Anisomorphic ray-casting manipulation for interacting with 2D GUIs

C. Andujar F. Argelaguet

Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya,
Jordi Girona 1-3, Ed. Omega, 08034, Barcelona, Spain

Abstract

The accommodation of conventional 2D GUIs with Virtual Environments (VEs) can greatly enhance the possibilities of many VE applications. In this paper we present a variation of the well-known ray-casting technique for fast and accurate selection of 2D widgets over a virtual window immersed into a 3D world. The main idea is to provide a new interaction mode where hand rotations are scaled down so that the ray is constrained to intersect the active virtual window. This is accomplished by changing the control-display ratio between the orientation of the user's hand and the ray used for selection. Our technique uses a curved representation of the ray providing visual feedback of the orientation of both the input device and the selection ray. We have implemented this technique and evaluated its effectiveness in terms of performance and user preference. Our experiments on a four-sided CAVE indicate that the proposed technique can increase the speed and accuracy of component selection in 2D GUIs immersed into 3D worlds.

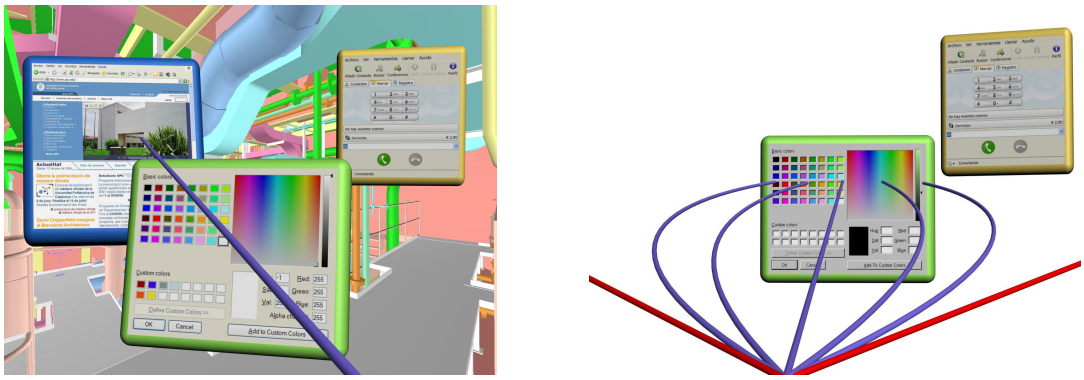
1 Introduction

In the recent years a considerable amount of research has been devoted to develop techniques for making 2D applications available from within VEs. As a consequence, a number of tools for launching and/or sharing existing 2D applications into VE and Augmented Reality (AR) applications have been proposed. *Hardware oriented* approaches provide access to external applications through separate devices such as PDAs and tablet PCs [1]. *Software oriented* approaches [2, 3, 4, 5] access 2D display contents generated by external applications and display them as texture-mapped rectangles. These techniques let the users interact with 2D applications through VR input devices such as gloves and 6-DOF sensors (see Figure 1).

There are two main software technologies for accessing 2D GUIs from a virtual world. The most common solution relies on a modified VNC client. VNC (Virtual Network Computing) [6] is a remote display system which allows viewing a computing desktop running elsewhere on a network. VNC provides a distribution mechanism for desktops by transmitting frame buffer contents to the remote client and receiving keyboard and pointing device events. VNC is the foundation of most systems providing immersion of 2D applications into 3D space [2, 3, 4]. An alternative technique consists in modifying an extensible 2D GUI toolkit so that native widgets are rendered directly onto a texture [5].

At a low level, user-interaction tasks with 2D GUIs immersed into 3D worlds can generally be characterized as selection or manipulation tasks [7]. However, selection and manipulation of 2D GUIs has some specific characteristics which must be considered when designing appropriate interaction techniques:

- Interaction with 2D GUIs is often dominated by selection rather than manipulation. Although selection is 2 DoF, manipulation (e.g. moving a slider or rotating a dial) is mostly 1 DoF (excluding the drag-and-drop metaphor).
- Application-control GUIs (such as those enabling the adjustment of display parameters)



(a) (b)

Figure 1: External 2D applications (web browser, color selector and P2P telephony) immersed into a 3D world through a modified VNC client (a). Visual feedback supporting the proposed technique (b). Several rays corresponding to different orientations of the input device are shown.

are typically manipulated frequently but in short intervals, thus making inappropriate the use of techniques that require some kind of user set-up (e.g. the Voodoo Dolls technique [8]) or have a large impact on the displayed image (e.g. the Scaled-World Grab technique [9]).

An additional problem involved in accessing external applications from within VEs is that the only possible adaptation of the GUI to the VE is to adjust the size and location of the virtual window containing the different widgets. For example, a window might include small, nearby buttons which can be difficult to select using 3D interaction (see e.g. the color selector in Figure 1-a). Therefore, interaction techniques for such applications could differ from those used for GUIs specifically tailored to 3D environments [10].

A fundamental technique for manipulating 3D objects is *pointing*. Pointing techniques enable the user to select and manipulate objects by simply pointing at them. A number of studies have demonstrated that pointing techniques often result on better selection effectiveness than *virtual hand* techniques [11]. Different variations of this technique differ basically on three aspects: the computation of the *pointing direction* (i.e. the mapping of the input device position and orientation onto the direction of the ray), the shape of the selection volume, and its visual representation (feedback). Ray-casting is the simplest and most popular pointing technique. In classic ray-casting implementations, the pointing direction is given directly by a virtual ray controlled by a 6-DOF sensor and visual feedback is provided by drawing a line extending out from the user's hand. Classic ray-casting implementations are isomorphic [12] in the sense that they use a direct, one-to-one mapping between hand rotation in the physical world and ray rotation in the virtual world.

