, users were requested to complete the involved tasks as quickly as possible

Apparatus

All the experiments were conducted on a four-sided CAVE with a 6-DOF wanda and a Polhemus Fastrak tracking system with 2 receivers providing 60 updates/s with 4 ms latency. The virtual window used in the experiments was initially placed at 1.5 m from the CAVE center, covering about 20 degrees of the user's field-of-view.

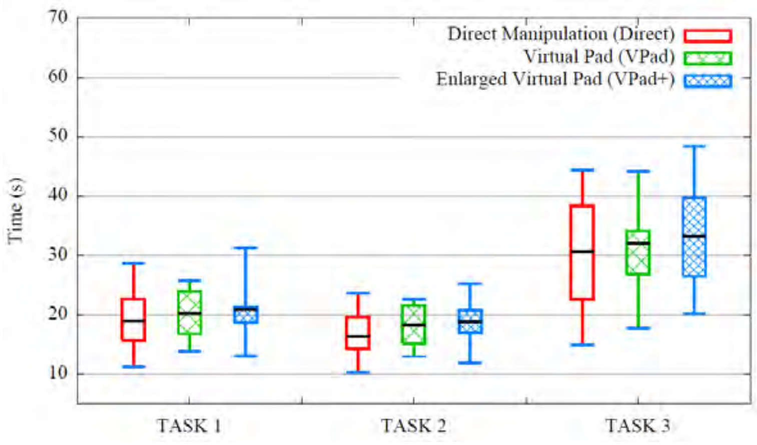
Participants

Thirteen users (undergraduate and graduate students) participated in the study, aged 22-38, 11 male and 2 female. Most participants (6) had some experience with VE applications; 5 had no experience and 2 were experienced users.

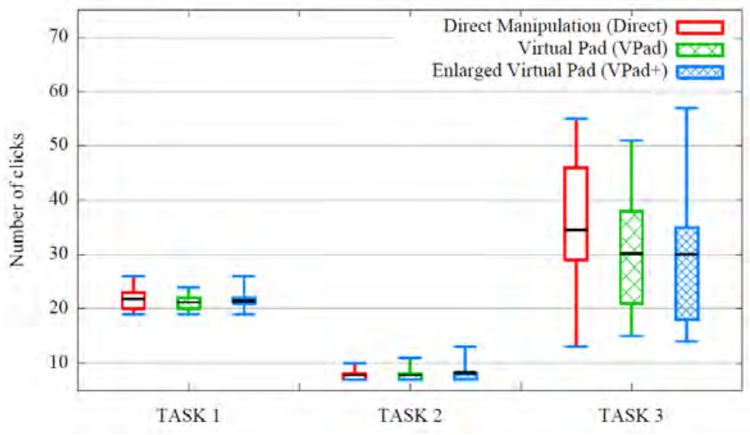
Results

Figure 6.17a shows the completion time for each task. Tasks 1-3 correspond to the dialogs (a)-(c) shown in Figure 6.16. Note that on average the completion times are quite similar on tasks 1 and 2 which emphasize on selection. We performed a correlated samples one-way ANOVA on the data with completion time as the dependent variable and the decoupling (Direct, VPad or VPad+) as the independent variable. We found no significant differences in tasks 1 and 3, but it showed a significant difference for task 2 ($F = 6.29; p < 0.01$). Tukey pair-wise HSD tests revealed a significant difference between Direct vs VPad ($p < 0.05$) and between Direct vs VPad+ ($p < 0.01$).

Regarding button clicks, Figure 6.17b shows the button clicks for each task. Note that in tasks emphasizing on selection (1 and 2), the number of clicks is a good measure of the number of mistakes. This does not applies to task 3, because sliders and spin boxes support different interaction modalities. For example, the value of the spin box can be modified



(a)



(b)

Figure 6.17: *(a) Time to complete the tasks involved for each dialog. (b) Number of button clicks performed during each task.*

with repeated clicks or by holding down the wanda button. A correlated samples one-way ANOVA on the data with button clicks as the dependent variable revealed no significant differences.

Survey

After the experiments, subjects were requested to rate the arm fatigue with VPad and Direct manipulation. As expected, all users reported less muscular fatigue when using VPad. Some users even find it comfortable to slightly rest the wanda against the body. Virtual pads in a completely horizontal orientation are also perfectly possible and greatly reduce arm fatigue. Note that direct manipulation with a virtual window in this position is unfeasible on most projection-based VR systems such as the CAVE using front projection on the floor screen.

Discussion

The Virtual Pad metaphor has many points in common with pen-and-tablet interfaces [55] and transparent props [104], but also important differences. Besides the fact that it does not require the user to hold any physical prop, the basic difference is that the Virtual Pad decouples the motor and the visual space. It allows to scale virtual windows avoiding visual obtrusion and readability issues while the motor space can be sized according to user preferences and the desired speed/accuracy trade-off.

From the user evaluation, we can state that decoupling the motor space from the visual space does not introduce significant differences in interaction time and erroneous selections. This is also supported by the work of Wang and MacKenzie in [124] which revealed that moderate disparity between motor and visual spaces does not have significant effects on interaction performance.

Interestingly, although we expected differences between VPad the VPad+ conditions, it did not happen. The results somehow match the experiments with 2D mouse based interfaces where constant CD gain conditions do not have significant differences in user performance [4]. Furthermore, note that the corresponding dialog on Figure 6.16b includes several combo boxes. A click on a combo box caused a new pop-up window to appear, which automatically received the focus. In our experiment, focus changes did not affect the pad size. Therefore movements on the pad were mapped to the smaller pop-up window (about 33% of the parent's width), resulting in a much lower CD ratio. We believe that the unexpected CD ratio change (twice per pop-up, eight times per trial) were particularly distracting, since no significant performance differences were found between VPad and VPad+ for Tasks 1 and 3.

