Eurographics Symposium on Virtual Environments (2006)
Roger Hubbold and Ming Lin (Editors)

Hybrid Interfaces in VEs: Intent and Interaction

Gerwin de Haan, Eric J. Griffith, Michal Koutek and Frits H. Post

Data Visualization Group, http://visualisation.tudelft.nl
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, The Netherlands

Abstract

Hybrid user interfaces (UIs) integrate well-known 2D user interface elements into the 3D virtual environment, and provide a familiar and portable interface across a variety of VR systems. However, their usability is often reduced by accuracy and speed, caused by inaccuracies in tracking and a lack of constraints and feedback. To ease these difficulties often large widgets and bulky interface elements must be used, which, at the same time, limit the size of the 3D workspace and restrict the space where other supplemental 2D information can be displayed. In this paper, we present two developments addressing this problem: supportive user interaction and a new implementation of a hybrid interface. First, we describe a small set of tightly integrated 2D windows we developed with the goal of providing increased flexibility in the UI and reducing UI clutter. Next we present extensions to our supportive selection technique, IntenSelect. To better cope with a variety of VR and UI tasks, we extended the selection assistance technique to include direct selection, spring-based manipulation, and specialized snapping behavior. Finally, we relate how the effective integration of these two developments eases some of the UI restrictions and produces a more comfortable VR experience.

Categories and Subject Descriptors (according to ACM CCS): H 5.2 [User Interfaces]: Interaction Styles; I 3.6 [Methodology and Techniques]: Interaction Techniques; I.3.7 [Three-Dimensional Graphics and Realism]: Virtual Reality

1. Introduction and Motivation

Poor user interfaces (UIs) in virtual environments have long been cited as one of the major factors preventing widespread acceptance of Virtual Reality. A significant amount of research has been devoted to developing new and improved VR UIs, but a consensus has not yet been reached on what a good VR UI is. One trend has been to focus on using so-called hybrid UIs, which incorporate well-known 2D user interface elements into the 3D environment, rather than developing entirely 3D interfaces using new metaphors. These interfaces benefit from the familiarity and their relative portability across a variety of VR systems. However, tracking inaccuracies and limitations on rendering text in VR generally necessitate large widgets. Large widgets make for bulky user interfaces, which can occlude objects in the scene. They also limit the size of the 3D workspace and restrict where other supplemental 2D information can be displayed.

We primarily employ Virtual Reality as a means for enhancing the interactive scientific visualization and data exploration process. Our applications focus on the visualization and control of (real-time) simulations of physical processes, and Cloud Explorer [GPK*05], which is illustrated in Figure 1, is currently our main application of interest. The goal of this application is to facilitate cumulus cloud life-cycle studies. Large data sets result from atmospheric simulations, from which various information modalities and features are extracted in an off-line preprocessing phase. The resulting data can be interactively explored in a virtual environment, such as our Virtual Workbench setup, which is equipped with tracked glasses, stylus and a transparent acrylic hand-held panel called the PlexiPad.

During the course of developing this and other applications, we increasingly came across situations that warranted 2D input, 2D output, or both. Our existing UI solutions, namely primitive buttons, sliders, and graphs, were not able to meet all of our requirements. They were promising, but they suffered from the common problems of being overly large and inflexible. In addition, they often remained frus-

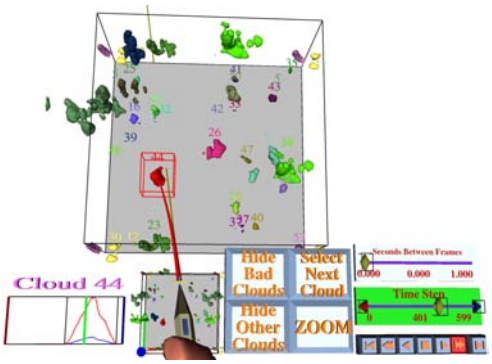

Figure 1: *Overview of the original Cloud Explorer, our application for cumulus cloud life-cycle studies*

trating to interact with, even after improving tracker registration and the use of passive haptic feedback on the Workbench surface or the PlexiPad. To address these issues, we developed two strategies in parallel: an improved interaction technique and an improved hybrid interface, which we then successfully integrated.

The remainder of this paper is organized as follows: first we describe some previous work in the field of hybrid interfaces and interaction assistance. In Section 3 we describe our work on the hybrid interfaces, followed by the extensions of the interaction technique in Section 4. Finally, we describe the combination of these developments and discuss our results and future work.