Unfortunately, ray-casting techniques do not perform well when selecting small or distant objects [13]. 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 high level of angular accuracy. Accurate selection is also compromised by the hand instability caused by the absence of constraints on the hand movements (lack of physical support for manipulation) [14]. As a result, users attempting to select small buttons with this technique have to make a considerable effort to stabilize their wrist.

In this paper we present a new interaction technique for fast and accurate selection of 2D widgets over a virtual window. The main idea is to provide more accuracy to the ray-casting technique by changing the control-display (C-D) ratio [15] when the user is interacting with the active window.

The C-D ratio is a coefficient that maps the physical movement of the pointing device to the resulting on-screen movement in a system where there is an anisomorphism between the pointing device and

the display (e.g. a 2D mouse). The C-D ratio often defines the distance that must cover the device in the physical world (dx) to move the cursor on the screen by a given distance (dX). The adaptation of the C-D ratio dx/dX has been successfully used in many 2D and 3D interaction techniques (see [16] for a review). However, these techniques have been designed for manipulating 3D objects and thus do not address the specific problems involved in 2D GUI manipulation.

Our technique adapts the C-D ratio in order to scale down hand rotations and enable accurate selection and manipulation of small GUI objects. When the user is pointing at a virtual window, we increase the CD-ratio between the user's hand and the ray used for selection, so that the ray rotates more slowly than the user's hand, thus reducing the effect of hand instability. Unlike other techniques designed for accurate 3D object manipulation which define the C-D ratio inversely proportional to the hand speed [16], we compute the C-D ratio by considering the size and position of the virtual window relative to the user's hand at the moment the scaled mode is activated. While the scaled mode is active, the C-D ratio remains constant. Our technique uses a curved representation of the ray providing integrated visual feedback of both the orientation of the input device and the selection ray. We call this technique *friction surfaces* because the users' feeling in scaled mode is that they control a flexible ray that gets curved as it moves over a virtual friction surface defined by the 2D window (see Figure 1-b).

We conducted a usability evaluation to measure the performance and effectiveness of the technique. Our experiments on a four-sided CAVE indicate that the proposed technique can be used to increase the speed and accuracy of component selection in 2D GUIs immersed into 3D worlds.

The main contributions of the paper are:

- A new technique for interacting with 2D applications immersed into VEs. The technique has its roots in ray-casting selection and C-D based techniques [16] but adopts a completely different approach for activation/deactivation of the scaled mode and for computing the C-D ratio. One of the advantages over approaches using dynamic adjustment of the C-D ratio is that the angular difference between the user's hand orientation and the selection ray orientation is bounded by a user-defined constant.
- An evaluation of the performance and usability of the technique on a four-sided CAVE that indicates that it is particularly suitable for interaction with external applications immersed into VEs.
- A comparison with isomorphic ray-casting and with other approaches modifying the C-D ratio, and particularly with [17].

The rest of the paper is organized as follows. Section 2 reviews related work on interaction techniques for accurate selection and/or manipulation. Section 3 describes the proposed interaction technique and discusses the main differences with related approaches. We present effectiveness and usability results in Section 4, and provide concluding remarks in Section 5.

2 Previous Work

A number of 3D interaction techniques have been proposed for manipulating objects in immersive VEs, including *exocentric techniques* [18], *egocentric techniques* such as virtual hand and virtual pointer [19] and *hybrid techniques* [20, 9, 8]. The ray-casting technique is a powerful virtual pointer technique for 3D manipulation. However, classic ray-casting does not perform well when selecting small or distant objects [13]. Indeed, accurate interaction with small or distant objects is one of the main challenges in 3D manipulation. A number of techniques have been proposed for achieving more accuracy on such tasks.

One technique is to explicitly scale or zoom in the workspace in order to provide accurate manipulation. The WIM (World-In-Miniature) [18] is an exocentric technique that provides the user with a miniature handheld model of the VE. Scaled-World grab [9] provides a manipulation mode where the entire VE around the user's viewpoint is scaled down so that the selected object can be manipulated using the virtual hand technique. HOMER [20] uses ray-casting the select an object and then the user's virtual hand instantly moves to the object and attaches to it. The Voodoo Dolls [8] is a two-handed manipulation technique that enables

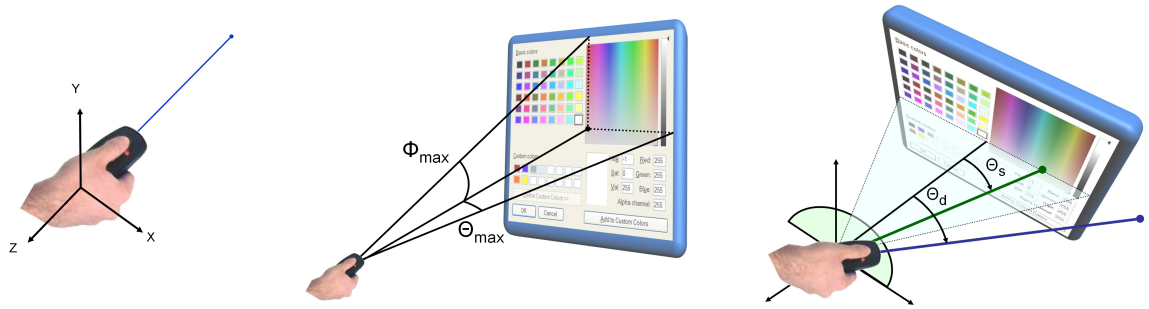
users to scale their workspace by selecting a voodoo doll of appropriate size. All these techniques put the emphasis on 3D object manipulation rather than selection.