In addition, we also expected improved accuracy rates, buy it neither happened. As observed by Wingrave et al. in [132], decreasing the degree of accuracy required may induce users to decrease their accuracy; they no longer need to be precise. This might also happened for the VPad+ condition.

In summary, the Virtual Pad metaphor is recommended when the footprint is a major requirement or when the GUI has not been designed specifically for a VE. By allowing to readjust the control space, user can interact with virtual windows with a more comfortable position. Although statistically significant, the average time for VPad on task 2 was only 11% higher than Direct, so this overhead can be easily accepted in exchange for a more comfortable and flexible interaction.

Moreover, the virtual pad can be used to define the area of the motor space allowed for interaction with virtual windows, thus allowing to seamlessly decouple the interaction between the virtual environment and the virtual windows. The virtual pad can be invisible, but reachable due to proprioception.

Conclusions

The thesis this PhD dissertation is defending can be summarized as follows:

> *Although 3D interaction techniques for object selection have been used for many years, they still exhibit major limitations regarding effective, accurate selection of targets in real-world VR applications. Some of these limitations are concerned with visual feedback issues (occlusion, visibility mismatch, depth perception in stereoscopic displays) and the inherent features of the human motor system (instability when interacting in free space, speed-accuracy trade-off, neuromotor noise). More efficient 3D interaction techniques can be designed by devising new strategies for controlling the selection tool and for providing appropriate visual feedback, drawing the inspiration from Fitts' law, occlusion management literature and depth perception studies.*

We have provided theoretical and empirical evidence about important limitations of major virtual pointing techniques. Although virtual pointing techniques typically deliver superior performance than their virtual hand counterparts, they require appropriate continuous visual feedback about the scene and the selection tool to complement the information derived from proprioception.

Providing precise visual feedback about the selection tool on stereoscopic displays is a challenging problem. Visual feedback solutions should try to encompass the opposite goals of guaranteeing a clear, unoccluded indication of the selection tool while minimizing the disagreement among parallax, interposition and perspective depth cues. We have shown that drawing a ray extending out from the user's hand, which is the most typical representation of the selection tool, hinders selection tasks as the perception of the tool's tilt angle relies on depth perception.

Occlusion is a major source of difficulties when indicating objects in complex 3D scenes. We have shown the negative effects of occlusion and visibility mismatch (both within users and among users) in selection and referral tasks.

Eye-hand visibility mismatch has been proven to be a factor leading to a significant loss of performance in all hand-rooted pointing techniques. Our analysis suggests that for these techniques the effective width of an object should be computed in terms of the part of the object simultatenously unoccluded from the user's hand and eye locations. This results in a paradoxical situation where some objects might exhibit a null effective width despite being visible on the screen.

In order to overcome these visual-feedback limitations, we have proposed a new strategy for controlling the selection tool dubbed Ray Casting from the Eye (RCE). RCE encompasses the efficient rotational control of hand-rooted techniques (thus minimizing the physical effort) with the mismatch-free feature of eye-rooted techniques. RCE clearly outperforms classic ray casting, particularly in real-world, complex 3D scenes. RCE is especially effective when used in combination with the newly introduced viewfinder metaphor. By flattening the surroundings of the pointing direction, the viewfinder locally turns a 3D object selection task into a simpler 2D selection task where depth perception is no longer an issue.

RCE control can be freely combined with further facilitation techniques for object selection such as speed-based adaptation of the CD-ratio. RCE combined with PRISM-based CD-ratio adjustment offers excellent performance even in cluttered scenes with objects spanning a large range of depth values.

The analysis of major limitations of existing selection techniques and the proposed solutions were published in:

- Ferran Argelaguet and Carlos Andujar. **Efficient 3d pointing selection in cluttered virtual environments**. IEEE Computer Graphics and Applications, 29(6):34-43, 2009.
- Ferran Argelaguet, Carlos Andujar and Ramon Trueba. **Overcoming eye-hand visibility mismatch in 3D pointing selection**. In Proceedings of the 15th ACM Symposium on Virtual Reality Software and Technology, VRST '08, pages 43-46, 2008.
- Ferran Argelaguet and Carlos Andujar. **Visual feedback techniques for virtual pointing on stereoscopic displays**. In Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology, VRST '09, pages 163-170, 2009.

Visibility mismatch among users in collaborative virtual environments is a common hindrance during referral tasks, where one user is willing to show some object or feature to another user. Typically, referral tasks are accomplished using selection techniques, with one user (the presenter) selecting an object and asking the other users (observers) to look at it. Viewpoint mismatch often results in the selected object to be visible only for the presenter. According to proxemics theory, users carrying out a collaborative tasks should stay at the close phase of the social distance (about 1.2 to 2.1 m in Western culture). However, our experiment performed in a co-located CVE showed that the viewpoint mismatch forces observers to stay close to the presenter. Such a close distance between users does not agree with social protocols and can cause users to bump accidentally into other users. Furthermore, this discourages other users to freely navigate through the virtual environment.