2. Related work

Windows and window-like constructs have appeared in VEs for many years, and more examples appear in the literature than are listed here. One early example is from Fisher et al. [FMHR86], where information windows and reconfigurable control panels are briefly mentioned. Other early work focused on importing X Windows into VEs through the use of bitmaps [FMHS93], textured polygons [Dyk94], and specialized widget toolkits [AS95a, AS95b]. This approach, however, favors the use of a suite of existing 2D applications, which need only minimal communication with the VE application, e.g. receiving mouse events. Another popular window metaphor has been the hand-held window. These have been implemented for augmented reality [SG97], head-mounted displays [AS95a, BHB98, LSH99a], workbenches [SES99,dHKP02], and using a 3Com PalmPilot in a CAVE-like environment [WDC99]. This approach is tightly integrated into the VE, and typically uses a pen-and-tablet metaphor for user input. The window is fixed to the panel, though, and it limits the user to one active window at a time. Cuppens et al. [CRC04] and Larimer and Bowman [LB03] have made two recent attempts at more complete 2D UI solutions for VEs. Cuppens et al. specify their UI via XML and provide interaction via a PHANToM device for a pen-and-tablet metaphor. They currently limit the UI to menus, toolbars, and dialogs. Larimer and Bowman focus on CAVE-like environments by placing their windows tangent to a user-centered sphere. Their windowing library builds on an existing GUI toolkit, and it relies on ray-casting for interaction.

Various approaches are used to alleviate the difficulties in interaction with (small/2D) interface elements in a VE. The use of (passive) haptic feedback and constraints limits the users actions to match 3D location of the elements. The use of physical surfaces such as the projection screen of a workbench or a tracked handheld panel [SG97] are examples of placeholders which provide passive feedback. User tests by Lindeman et al. [LSH99b] indicate that the addition of passive-haptic feedback for use in precise UI manipulation tasks can significantly increase user performance, and that users prefer this type of operation. In [FK05, Osa05] the Control-Display ratio of the interaction is changed dynamically, and user's movements are scaled down to support small and precise interaction. As this approach affects the co-location of the interaction device in the virtual environments this solution is mainly limited to non see-through HMDs.

Selection-by-volume techniques are used for selecting small objects, which are difficult to select using regular direct- or ray-based selection [Dan05, LG94]. There are, in general, two approaches for singling out an object between many (cluttered) objects in a selection volume, which is usually the case in grouped user interface elements. The distinction can be made explicitly by the user, for example by using a menu [Dan05] or a selection sweep to single out one item [SP04]. These require a switch of interaction mode (or extra buttons), which is not desirable for menu or widget operation. The other approach is the use of scoring metrics to automate the determination of the best scoring object [SRH05, LG94]. As these metrics are calculated on a per-frame basis, they are hampered by the same tracking inaccuracies and hand jitter that make regular interaction techniques so difficult. In the IntenSelect technique [dHKP05] these per-frame scores are accumulated over time for each object. As a result a dynamic, time-dependent object ranking with history is constructed. The highest ranking object is indicated by snapping the bending selection ray to it.

3. Windows and Widgets

Windows in virtual environments can serve as tools for both input and output. Dialog boxes and tool bars are common examples of input windows, and they have been implemented in VE applications such as [CRF97,vTRvdM97]. Two examples of output windows in VEs include a map of an airplane interior [AS95a] and a simple clock [LB03]. 2D input functionality is often used for system control, such as enabling or disabling modes in the VE, and symbolic input, such as entering numbers. Virtual environments incorporating 2D in-

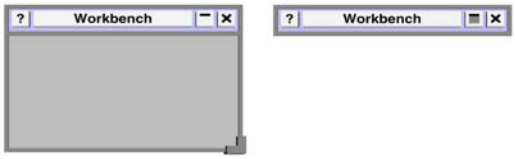

Figure 2: *A window, left, with (from left to right) information, title, windowshade, and hide widgets in the titlebar and a sizing widget in the lower right-hand corner. On the right, the same window is pictured in windowshade mode.*

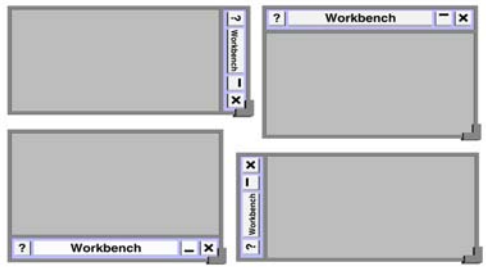

Figure 3: *The four possible title bar locations.*

put elements are termed hybrid interfaces. 2D output, on the other hand, is used to provide additional relevant information about the VE. If done properly, this helps users learn more about the VE [BWH99]. Virtual environments with such information are called information rich.