The use of physical props to constrain the interaction can help to reduce hand instability [14]. Pen-and-tablet interfaces [21] register a virtual window with a physical prop held in the non-dominant hand. Users interact with these handheld windows using either a virtual hand or a virtual pointer held in the dominant hand. Hand-held windows also take advantage of the proprioceptive sense as they are close to the non-dominant hand. Some systems use hand-held windows whose physical prop is a lightweight, transparent surface that the user carries around, increasing precision. The Transparent Props [22] technique consists of a tracked hand-held pen and a pad. The pad can serve as a palette for tools and controls as well as a window-like see-through interface. Although combining transparent props with other two-handed techniques can be difficult and storing the physical surface when not in use can be an issue, these techniques provide a powerful and flexible interface for manipulating 2D GUIs.

A different family of techniques uses static or dynamic adjustment of the C-D ratio. This concept has been applied to many different VE-related tasks including navigation [23], perception [24], selection [15] and manipulation [12, 16]. Blanch *et at.* [15] improve selection through *semantic pointing*. Semantic pointing is based on defining two independent sizes for each potential target presented to the user. One size is used in *visual space* and it is adapted to the amount of information conveyed by the object. The other size is used in *motor space* and is adapted to the object's importance for the manipulation. This decoupling between visual and motor size is achieved by changing the C-D ratio according to cursor distance to nearby targets.

Some techniques amplify rotations so that most manipulations can be accomplished with a single motion, thus minimizing the need for releasing the virtual object and grabbing it again [19, 25, 12]. Usability properties of different rotation mappings are discussed in [25, 12] and different C-D gains between real and virtual hands are evaluated in [20]. Poupyrev et al. [26] derive the equations for linear and non-linear amplifications of spatial rotations. Non-isomorphic rotational mappings for enhanced 3D rotations of objects have been explored in [12]. The authors provide several guidelines to design non-isomorphic techniques for rotating objects. In particular, they identify two *compliances* between the user movements and the sensory feedback which affect the effectiveness of 3D rotation. *Directional compliance* means that the object rotates in the same direction as the input device. Directional compliance ensures correspondence between the visual, kinesthetic and proprioceptive feedbacks of motor movement. Note that absolute, non-isomorphic mappings do not always maintain directional compliance on 3D rotations [12]. On the other hand, *nulling compliance* ensures that rotating the device into an initial orientation would also rotate the controlled virtual object into a zero orientation. Our proposed techniques maintains both compliances. Note however that 3D object rotation is a problem significantly different from the problem being addressed in this paper, i.e. controlling a ray for moving a 3D cursor over a virtual window. The above techniques put the emphasis on amplifying rotations to allow a more effective use of limited tracking ranges and avoid the *clutching* problem [12].

The PRISM (Precise and Rapid Interaction through Scaled Manipulation) [16] uses C-D adjustment to provide fast and accurate manipulation of 3D objects. PRISM uses the hand speed of the user to gradually switch between different modes by altering the C-D ratio. The mapping is controlled basically by three speed constants defining four intervals. In the first interval, the speed is below a certain minimum velocity and the movement is considered jitter, so the C-D ratio is set to ∞. The second interval corresponds to scaled motion, where C-D ratio is inversely proportional to the hand speed. Above the second speed constant (about 20 degrees/second for ray-casting), the C-D ratio is set to 1. Motion above the third speed constant is used only for offset recovery. The first version of PRISM operated only on translation [16] but it has been recently extended to 3D rotations [17]. The authors report significant user performance improvements when using ray casting with PRISM rotations. Although similar to PRISM in concept, our technique adopts completely different strategies for activation/deactivation, computing the C-D ratio and providing integrated visual feedback alleviating the anisomorphism of the movement. A de-



(a) (b) (c)

Figure 2: Elements involved in the computation of the CD-ratio: Device coordinate system and device ray *(a)*, spherical coordinates used at activation time to fix the CD-ratio *(b)*, and spherical coordinates used for computing the selection ray direction during scaled mode *(c)*.

tailed comparison with PRISM is provided in Section 3.5.

3 Our approach

3.1 Overview

The *friction surfaces* technique has been conceived 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 ray-casting technique which adapts the C-D ratio according to the size and position of the virtual window.