In order to better support referral tasks, we explored which occlusion management techniques ameliorate this situation and whether they hinder or not the spatial understanding of the virtual environment. X-Ray vision techniques have been found to provide the best solution for inter-personal occlusion. We have shown that semi-transparency and cut-away views are suitable for guaranteeing the visibility of the objects being referred. The empirical evaluation of these techniques showed that observers are able to gather the same amount of spatial understanding with these techniques than when following the presenter, with additional advantages related to freedom of movement and social protocols.

The evaluation of show-through techniques was published in:

- Ferran Argelaguet, André Kunert, Alexander Kulik, Carlos Andujar, and Bernd Froehlich. **See-through techniques for referential awareness in collaborative virtual reality**. International Journal of Human-Computer Studies. Volume 69, Issue 6, June 2011, Pages 387-400.
- Ferran Argelaguet, André Kunert, Alexander Kulik and Bernd Froehlich. **Improving co-located collaboration with show-through techniques**. In IEEE Symposium on 3D User Interfaces 2010, pages 55-92, 2010.

The review of Fitts' law and existing pointing facilitation techniques for 3D object selection, revealed that although expanding targets techniques have been used to improve 2D target acquisition tasks, they were never applied for 3D object selection tasks.

Following Fitts' law guidelines, we proposed two orthogonal techniques for improving 3D target acquisition tasks: Dynamic Scaling and Forced Disoclussion. Both techniques increase the effective size of potential targets, thus reducing the accuracy needed during corrective movements. In an ideal scenario, increasing the size of potential targets should result in improvements on selection performance and reduced error rates. However, the user study showed that their drawbacks exceeded their benefits. As with most of Fitts' law based techniques which increase the size of potential targets, time and error rate improvements are more apparent in situations where targets are isolated. In cluttered environments, the benefits of these techniques tend to degrade, and can even be detrimental to selection time. First, unwanted expansions of objects surrounding the target might discourage its selection and second, they might break the object layout.

Both approaches presented additional limitations related to the eye-hand visibility mismatch, as they were evaluated in combination with classic ray casting selection. While Dynamic Scaling computes the potential occlusions among objects in image space, Forced Disocclusion required an alternative ray-scene intersection. Nevertheless, these limitations are avoided when used in combination with Ray Casting from the Eye.

The Dynamic Scaling and the Forced Disocclussion approaches were presented in:

- Ferran Argelaguet and Carlos Andujar. **Improving 3D selection in immersive environments through expanding targets**. SG '08 Proceedings of the 9th international symposium on Smart Graphics, 5166/2008, pages 45-57, 2008.

The operation of 2D widgets using 3D input devices relies also on selection techniques. The last contribution of this dissertation (although the first contribution chronologically) is concerned about improving the deployment and the interaction of 2D graphical user interfaces embedded in virtual environments. First, we proposed a cost-effective approach to embed 2D GUIs into virtual environments. Rather than providing new tools, we focused on how to adapt well established 2D GUI toolkits into VEs. This greatly simplifies the GUI design and prototyping steps as developers and designers benefit from the widget functionality and the design tools provided by existing 2D toolkits.

When 2D GUI toolkits designed for desktop computers are deployed to virtual environments, they might include small, nearby buttons which can be difficult to select and manipulate using standard 3D interaction techniques. Although 2D toolkits support widget customization, complex 2D GUIs will require a complete redesign increasing the development step.

Instead of developing methods to automatically redesigning existing 2D GUIs, we developed new strategies for controlling the selection tool for the particular case of selecting 2D widgets embedded in 3D space. We explored whether decreasing the CD ratio according to the size of virtual windows (Friction Surfaces) and decoupling the motor and the visual space (Virtual Pad) allow for increased precision and increased comfort when interacting with 2D GUIs embedded in VEs.

The evaluation of Friction Surfaces showed that selection of small widgets, in comparison with standard raycasting selection, is enhanced both in terms of user performance and error rate. The reduction of the CD ratio increased the accuracy of selection and manipulation tasks without decreasing selection time.

Regarding the Virtual Pad, its evaluation showed that the decoupling between the motor and the visual space results in no significant performance penalty with respect to direct interaction. When the reduction of the visual footprint of the GUI is a strong requirement, the virtual pad allows the user to adjust its working space to maximize comfort without decreasing performance.

Contributions regarding 2D GUI interaction were published in:

- Carlos Andujar and Ferran Argelaguet. **Anisomorphic ray-casting manipulation for interacting with 2D GUIs**. Computer & Graphics, 31(1):15-25, 2007.
- Carlos Andujar, Marta Fairen and Ferran Argelaguet. **A cost-effective approach for developing application-control guis for virtual environments**. In IEEE Symposium on 3D User Interfaces (3DUI 2006), pages 45-52, 2006.
- Carlos Andujar and Ferran Argelaguet. **Virtual pads: Decoupling motor space and visual space for flexible manipulation of 2D windows within VEs**. In IEEE Symposium on 3D User Interfaces (3DUI 2007), pages 99-106. 2007.
- Carlos Andujar and Ferran Argelaguet. **Friction surfaces: Scaled ray-casting manipulation for interacting with 2D GUIs**. 12th Eurographics Symposium on Virtual Environments, pages 101-108, 2006.

As a concluding remark, we would like to note that the interaction techniques proposed in this dissertation do not consider any specific input and output device, although they are obviously constrained by the limitations of current input and output devices. Improvements in tracking technology will allow for a more accurate tracking of the user's actions, and better displays will enhance the user's perception of the virtual environment. We believe though that the major contributions of this dissertation will still be valid despite forthcoming advances in VR technology. We can provide the user with extremely realistic volumetric displays and perfectly accurate tracking systems, but pointing gestures will be still limited by the human motor system, which is unlikely to improve in the near future.