When developing our hybrid interface, we tried to meet several goals. We want our interface to be tightly integrated into the VE to facilitate bi-directional communication, thus eliminating the barrier between "information rich" and "hybrid interface". Our primary VR setup is a Virtual Workbench so our solution must work well with it. The UI elements should not be unreasonably large because are intended to be supplemental rather than the primary focus. The interface should be intuitive to use. There should not be a limit on the number of windows visible, and the user should have control over where and which ones are visible. The windows and widgets should be natively rendered in 3D for aesthetics, readability, and to take advantage of the suggested increase in accuracy provided [LST01].

In the remainder of this section, we present the specifics of our implementation. We describe the windows, the modifications to the existing widgets, and the relevant motivations behind the decisions we made.

3.1. Windows

Figure 2 illustrates the basic window we came up with. It features a title bar with familiar widgets, a body area, and a sizing widget. The window is rendered as geometry with the window body being flat, while the border and widgets are 3D. This gives better visual cues when the button widgets are pressed, and it allows for direct selection of the widgets by placing the stylus tip inside of them. The window has its own 3D coordinate system to allow for 3D content to be attached to it or placed on it, and it takes advantage of the stencil buffer to allow 2D content to be overlayed onto it. As the window is resized, the components must be moved and resized rather than rescaled to avoid awkward stretching effects.

We designed the windows to be fixed to a particular plane in space. Within the plane, they can be moved and resized, depending on the window, as if it were on a 2D desktop. Translation perpendicular to the plane or rotation is disallowed to preserve the likeness to traditional 2D interfaces. For our purposes, we fix windows onto two planes: the surface of the Workbench and the plane of our PlexiPad. The Workbench surface has the advantage of being the focus plane while the PlexiPad can be moved out of sight by the user when it is not needed. Furthermore, they both provide optional passive haptic feedback, which has been shown to be both intuitive to learn [BHB98] and to speed certain 2D tasks [LSH99a]. In other systems, other planes may make more sense, such as the walls in a CAVE-like or Power Wall system.

The window in Figure 2 has a title bar, four title bar widgets, and a sizing widget. However, they are all optional, and it need not have any of these widgets if they are not necessary. The sizing widget, as the name suggests, is used to resize the window. The remaining title bar widgets are the information widget, the title widget, the widget, and the hide widget. The information widget displays extra information about the window. The title widget shows a caption for the window, can be used to move the window around, or both. The windowshade widget hides the body of the window so only the title bar is visible, and then shows the body again if used twice (see Figure 2). The hide widget hides the window.

We have made it possible to place the title bar on any side of the window. See Figure 3. This has two advantages. First, in systems such as the Workbench, the user is closer to the bottom of the window. With the title bar at the bottom, it is not obscured by any potential 3D content in the window, and it is closer at hand if the user wishes to move the window. Secondly, when the window is in windowshade mode, the title bar determines where the window "pops out". This allows for windows to be arrayed around the edges of the Workbench or the PlexiPad in a convenient manner, and only restored when their functionality is needed.

3.2. Widgets

Our windows make use of both our set of existing 3D widgets, and a new set of widgets that are more window aware. We have also extended some of our existing widgets to take

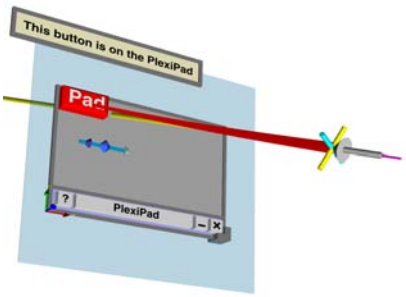

Figure 4: *A window with a button, a slider, a visible tooltip, and the standard window widgets.*

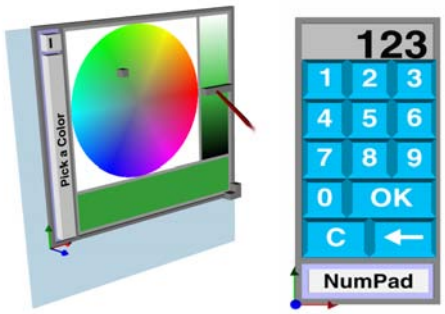

Figure 5: *Dialogs: Color picker (left) and numeric entry (right).*

advantage of the windows. Figure 4 illustrates a window with a button from our old widget set, a slider from the new widget set, and a tooltip. To use the existing widgets, a developer need only place them in the window's coordinate system at an appropriate location. The new widgets may only be placed on windows or in their title bars because they occasionally inform the window about their state.