We start by introducing some notation that will be used in the rest of the paper. 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 2-a, with the negative Z axis defining the user's hand pointing direction. This pointing direction will be referred to as the *device ray*. The device ray is isomorphically controlled by the hand and plays the role of the control component in the C-D ratio. The *Zero orientation* is an orthonormal reference basis used for scaling down the rotation of the user's hand while in scaled mode. The zero orientation is defined from DCS and the window's center when the scaled mode is activated (discussed in Section 3.2) and then remains unchanged until the mode is deactivated. We call *selection ray* the ray resulting from scaling down the rotation of the hand with respect to the zero frame. As its name suggests, the intersection of the selection ray with the virtual window's plane is used as the cursor for selection and manipulation purposes. In our approach, both the selection ray and the device ray originate at the current device position, as we only scale down rotations, not translations. The selection ray's direction is the result of applying an isotropic linear mapping to the rotation measured from the zero orientation. Finally, the *feedback ray* is the curved line segment that will be displayed for providing visual feedback about the linear mapping (the device and selection rays are not rendered). Note that in the classic ray-casting manipulation, the selection ray coincides with the device ray, thus providing isomorphic manipulation. In our case, isomorphism is preserved only when no virtual window is active.

3.2 Activation

We use two distinct modes: one which scales hand rotations when accuracy is needed (scaled mode) and one which provides direct, isomorphic interaction (normal mode). We now describe different strategies for activating the scaled mode and discuss the associated computations. We have considered both manual and automatic activation. Manual activation/deactivation requires the user to issue a trigger event (e.g. a button press). Automatic activation simply takes place whenever the selection ray enters a virtual window. The mode

is set back to normal mode when the selection ray leaves the window; that happens when the hand rotation with respect to the zero orientation reaches a certain maximum value (e.g. 45 degrees). The deactivation causes the selection ray to coincide again with the device ray. We have found that this lost of continuity is not disturbing to the users because it simply causes a curvature change in the bent ray used for feedback (see Section 3.4).

When the scaled mode is activated, the zero orientation is defined. In our implementation we have defined this reference frame by forcing the Z-axis to be oriented along the segment joining the window center and the device position (this segment is shown in Figure 2-b). 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 $\vec{u}$ be 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 $\vec{y} = \vec{z} \times \vec{x}$. The zero orientation is then defined by the unit vectors $\vec{x}$, $\vec{y}$, $\vec{z}$.

We then compute the range of directions the selection ray can travel before leaving the active window as follows. 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 1 and 2, where (x, y, z) 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:

$$\theta_i = \tan^{-1}\left(\frac{-x}{-z}\right) \tag{1}$$

$$\phi_i = \sin^{-1}\left(\frac{y}{\sqrt{x^2+y^2+z^2}}\right) \tag{2}$$

We can compute the maximum rotation angles of the selection ray in each direction as $\theta_{max} = \max_i\{|\theta_i|\}$ and $\phi_{max} = \max_i\{|\phi_i|\}$. Since we want to use an isotropic scale on both directions, we just use the maximum of these two angles. Therefore, the C-D ratio r is computed as

$$r = \frac{\psi}{\max(\theta_{max}, \phi_{max})}, \tag{3}$$

where ψ is a user-defined constant that defines the range of directions of the input device that approximately map onto a selection ray within the virtual window. In the user studies described in next section we used $\psi = \pi/4$, thus providing the user with a 90 degrees arc for interacting with the active virtual window. In our implementation, ψ can be adjusted by the user and the arc swept by $-\psi, \psi$ is rendered just below the user's hand position (see Figure 10). If $r < 1$, the window is sufficiently close to the user and thus the scaled mode is not activated.

When the scaled mode is deactivated, 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. This delay has to be applied only when the selection ray enters another window immediately after deactivation.

3.3 Computation of the selection ray

In scaled mode, the selection ray is computed using the C-D 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, ϕ_s of a target point T by scaling down the device rotation measured from the zero orientation, $\theta_s = \theta_d/r$, $\phi_s = \phi_d/r$ (see Figure 2-c). The coordinates of T require a simple conversion back to cartesian coordinates,

$$x = -\rho\sin(\theta_s)\cos(\phi_s) \tag{4}$$

$$y = \rho\sin(\phi_s) \tag{5}$$

$$z = -\rho\cos(\theta_s)\cos(\phi_s) \tag{6}$$

where $\rho > 0$ is an arbitrary value.

The resulting selection ray passes through the current device position and the computed point T.

3.4 Feedback

We have considered two distinct options for providing 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). 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.

Curved line segments have been proposed for different purposes related with ray-casting selection. IntenSelect [27] uses dynamic rating of the objects falling inside a conic selection volume to bend the selection ray so that it snaps with the highest ranking object. Flexible pointers are used in [28] to point more easily to fully or partially obscured objects using two-handed interaction.

We draw the curved line segment using a Bézier spline (see Figure 1-b). 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 friction surface defined by the window. Visual feedback is completed by drawing a cross-shaped cursor in the intersection point P.

3.5 Comparison with previous techniques

As stated below, friction surfaces is similar to the PRISM [17] in concept. Both techniques adapt the C-D ratio to provide accurate manipulation of distant objects. We now synthesize the main differences between both approaches:

- In PRISM the CD-ratio depends on the speed of user's movements; in our case, the CD-ratio depends on the size and position of the window with respect to the user, and it remains constant while in scaled mode. Therefore our adjustment of the control-display ratio is static, unlike the dynamic adjustment used in PRISM.
- In PRISM the activation of the scaled mode depends on several speed constants; in our case activation takes place when the ray enters a virtual window.
- Our technique provides integrated feedback of both the device and the selection ray through a curved line segment.
- Our technique does maintain nulling compliance [12]. Nulling compliance preserves the consistent correspondence between the origins of the coordinate systems in physical and virtual spaces. This consistency is particularly important when the input device's orientation is easily perceived by the user through proprioception (a common situation in devices typically used for pointing, such as the wand). 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 [17].