Although new and better interaction techniques will arise, or in a mid-term future, brain-computer interfaces might partially replace traditional gesture-based interfaces, we believe that the presented techniques are a good choice for both current and upcoming VR applications.

Bibliography

[1] Johnny Accot and Shumin Zhai. Refining Fitts' law models for bivariate pointing. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 193–200. ACM, 2003.

[2] Maneesh Agrawala, Andrew C. Beers, Ian McDowall, Bernd Fröhlich, Mark Bolas, and Pat Hanrahan. The two-user responsive workbench: support for collaboration through individual views of a shared space. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 327–332. ACM Press/Addison-Wesley Publishing Co., 1997.

[3] American Psychological Association. Certification of compliance with ethical principles. http://www.apa.org/ethics/code/index.aspx.

[4] Ravin Balakrishnan. "Beating" Fitts' Law: Virtual enhancements for pointing facilitation. *International Journal of Human-Computer Studies*, pages 857–874, 2004.

[5] Jim Blascovich, Andrew C. Beall, Jack M. Loomis, Jeremy N. Bailenson, Jeremy N. Bailenson, and In Neuromancer. Interpersonal distance in immersive virtual environments. *Personality and Social Psychology Bulletin*, 29:1–15, 2003.

[6] Joan De Boeck, Tom De Weyer, Chris Raymaekers, and Karin Coninx. Using the Non-Dominant Hand for Selection in 3D. *3DUI '06: Proceedings of the 3D User Interfaces*, pages 53–58, 2006.

[7] Doug A. Bowman, Brian Badillo, and Dhruv Manek. Evaluating the need for display-specific and device-specific 3D interaction techniques. In *ICVR'07: Proceedings of the 2nd international conference on Virtual reality*, pages 195–204, Berlin, Heidelberg, 2007. Springer-Verlag.

[8] Doug A. Bowman, Joseph L. Gabbard, and Deborah Hix. A survey of usability evaluation in virtual environments: classification and comparison of methods. *Presence: Teleoperators and Virtual Environments*, 11(4):404–424, 2002.

[9] Doug A. Bowman and Larry F. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. *SI3D '97: Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 35–ff, 1997.

[10] Doug A. Bowman, Larry F. Hodges, and Jay Bolter. The virtual venue: User-computer interaction in information-rich virtual environments. *Presence: Teleoperators and Virtual Environments*, 7(5):478–493, 1998.

[11] Doug A. Bowman, Donald B. Johnson, and Larry F. Hodges. Testbed evaluation of virtual environment interaction techniques. In *VRST '99: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 26–33. ACM, 1999.

[12] Doug A. Bowman, Ernest Kruijff, Joseph J. LaViola, and Ivan Poupyrev. *3D User Interfaces: Theory and Practice.* Addison Wesley, 2004.

[13] Doug A. Bowman, Chadwick A. Wingrave, and J Campbell. Using pinch gloves for both natural and abstract interaction techniques in virtual environments. *HCI International*, pages 629–633, 2001.

[14] Stephen Brewster. Multimodal feedback for the acquisition of small targets. *Ergonomics*, 48:1129–1150(22), 2005.

[15] Matthias Bues, Roland Blach, and Frank Haselberger. Sensing Surfaces: bringing the desktop into virtual environments. In *EGVE '03: Proceedings of the workshop on Virtual Environments 2003*, pages 303–304. ACM, 2003.

[16] Michael Burns and Adam Finkelstein. Adaptive cutaways for comprehensible rendering of polygonal scenes. *ACM Transactions on Graphics*, 27(5):1–7, 2008.

[17] S. K. Card, W. K. English, and B.J. Burr. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a crt. *Ergonomics*, 2:601–613, 1978.

[18] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. A morphological analysis of the design space of input devices. *ACM Transactions on Information Systems*, 9(2):99–122, 1991.

[19] Géry Casiez, Daniel Vogel, Ravin Balakrishnan, and Andy Cockburn. The impact of control-display gain on user performance in pointing tasks. In *Human-Computer Interaction*, volume 23, pages 215–250. Taylor & Francis, 2009.

[20] Pedro Concejero Cerezo. Código ético de la investigación en usabilidad e Interacción Persona-Ordenador. Pruebas con usuarios. Asociación Interacción Persona-Ordenador, 2006.

[21] Jeff Chastine and Ying Zhu. The cost of supporting references in collaborative augmented reality. In *Proceedings of graphics interface 2008*, GI '08, pages 275–282. Canadian Information Processing Society, 2008.

[22] Jeffrey W. Chastine, Kristine Nagel, Ying Zhu, and Luca Yearsovich. Understanding the design space of referencing in collaborative augmented reality environments. In *Proceedings of Graphics Interface 2007*, GI '07, pages 207–214. ACM, 2007.

[23] Andy Cockburn and Philip Brock. Human on-line response to visual and motor target expansion. *GI '06: Proceedings of Graphics Interface 2006*, pages 81–87, 2006.

[24] Chris Coffin and Tobias Hollerer. Interactive perspective cut-away views for general 3d scenes. In *Proceedings of the IEEE 3D User Interfaces*, 3DUI '06, pages 25–28, 2006.

[25] Infiscape Corporation. VRJuggler - http://www.vrjuggler.org.

[26] Nokia Corporation. Qt - cross platform application and UI framework. http://qt.nokia.com.

[27] E. Crossman. The measurement of perceptual load in manual operations. Master's thesis, PhD thesis, University of Birmingham, 1956.