One of the important new widgets is the tooltip. We have included it because it helps address the problem of VE clutter. Text must be reasonably large to be legible in a virtual environment, and it is often a challenge to think of a descriptive caption for menu items or buttons that is not too long. At the same time, experienced users will not have much need for overly descriptive captions as they are already familiar with the commands available to them. With tooltips, smaller buttons, sometimes with an icon instead of text, can be used thereby saving both space and not leaving the novice user stranded. Tooltips can be easily associated with all of the new widgets, and the older widgets have been extended to provide them as well, if deemed necessary. Figure 4 illustrates the tooltip for a button being shown.

3.3. Dialogs

In 2D UIs, dialogs or dialog boxes are commonly used to request input from the user. A familiar example is the dialog box to open a file. Dialog boxes have already found a home in virtual environments as well. Both [LB03] and [CRC04] incorporate dialog boxes for the purpose of influencing the environment. We have constructed two simple dialog boxes, which are illustrated in Figure 5.

The first is a simple color picking dialog box. It uses the HSL color model to allow the user to interactively select a color. She does so by separately moving a widget to pick the hue and saturation and a widget to pick the lightness. As she manipulates the widgets, they are constrained to the appropriate places on the window, and the rectangle at the bottom of the window updates to show the currently selected color. We draw the color wheel and lightness bar with fragment shaders to give the user the freedom to make the dialog as large or as small as she prefers.

The second dialog is a simple numeric entry dialog. When placed on a device such as the PlexiPad, the user has ready access to it only at those times he feels he needs it. In Cloud Explorer, this dialog is used to allow the user to select a particular cloud by its identifying number.

3.4. The Graph Window

The new graph window from Cloud Explorer (Figure 6) is an example of a window that "puts it all together". Whenever a cloud in the data set is selected, a graph window is shown with information about that cloud. The window has many of the normal widgets, but it also has two extra title bar widgets. These widgets add or remove graphs from the window. A number of plots can be selected to display in each graph. As Cloud Explorer runs, the user browses through time in the data set, and the slider updates to reflect the current simulation time, as well as the limits of the playback. Furthermore, the user can directly adjust both limits and the current time step from the graph window.

4. Supporting Interaction in VR

Results of our previous user test indicated that selection-by-volume techniques and IntenSelect can improve object selection performance, especially in difficult situations where objects were small, moving and cluttered [dHKP05]. Although the selection technique was originally designed with the dynamic nature of data objects such as used in Cloud Explorer in mind, some of its properties proved to be very useful in the fast control of small and cluttered user interface elements. The scoring mechanism allowed easy integration of more advanced features, which we have used to support more elaborate interaction tasks in real VR applications, specifically in the case of interface control.

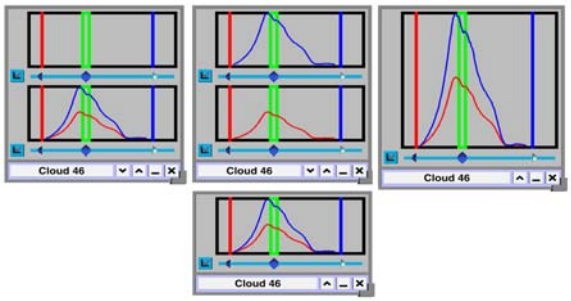

Figure 6: *The graph window features bi-directional communication and extra title bar widgets. Graphs are added or removed by picking the up or down widgets in the title bar. The plots shown in each graph are selected from a dialog which pops up when the button widget to the lower left of each graph is picked.*

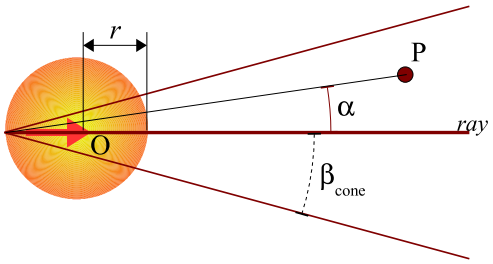

Figure 7: *Scoring Metric: Selection sphere combined with the selection cone (2D section)*