Besides these aspects, an important difference is that our technique 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.

4 Evaluation

We conducted a usability evaluation to measure the effectiveness of Friction Surfaces (FS) compared with classic, isomorphic ray-casting (RC) and ray-casting with PRISM rotation (PRISM) [17]. 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. Before each experiment users were provided with a short training session which required them to complete practice trials using both interaction techniques.

4.1 Evaluation test

The evaluation test has been designed to evaluate the task performance in terms of time-to-complete

a given task and maximum accuracy achieved in a fixed period of time. All the experiments were conducted on a four-sided CAVE with a 6-DOF wanda [29] and a Polhemus Fastrak tracking system with 2 receivers providing 60 updates/s with 4 ms latency. Users can use the wanda's analog joystick to adjust ψ and thus the C-D ratio. 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. The scaled mode was activated automatically each time the device ray entered a virtual window.

The dialogs used in the experiments are shown in Figure 3. 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 basically combo boxes and selection lists. The third dialog is designed also 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, so users can be more focused at purely interaction tasks.

For the first three dialogs, users were requested to complete the involved tasks as quickly as possible, using isomorphic ray-casting, friction surfaces and ray-casting with PRISM rotation in a random order. 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. All users were requested to complete 1-2 trials using the three techniques in a random order.

4.2 Experimental results

Figure 4 shows the completion time for each task. Tasks 1-4 correspond to the dialogs (a)-(d) shown in Figure 3. Note that on average the completion time is lower with our technique. We performed a correlated samples one-way ANOVA on the data with completion time as the dependent variable and the interaction technique (RC, PRISM or FS) as the independent variable. Tukey pair-wise HSD tests were performed when significant differences were found.

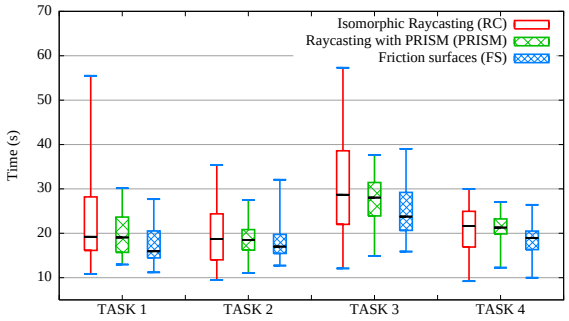


Figure 4: Time to complete the tasks involved in the four dialogs.

ANOVA results for completion times are shown in Table 1. FS was found to be significantly faster than RC on tasks 1 and 3, whereas PRISM difference with respect to RC was nonsignificant. Time differences on Task 2 (involving large targets) and Task 4 (where interaction with the sliders had a strict time budget) were nonsignificant.

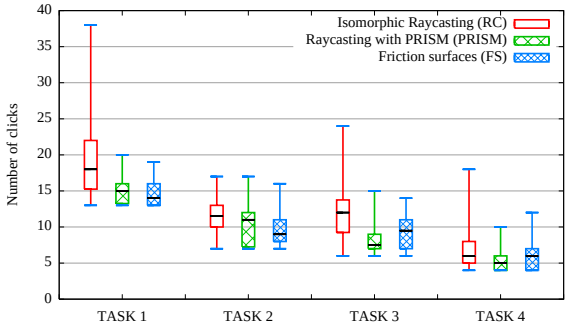
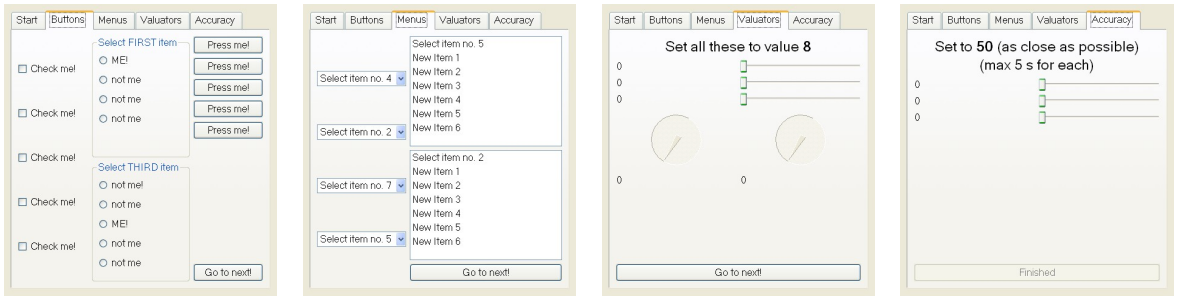


Figure 5: Number of button clicks performed during each task.

The second performance metric we measured is the number of times the user pressed the wanda button. Figure 5 shows the button clicks for each task. Note that in tasks emphasizing on 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). We performed a correlated samples one-way ANOVA on the data with button clicks as the dependent variable and the interaction technique (RC, PRISM or FS) as the independent vari-



(a) (b) (c) (d)

Figure 3: The test dialogs used in the experiments.