[28] P. Crossman, E. Goodeve. Feedback control of hand-movement and Fitts' law. *The Quarterly Journal of Experimental Psychology Section A: Human Experimental Psychology*, 35:251–278, 1983.

[29] Raimund Dachselt and Anett Hübner. A survey and taxonomy of 3D menu techniques. *EGVE 06: Proceedings of the 12th Eurographics Symposium*, pages 89–99, 2006.

[30] Raimund Dachselt and Anett Hübner. Three-dimensional menus: A survey and taxonomy. *Computers & Graphics*, 31(1):53–65, 2007.

[31] Philippe Dax. VREng - virtual reality engine. http://sourceforge.net/projects/vreng/.

[32] Gerwin de Haan, Michal Koutek, and Frits H. Post. IntenSelect: Using dynamic object rating for assisting 3D object selection. *Eurographics Symposium on Virtual Environments*, pages 201–209, 2005.

[33] J. Diepstraten, D. Weiskopf, and T. Ertl. Transparency in interactive technical illustrations. *Eurographics*, 21(3):317–325, 2002.

[34] Stephen DiVerdi, Daniel Nurmi, and Tobias Höllerer. ARWin-a desktop Augmented Reality Window manager. In *ISMAR '03: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, page 298, 2003.

[35] Phillip Dykstra. X11 in virtual environments: combining computer interaction methodologies. *X Resour.*, (9):195–204, 1994.

[36] N. Elmqvist and P. Tsigas. A taxonomy of 3D occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1095 –1109, 2008.

[37] Niklas Elmqvist, Ulf Assarsson, and Philippas Tsigas. Employing dynamic transparency for 3D occlusion management: Design issues and evaluation. *Human-Computer Interaction INTERACT 2007*, pages 532–545, 2008.

[38] P. Fitts. The information capacity of the human motor system is controlled by the amplitude of movement. *Journal of Experimental Psychology*, 6(47):381–391, 1954.

[39] P. Fitts and J Peterson. Information capacity of discrete motor response. *Journal of Experimental Psychology*, 2(67):103–112, 1964.

[40] Andrew Forsberg, Kenneth Herndon, and Robert Zeleznik. Aperture based selection for immersive virtual environments. *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 95–96, 1996.

[41] Scott Frees and Drew Kessler. Precise and Rapid Interaction throught Scaled Manipulation in immersive virtual environments. *Proc. of IEEE Virtual Reality 2005*, pages 99–106, 2005.

[42] Scott Frees, G. Drew Kessler, and Edwin Kay. PRISM interaction for enhancing control in immersive virtual environments. *ACM Transactions on Computer-Human Interaction*, 14(1):2, 2007.

[43] Bernd Fröhlich, Roland Blach, Oliver Stefani, Jan Hochstrate, Jörg Hoffmann, Karsten Klüger, and Matthias Bues. Implementing multi-viewer stereo displays. In *WSCG (Full Papers)*, pages 139–146, 2005.

[44] Joseph L. Gabbard. A taxonomy of usability characteristics for virtual environmnets. Master's thesis, Departament of Computer Science, Virginia Tech, 1997.

[45] Joseph L. Gabbard, Deborah Hix, and J. Edward Swan. User-centered design and evaluation of virtual environments. *IEEE Computer Graphics and Applications*, 19(6):51–59, 1999.

[46] T Germer, T Götzelmann, M Spindler, and Thomas Strothotte. Springlens : Distributed nonlinear magnifications. *Eurographics'06 Short Papers*, pages 123–126, 2006.

[47] Tovi Grossman and Ravin Balakrishnan. Pointing at trivariate targets in 3D environments. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 447–454. ACM, 2004.

[48] Tovi Grossman and Ravin Balakrishnan. The design and evaluation of selection techniques for 3D volumetric displays. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 3–12. ACM, 2006.

[49] Edward T. Hall. *The Hidden Dimension*. Doubleday, 1966.

[50] Patrick Hartling and Carolina Cruz-Neira. Tweek: A framework for cross-display graphical user interfaces. In *Computational Science and Its Applications - ICCSA 2005*, volume 3482 of *Lecture Notes in Computer Science*, pages 317–329. Springer Berlin / Heidelberg, 2005.

[51] Kenneth P. Herndon, Andries van Dam, and Michael Gleicher. The challenges of 3D interaction: a CHI '94 workshop. *SIGCHI Bull.*, 26(4):36–43, 1994.

[52] A. Hill and A. Johnson. Withindows: A framework for transitional desktop and immersive user interfaces. In *IEEE Symposium on 3D User Interfaces 2008*, pages 3–10, March 2008.

[53] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. Passive real-world interface props for neurosurgical visualization. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 452–458. ACM, 1994.

[54] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. A survey of design issues in spatial input. In *UIST '94: Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 213–222. ACM, 1994.

[55] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '97, pages 234–241. ACM, 1997.

[56] Liang J. and Green M. JDcad: A highly interactive 3D modeling system. *Computers & Graphics 18*, 4:499–506, 1994.

[57] Robert J.K. Jacob, Audrey Girouard, Leanne M. Hirshfield, Michael S. Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. Reality-based interaction: a framework for post-WIMP interfaces. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 201–210. ACM, 2008.

[58] R. J. Jagacinski and D. L. Monk. Fitts' law in two dimensions with hand and head movements. *Journal of Motor Behavior*, 17:77–95, 1985.