Source	SS	DoF	MS	F	P	HSD test	
Technique	443.74	2	221.87	4.75	0.013	RC vs PRISM	nonsignificant
Error	2242.79	48	46.72			RC vs FS	$P < 0.01$
Subjects	1431.68	24				PRISM vs FS	nonsignificant
Total	4118.21	74					

Completion time for task 1

Source	SS	DoF	MS	F	P	HSD test	
Technique	28.64	2	14.32	0.58	0.55	RC vs PRISM	n/a
Error	1170.80	48	24.39			RC vs FS	n/a
Subjects	869.93	24				PRISM vs FS	n/a
Total	2069.38	74					

Completion time for task 2

Source	SS	DoF	MS	F	P	HSD test	
Technique	361.82	2	180.91	3.70	0.031	RC vs PRISM	nonsignificant
Error	2346.24	48	48.88			RC vs FS	$P < 0.05$
Subjects	2960.75	24				PRISM vs FS	nonsignificant
Total	5668.82	74					

Completion time for task 3

Source	SS	DoF	MS	F	P	HSD test	
Technique	71.51	2	35.75	3.28	0.045	RC vs PRISM	nonsignificant
Error	521.87	48	10.87			RC vs FS	nonsignificant
Subjects	948.10	24				PRISM vs FS	nonsignificant
Total	1541.49	74					

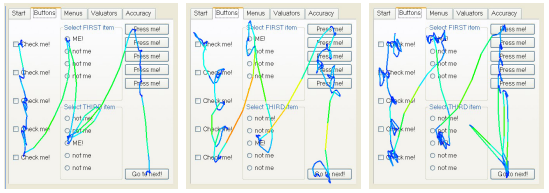
Completion time for task 4

Source	SS	DoF	MS	F	P	HSD test	
Technique	2727.72	2	1363.86	4.62	0.014	RC vs PRISM	nonsignificant
Error	14166.34	48	295.13			RC vs FS	$P < 0.05$
Subjects	17126.36	24				PRISM vs FS	nonsignificant
Total	34020.43	74					

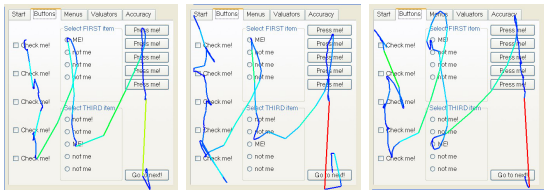
Total completion time

Table 1: ANOVA tables for completion times

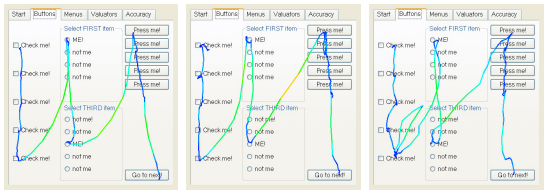
able. ANOVA results are shown in Table 2. For most tasks, FS and PRISM were significantly better than RC, with nonsignificant differences between PRISM and FS. Regarding the number of mistakes and the Heisenberg effect [30], it should be noted that PRISM incorporates a noise filter. Any motion below 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.



(a) (b) (c)



(d) (e) (f)



(g) (h) (i)

Figure 6: 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.

We also recorded the path traced by the 3D cursor over the virtual window. Figure 6 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, which is reflected by the loops around the small targets in Figure 6a-c. This contrasts with the smoother paths produced by PRISM and FS (Figure 6d-i).

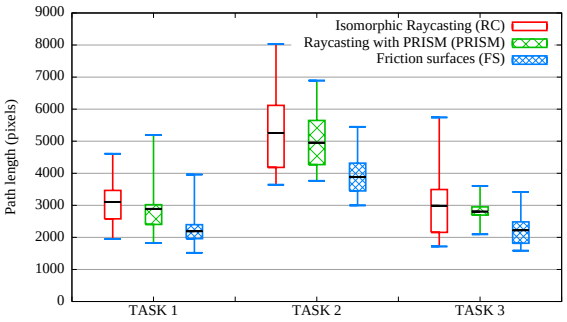


Figure 7: Length of the path traced by the cursor.

The length of the paths traced by the cursor, in native window pixel units, are shown in Figure 7. Figure 8 shows the number of times the user had to rectify the movement, measured as the number of times two consecutive edges of the path define an angle greater than 90 degrees. ANOVA results are shown in Table 3. Both PRISM and FS were found to produce less turns than RC. Regarding path lengths, FS lead to cursor paths significantly shorter than both RC and PRISM.

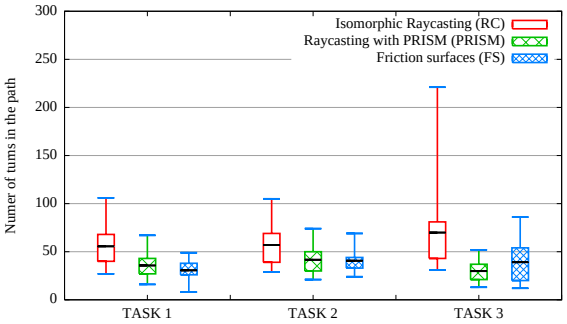


Figure 8: Number of turns in the path traced by the cursor.