[59] Joseph J. LaViola Jr. A survey of hand posture and gesture recognition techniques and technology. Technical Report CS-99-11, Brown University. Computer Science Department,, 1999.

[60] Y. Jung, S. Webel, M. Olbrich, T. Drevensek, T. Franke, M. Roth, and D. Fellner. Interactive textures as spatial user interfaces in x3d. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 147–150. ACM, 2010.

[61] John Kelso, Steven G. Satterfield, Lance E. Arsenault, Peter M. Ketchan, and Ronald D. Kriz. DIVERSE: a framework for building extensible and reconfigurable device-independent virtual environments and distributed asynchronous simulations. *Presence: Teleoperators and Virtual Environments*, 12(1):19–36, 2003.

[62] Stuart T. Klapp. Feedback versus motor programming in the control of aimed movements. *Journal of Experimental Psychology: Human Perception and Performance*, 1:147–153, 1975.

[63] Werner König, Jens Gerken, Stefan Dierdorf, and Harald Reiterer. Adaptive pointing - design and evaluation of a precision enhancing technique for absolute pointing devices. In *Human-Computer Interaction, INTERACT 2009*, volume 5726 of *Lecture Notes in Computer Science*, pages 658–671. Springer Berlin / Heidelberg, 2009.

[64] Robert E. Kraut, Mark D. Miller, and Jane Siegel. Collaboration in performance of physical tasks: effects on outcomes and communication. In *Proceedings of the 1996 ACM conference on Computer Supported Cooperative Work*, CSCW '96, pages 57–66, 1996.

[65] Per Ola Kristensson and Alan Blackwell. Ethics in large-scale user studies: Guidelines vs. practice. In *CHI 2011 Workshop on Ethics in Large Scale Trials & User Generated Content*, 2011.

[66] Daniel Larimer and Doug A. Bowman. VEWL: A framework for building a windowing interface in a virtual environment. *in Proceedings of IFIP TC13 International Conference on Human-Computer Interaction, Interact'2003*, pages 1–5, 2003.

[67] Joseph J. LaViola, Jr. A discussion of cybersickness in virtual environments. *SIGCHI Bulletin*, 32:47–56, January 2000.

[68] Geoff Leach, Ghassan al Qaimari, Mark Grieve, Noel Jinks, and Cameron McKay. Elements of a three-dimensional graphical user interface. *INTERACT '97: Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, pages 69–76, 1997.

[69] SangYoon Lee, Jinseok Seo, Gerard J. Kim, and Chan-Mo Park. Evaluation of pointing techniques for ray casting selection in virtual environments. *Third International Conference on Virtual Reality and Its Application in Industry*, 4756(1):38–44, 2003.

[70] Q. Limbourg, J. Vanderdonckt, L. Bouillon B. Michotte, and V. Lopez. UsiXML: A language supporting multi-path development of user interfaces. *In 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCIDSVIS*, 2004.

[71] R.W. Lindeman, J.L. Sibert, and J.K. Hahn. Hand-held windows: towards effective 2D interaction in immersive virtual environments. In *Virtual Reality, 1999. Proceedings., IEEE*, pages 205 –212, 1999.

[72] R. Linderman, J. Sibert, and J. Hahn. Towards usable VR: an empirical study of user interfaces for immersive virtual environments. *Proceedings of the SIHCHI conference on Human factors in computing systems.*, pages 64–71, 1999.

[73] C. L. MacKenzie, R. G. Marteniuk, C. Dugas, D. Liske, and B. Eickmeier. Three-dimensional movement trajectories in Fitts' task: Implications for control. *The Quarterly Journal of Experimental Psychology Section A: Human Experimental Psychology*, 39(4):629–647, 1987.

[74] I. Scott MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7:91–139, 1992.

[75] I. Scott MacKenzie and William Buxton. Extending Fitts' law to two-dimensional tasks. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 219–226. ACM, 1992.

[76] I. Scott MacKenzie, Tatu Kauppinen, and Miika Silfverberg. Accuracy measures for evaluating computer pointing devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '01, pages 9–16. ACM, 2001.

[77] I. Scott MacKenzie and Colin Ware. Lag as a determinant of human performance in interactive systems. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, pages 488–493. ACM, 1993.

[78] José Pascual Molina Massó, Jean Vanderdonckt, Francisco Montero Simarro, and Pascual González López. Towards virtualization of user interfaces based on UsiXML. *Web3D '05: Proceedings of the tenth international conference on 3D Web technology*, pages 169–178, 2005.

[79] Michael McGuffin and Ravin Balakrishnan. Acquisition of expanding targets. *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 57–64, 2002.

[80] Michael F. McTear. Spoken dialogue technology: enabling the conversational user interface. *ACM Comput. Surv.*, 34(1):90–169, 2002.

[81] David E. Meyer, Richard A. Abrams, Sylvan Kornblum, Charles E. Wright, and J. E. Keith Smith. Optimality in human motor performance: ideal control of rapid aimed movements. *Psychological Review*, 95:340–370, 1988.

[82] Mark R. Mine. Virtual environments interaction techniques. Technical Report TR94-018, Dept. of Computer Science, University of North Carolina at Chapel Hill, 1995.

[83] Mark R. Mine, Frederick P. Brooks, Jr., and Carlo Sequin. Moving objects in space: Exploiting proprioception in virtual-environment interaction. *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 19–26, 1997.

[84] Atsuo Murata. Extending effective target width in Fitt's law to two-dimensional pointing task. In *International Journal of Human-Computer Interaction*, volume 11(2), pages 137–152. L. Erlbaum Associates Inc., 1999.