Figure 9 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 3-d). Each slider had increasing ranges and thus increas-

Source	SS	DoF	MS	F	P	HSD test	
Technique	361.70	2	180.85	12.41	$< .0001$	RC vs PRISM	$P < 0.01$
Error	698.96	48	14.56			RC vs FS	$P < 0.01$
Subjects	350.88	24				PRISM vs FS	nonsignificant
Total	1411.54	74					

Button clicks for task 1

Source	SS	DoF	MS	F	P	HSD test	
Technique	39.12	2	19.56	3.10	0.053	RC vs PRISM	n/a
Error	302.21	48	6.29			RC vs FS	n/a
Subjects	260.98	24				PRISM vs FS	n/a
Total	602.32	74					

Button clicks for task 2

Source	SS	DoF	MS	F	P	HSD test	
Technique	275.70	2	137.85	13.62	$< .0001$	RC vs PRISM	$P < 0.01$
Error	485.62	48	10.11			RC vs FS	$P < 0.01$
Subjects	343.65	24				PRISM vs FS	nonsignificant
Total	1104.98	74					

Button clicks for task 3

Source	SS	DoF	MS	F	P	HSD test	
Technique	46.58	2	23.29	5.53	0.006	RC vs PRISM	$P < 0.01$
Error	202.08	48	4.21			RC vs FS	nonsignificant
Subjects	166.48	24				PRISM vs FS	nonsignificant
Total	415.14	74					

Button clicks for task 4

Source	SS	DoF	MS	F	P	HSD test	
Technique	2187.92	2	1093.96	17.68	$< .0001$	RC vs PRISM	$P < 0.01$
Error	2968.74	48	61.84			RC vs FS	$P < 0.01$
Subjects	1770.21	24				PRISM vs FS	nonsignificant
Total	6926.88	74					

Total number of clicks

Table 2: ANOVA tables for button clicks

Source	SS	DoF	MS	F	P	HSD test	
Technique	128446712	2	64223356	28.4582	$< .0001$	RC vs PRISM	nonsignificant
Error	108324442	48	2256759			RC vs FS	$P < 0.01$
Subjects	87964318	24				PRISM vs FS	$P < 0.01$
Total	324735474	74					

Path length

Source	SS	DoF	MS	F	P	HSD test	
Technique	90219.42	2	45109.71	31.41	$< .0001$	RC vs PRISM	$P < 0.01$
Error	68933.80	48	1436.12			RC vs FS	$P < 0.01$
Subjects	88522.27	24				PRISM vs FS	nonsignificant
Total	247675.50	74					

Number of turns in the path

Table 3: ANOVA tables for path length and number of turns.

ing levels of difficulty. Both PRISM and FS performed much better than RC, with nonsignificant differences between PRISM and FS.

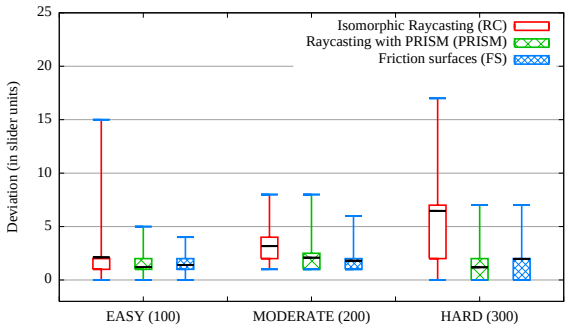


Figure 9: Deviation from the target value on the accuracy test. The integer value between parentheses is the slider's range.

4.3 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 our technique. 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, nicknamed the Heisenberg effect [30]. 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 the anisomorphic raycasting were a bit unnatural compared with isomorphic raycasting, although their performance was better with anisomorphic techniques.

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.

5 Concluding remarks and future work

The accommodation of conventional 2D GUIs with Virtual Environments (VEs) can greatly enhance the possibilities of many VE applications. In this paper we have presented a variation of the well-known ray-casting technique for accurate selection of 2D widgets over a virtual window immersed into a 3D world. A user study on a four-sided CAVE indicates that the proposed technique outperforms significantly isomorphic ray-casting in several performance metrics including completion time, number of mistakes and manipulation accuracy. We have also included a comparison with PRISM ray-casting. Both techniques perform significantly better than classic ray-casting, with little performance differences between them. PRISM seems to perform better than Friction Surfaces when extreme accuracy is required (e.g. adjusting a slider with pixel accuracy) whereas our technique is particularly suitable for fast selection of small targets. Friction Surfaces maintains both directional and nulling compliances [12] as it simulates approximately the interaction with a large spherical window (Figure 10). Note that rendering this simulated window would be impractical as it would occlude most of the scene.

An important issue is how these techniques compare with isomorphic ray-casting when the size and density of targets is an independent factor. We plan to conduct this evaluation as part of the future work. We also plan to integrate friction surfaces with dynamic rating techniques [27] so that the displayed ray is further bent to snap with the highest ranking object. This would further improve object selection, with little or no effect on object manipulation.

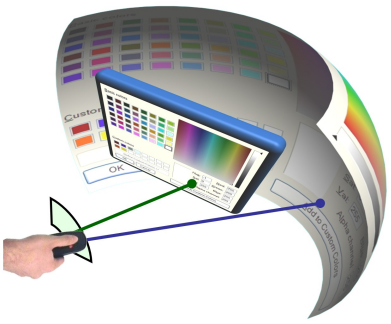


Figure 10: Manipulation with Friction Surfaces simulates approximately the interaction with a large spherical window.

6 Acknowledgments

The authors would like to thank all the participants who took part in this experiment for their kindness and patience. This work has been funded by the Spanish Ministry of Science and Technology under grant TIN2004-08065-C02-01.

References

[1] K. Watsen, R. Darken, and M. Capps. A handheld computer as an interaction device to a virtual environment. In *Proceedings of the International Projection Technologies Workshop*, pages 303–304, 1999.

[2] Matthias Bues, Roland Blach, and Frank Haselberger. Sensing surfaces: bringing the desktop into virtual environments. In *Proc. of the 9th Eurographics Workshop on Virtual environments EGVE'03*, pages 303–304, 2003.

[3] Stephen DiVerdi, Daniel Nurmi, and Tobias Hollerer. A framework for generic inter-application interaction for 3D AR environments. In *Augmented Reality Toolkit Workshop, 2003*, pages 86 – 93, 2003.

[4] Niklas Elmqvist. 3dwm: A platform for research and development of three-dimensional user interfaces. Technical Report CS:2003-04, Chalmers Dept. of Computing Science, 2003.

[5] C. Andujar, M. Fairen, and F. Argelaguet. A cost-effective approach for developing application-control guis for virtual environments. In *Proc. of 1st IEEE Symposium on 3D User Interfaces, 3DUI'06*, 2006.

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

[7] Doug A. Bowman and Larry F. Hodges. Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments. *Journal of Visual Languages and Computing*, 10(1):37–53, 1999.

[8] Jeffrey S. Pierce, Brian C. Stearns, and Randy Pausch. Voodoo dolls: seamless interaction at multiple scales in virtual environments. In *Proceedings of the 1999 Symposium on Interactive 3D graphics*, pages 141–145, 1999.

[9] Mark Mine, Jr. Frederick Brooks, and Carlo Sequin. Moving objects in space: exploiting proprioception in virtual-environment interaction. In *SIGGRAPH'97*, pages 19–26, 1997.

[10] A Dachselt, R.; Hübner. A survey and taxonomy of 3d menu techniques. In *Proceedings of the 12th Eurographics Symposium on Virtual Environments (EGVE'06)*, 2006.

[11] Doug A. Bowman, Ernst Kruijff, Joseph J. LaViola, and Ivan Poupyrev. *3D User Interfaces : Theory and Practice*. Addison Wesley, 2004. ISBN: 0-2017-5867-9.

[12] Ivan Poupyrev, Suzanne Weghorst, and Sidney Fels. Non-isomorphic 3D rotational techniques. In *Proceedings of the 2000 ACM conference on Human factors in computing systems (CHI 2000)*, pages 540–547, 2000.

[13] Ivan Poupyrev, Suzanne Weghorst, Mark Billinghurst, and Tadao Ichikawa. Egocentric object manipulation in virtual environments: Evaluation of interaction techniques. *Computers Graphics Forum*, 17(3):41–52, 1998.

[14] Robert Lindeman, John Sibert, and James Hahn. Towards usable VR: an empirical study of user interfaces for immersive virtual environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 64–71, 1999.

[15] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 519–526, 2004.

[16] Scott Frees and Drew Kessler. Precise and rapid interaction through scaled manipulation in immersive virtual environments. In *Proc. of IEEE Virtual Reality 2005*, pages 99–106, 2005.

[17] Scott Frees, Drew Kessler, and Edwin Kay. Prism interaction for enhancing control in immersive virtual environments. Technical Report LU-CSE-05-013, Lehigh University, Computer Science and Engineering Dept., 2005.

[18] Richard Stoakley, Matthew Conway, and Randy Pausch. Virtual reality on a WIM: Interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272, 1995.

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

[20] Doug Bowman and Larry Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *I3D '97: Proceedings of the 1997 ACM Symposium on Interactive 3D graphics*, pages 35–38., 1997.

[21] 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*, pages 234–241, 1997.

[22] Dieter Schmalstieg, Miguel Encarnacao, and Zsolt Szalavri. Using transparent props for interaction with the virtual table. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 147–153, 1999.

[23] Desney Tan, George Robertson, and Mary Czerwinski. Exploring 3d navigation: combining speed-coupled flying with orbiting. In *Proc. of the SIGCHI conference on Human factors in computing systems*, pages 418–425, 2001.

[24] L. Dominjon, A. Lecuyer, J.M. Burkhardt, P. Richard, and S. Richir. Influence of control/display ratio on the perception of mass of manipulated objects in virtual environments. In *Proceedings of IEEE International Conference on Virtual Reality (IEEE VR)*, 2005.

[25] Ken Hinckley, Joe Tullio, Randy Pausch, Dennis Proffitt, and Neal Kassell. Usability analysis of 3d rotation techniques. In *UIST '97: Proc. of the ACM symposium on User interface software and technology*, pages 1–10, 1997.

[26] Ivan Poupyrev, Suzanne Weghorst, Takahiro Otsuka, and Tadao Ichikawa. Amplifying spatial rotations in 3d interfaces. In *ACM Conference on Human Factors in Computing Systems, CHI'99*, pages 256–257, 1999.

[27] G. de Haan, M. Koutek, and F.H. Post. IntenSelect: Using Dynamic Object Rating for Assisting 3D Object Selection. In Erik Kjems and Roland Blach, editors, *Proceedings of the 9th IPT and 11th Eurographics VE Workshop (EGVE) '05*, pages 201–209, 2005.

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

[29] Ascension Technology Corporation. Wanda navigation device. http://www.ascension-tech.com/products/wanda.php.

[30] D. Bowman, C. Wingrave, J. Campbell, and V. Ly. Using pinch gloves for both natural and abstract interaction techniques in virtual environments. In *HCI International*, pages 629–633, 2001.