[85] Atsuo Murata and Hirokazu Iwase. Extending Fitts' law to a three-dimensional pointing task. *Human Movement Science*, 20(6):791 – 805, 2001.

[86] K. Nieuwenhuizen, J.-B. Martens, Lei Liu, and R. van Liere. Insights from dividing 3D goal-directed movements into meaningful phases. *Computer Graphics and Applications, IEEE*, 29(6):44 –53, nov. 2009.

[87] Johanna Octavia, Chris Raymaekers, and Karin Coninx. Adaptation in virtual environments: conceptual framework and user models. *Multimedia Tools and Applications*, pages 1–22, 2010.

[88] Association of Computing Machinery. Code of ethics. http://www.acm.org/about/code-of-ethics.

[89] Alex Olwal, Hrvoje Benko, and Steven Feiner. SenseShapes: Using statistical geometry for object selection in a multimodal augmented reality system. In *ISMAR '03: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, page 300, 2003.

[90] Alex Olwal and Steven Feiner. The flexible pointer: An interaction technique for selection in augmented and virtual reality. *In Conference Supplement of UIST '03 (ACM Symposium on User Interface Software and Technology)*, pages 81–82, 2003.

[91] Noritaka Osawa, Kikuo Asai, and Fumihiko Saito. An interactive toolkit library for 3D applications: it3d. *EGVE '02: Proceedings of the workshop on Virtual environments 2002*, pages 149–157, 2002.

[92] Andriy Pavlovych and Wolfgang Stuerzlinger. The tradeoff between spatial jitter and latency in pointing tasks. In *EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 187–196. ACM, 2009.

[93] Hans Peter, Wyss Roland, and Blach Matthias Bues. ISith – intersection-based spatial interaction for two hands. *IEEE Symposium on 3D User Interfaces 2006*, pages 59–61, 2006.

[94] Jeffrey S. Pierce, Andrew Forsberg, Matthew J. Conway, Seung Hong, Robert Zeleznik, and Mark R. Mine. Image plane interaction techniques in 3D immersive environments. *Symposium on Interactive 3D Graphics*, page 5, 1997.

[95] I. Poupyrev and T. Ichikawa. Manipulating objects in virtual worlds: Categorization and empirical evaluation of interaction. *Journal of Visual Languages & Computing*, 10:19–35, 1999.

[96] Ivan Poupyrev, Mark Billinghutst, Suzanne Weghorst, and Tadao Ichikawa. The go-go interaction technique: Non-linear mapping for direct manipulation in VR. *ACM Symposium on User Interface Software and Technology*, pages 79–80, 1996.

[97] Ivan Poupyrev, Suzanne Weghorst, Mark Billinghurst, and Tadao Ichikawa. A framework and testbed for studying manipulation techniques for immersive vr. In *VRST '97: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 21–28. ACM, 1997.

[98] Ivan Poupyrev, Suzanne Weghorst, Mark Billinghutst, and Tadao Ichikawa. Egocentric object manipulation in virtual environments: Empirical evaluation of interaction techniques. *Computer Graphics Forum*, 3(17):41–52, 1998.

[99] Ivan Poupyrev, Suzanne Weghorst, and Sidney Fels. Non-isomorphic 3d rotational techniques. *CHI '00: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 540–547, 2000.

[100] Schmidt R.A., H.N. Zalaznik, B. Hawkins, J.S. Frank, and J.T. Quinn. Motor-output variability: A theory for the accuracy of rapid motor acts. *Psychological Review*, 86(5):415 – 451, 1979.

[101] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[102] G. Robertson, M. van Dantzich, D. Robbins, M. Czerwinski, K. Hinckley, K. Risden, D. Thiel, and V. Gorokhovsky. The task gallery: a 3D window manager. *Human Factors in Computing Systems*, pages 494–501, 2000.

[103] Holger Salzmann, Mathias Moehring, and Bernd Froehlich. Virtual vs. real-world pointing in two-user scenarios. *IEEE Virtual Reality Conference*, pages 127–130, 2009.

[104] Dieter Schmalstieg, L. Miguel Encarnação, and Zsolt Szalavári. Using transparent props for interaction with the virtual table. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, I3D '99, pages 147–153. ACM, 1999.

[105] Greg Schmidt, Yohan Baillot, Dennis G. Brown, Erik B. Tomlin, and J. Edward II Swan. Toward disambiguating multiple selections for frustum-based pointing. *3D User Interfaces*, pages 87–94, 2006.

[106] John B. Smelcer. Spatial and temporal characteristics of rapid cursor-positioning movements with electromechanical mice in human-computer interaction. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 35:431–458(28), 1993.

[107] Barton A. Smith, Janet Ho, Wendy Ark, and Shumin Zhai. Hand eye coordination patterns in target selection. In *Proceedings of the 2000 symposium on Eye tracking research & applications*, ETRA '00, pages 117–122. ACM, 2000.

[108] J. Marques Soares, P. Horain, and A. Bideau. Sharing and immersing applications in a 3D virtual inhabited world. In *5th virtual reality international conference (VRIC 2003)*, pages 27–31, 2003.

[109] Kay M. Stanney. *Handbook of virtual environments: design, implementation, and applications.* Lawrence Erlbaum Associates, 2002.

[110] Anthony Steed. Selection/towards a general model for selection in virtual environments. *IEEE Symposium on 3D User Interfaces 2006*, pages 103–110, 2006.

[111] Anthony Steed and C Parker. 3D selection strategies for head tracked and non-head tracked operation of spatially immersive displays. *8th International Immersive Projection Technology Workshop*, pages 163–170, 2004.

[112] Frank Steinicke, Timo Ropinski, and Klaus Hinrichs. Object selection in virtual environments with an improved virtual pointer metaphor. *Computer Vision and Graphics, International Conference, ICCVG 2004*, 32:320–326, 2004.

[113] Frank Stenicke, Timo Ropinski, Gerd Bruder, and Klaus Hinrichs. Interscopic user interface concepts for fish tank virtual reality systems. *Virtual Reality Conference, IEEE*, pages 27–34, 2007.

[114] Stanislav L. Stoev and Dieter Schmalstieg. Application and taxonomy of through-the-lens techniques. *VRST '02: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 57–64, 2002.

[115] Vildan Tanriverdi and Robert J. K. Jacob. Interacting with eye movements in virtual environments. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272. ACM, 2000.

[116] R.J. Teather, A. Pavlovych, W. Stuerzlinger, and I.S. MacKenzie. Effects of tracking technology, latency, and spatial jitter on object movement. In *IEEE Symposium on 3D User Interfaces 2009*, pages 43–50, 2009.

[117] Alexandre Topol. Immersion of Xwindow applications into a 3D workbench. In *CHI '00: extended abstracts on Human factors in computing systems*, pages 355–356. ACM, 2000.

[118] F. Tyndiuk, G. Thomas, V. Lespinet-Najib, and C. Schlick. Cognitive comparison of 3D interaction in front of large vs. small displays. *VRST '05: Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 117–123, 2005.

[119] Gerard P. van Galen and Willem P. de Jong. Fitts' law as the outcome of a dynamic noise filtering model of motor control. *Human Movement Science*, 14(4-5):539 – 571, 1995.

[120] Lode Vanacken, Tovi Grossman, and Karin Coninx. Exploring the effects of environment density and target visibility on object selection in 3D virtual environments. *IEEE Symposium on 3D User Interfaces, 3DUI '07.*, pages 115–122, 2007.

[121] Lode Vanacken, Chris Raymaekers, and Karin Coninx. Evaluating the influence of multimodal feedback on egocentric selection metaphors in virtual environments. *Haptic and Audio Interaction Design*, pages 12–23, 2006.

[122] Daniel Vogel and Ravin Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 33–42. ACM Press, 2005.

[123] Vrco. Cavelib - http://www.vrco.com.

[124] Y. Wang and C. MacKenzie. Effects of orientation disparity between haptic and graphic displays of objects in virtual environments. In *INTERACT '99*, pages 391–398, 1999.

[125] Colin Ware and Ravin Balakrishnan. Reaching for objects in VR displays: lag and frame rate. *ACM Transactions on Computer-Human Interaction*, 1(4):331–356, 1994.

[126] Colin Ware and Kathy Lowther. Selection using a one-eyed cursor in a fish tank VR environment. *ACM Trans. Comput.-Hum. Interact.*, 4(4):309–322, 1997.

[127] Kent Watsen and Rudolph P. Darken. A handheld computer as an interaction device to a virtual environment. *3rd International Immersive Projection Technology Workshop (IPTW'99)*, 1999.

[128] B. Watson, V. Spaulding, N. Walker, and W. Ribarsky. Evaluation of the effects of frame time variation on VR task performance. In *Virtual Reality Annual International Symposium*, pages 38–44. IEEE, 1997.

[129] W. Weaver and C. E. Shannon. *The Mathematical Theory of Communication*. Urbana, Illinois: University of Illinois Press, 1949.

[130] Laurie M. Wilcox, Robert S. Allison, Samuel Elfassy, and Cynthia Grelik. Personal space in virtual reality. *ACM Trans. Appl. Percept.*, 3:412–428, 2006.

[131] Chadwick A. Wingrave and Doug A. Bowman. Baseline factors for raycasting selection. In *Proceedings of Virtual Reality International*, pages 10 (CD–ROM proceedings), 2005.

[132] Chadwick A. Wingrave, Doug A. Bowman, and Naren Ramakrishnan. Towards preferences in virtual environment interfaces. In *EGVE '02: Proceedings of the workshop on Virtual environments 2002*, pages 63–72. Eurographics Association, 2002.

[133] Chadwick A. Wingrave, Joseph J. Laviola, Jr., and Doug A. Bowman. A natural, tiered and executable UIDL for 3D user interfaces based on concept-oriented design. *ACM Transactions on Computer-Human Interaction*, 16(4):1–36, 2009.

[134] Chadwick A. Wingrave, Ryan Tintner, Bruce N. Walker, Doug A. Bowman, and Larry F. Hodges. Exploring individual differences in raybased selection: Strategies and traits. *IEEE Virtual Reality 2005*, pages 163–170, 2005.

[135] R.S Woodworth. The accuracy of voluntary movements [monograph supplement]. *The psychological review*, 1899.

[136] Shumin Zhai, William Buxton, and Paul Milgram. The "silk cursor": investigating transparency for 3D target acquisition. *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 459–464, 1994.

[137] Shumin Zhai, Jing Kong, and Xiangshi Ren. Speed-accuracy tradeoff in Fitts' law tasks–on the equivalency of actual and nominal pointing precision". *International Journal of Human-Computer Studies*, 61(6):823 – 856, 2004.

[138] Shumin Zhai, Paul Milgram, and William Buxton. The influence of muscle groups on performance of multiple degree-of-freedom input. In *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, CHI '96, pages 308–315. ACM, 1996.