4.1. Transition between direct and remote interaction

In near-field VR setups, such as our Workbench, often a mix of direct and remote interaction techniques are used. Remote interaction techniques are used to interact with objects that are out of (comfortable) reach of the user's arm and interaction device. For example, virtual objects that are placed under the glass surface or near the screen edges are not easily selected with direct techniques. However, direct interaction techniques in particular benefit from co-located interaction and passive haptic feedback from the PlexiPad and the Workbench surface. Direct interaction is nevertheless also hampered by hand jitter, tracking inaccuracies and calibration errors. To combine the benefits of direct interaction and supportive selection we have extended our IntenSelect technique to support direct interaction. We a use selection-by-volume approach to provide a fuzzy preselection of the objects that might be of interest. In the case of remote interaction we use a cone-shaped volume to accomplish this. For direct selection, we use a sphere surrounding the tip of the interaction device. In this selection sphere we also apply a scoring metric to assign scores to objects. The combination of the scoring metrics result in a single ranking list based on accumulated score. In this way, both remote and direct

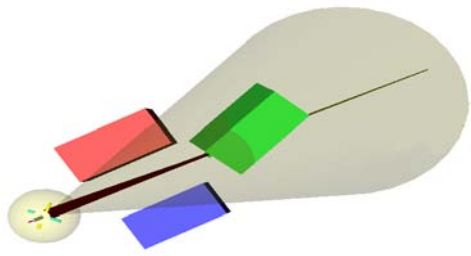

Figure 8: *Selection Volumes: The selection sphere (direct interaction) combined with the selection cone (remote interaction)*

interaction can be performed without ever switching the interaction tool. As shown in Figure 7, both scoring metrics are merged to allow a seamless transition of the scores. We have emphasized the direct scoring value, which results in nearby (directly selected) objects being preferred by the selection mechanism. In Figure 8, we show the visible selection volumes of both the direct and remote scoring. The radius r and cone opening angle β are determined beforehand, based on the system configuration regarding tracking resolution and size of the VE.

4.2. Snapping behavior

When an active object is highlighted by the use of our selection technique, the ray bends and snaps to a certain point on the object. For scenarios where only selection on a object level is relevant, this snapping point is only useful for visual feedback. In our previous implementation, the snapping point of an object was defined as the origin of the bounding volume of an object, which is usually the center point of the object. In some cases however, other snapping points must be provided. One of the more important cases where such a snapping point is needed, is object manipulation. In that case, the snapping point defines an origin of interaction around which the object is transformed. We will discuss the manipulation in the following section. Another scenario is the use of an exact snapping point in information panels or controls, where the snapping point on a surface triggers an information query (e.g. color picking dialog).

In our current implementation we now also provide user-defined and ray-intersection snapping points for use in interaction tools and widgets. When ray-based snapping mode is defined for an object and ranks highest on the scoring list, a regular ray intersection test is performed on the polygons of the active object. If an intersection is found, this point is used as the snapping point (similar to ray-casting). If no intersection is found, either a fixed or the last available snapping point can be used. Figure 9 depicts the three variations in snapping modes. Although still in early development, we have also experimented with nearest intersection point estimation as an optimal snapping point.

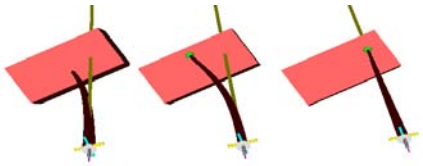

Figure 9: *Snapping Point variations, from left to right: Center point, (pre-defined) fixed point, and ray-based intersection point*

4.3. Object Manipulation

We have extended the basic IntenSelect selection technique with a manipulation technique to allow straightforward object manipulations with continuous visual feedback. By pushing the stylus button, the user seamlessly activates the manipulation mode of the currently selected object. As long as the user holds the button, the bending ray will remain snapped to the active object. For default objects, the manipulation mode enables the repositioning and reorienting of the object. Before starting such manipulation, the original spatial transformation between the virtual pointer and the active object's snapping point are stored. The subsequent translations and rotations of the stylus are then transferred to the object, similar to ray-based manipulation and the Spring Tools [KP01]. If the object can perform these transformations unrestricted, the original bending of the ray will be maintained during manipulation.

If, however, the object's transformations are influenced or restricted by interactions with the environment, the original transformation cannot be (fully) performed. Examples of these transformation restrictions include implicit object movement, constraints or collisions. In these cases, the resulting object transformation under all influences is applied, and the ray is deformed to match the object's final pose. As a result, the bending of the ray will maintain the connection to the object, regardless of external restrictions. In this way, we provide continuous visual feedback to the selection and manipulation actions. In Figure 10, the manipulation sequence of an object is shown. During this manipulation, a collision prevents the object from further motion, while the ray is deformed and remains connected to the snapping point on the object.

4.4. Scoring Response control

The IntenSelect scoring mechanism generally applies the same scoring metric to all the objects in the scene, regardless of their size, orientation or movement. That is, the same score determination is used per frame, including the same scaling parameters, the stickiness and snappiness, used to describe the time dependent behavior of the accumulated score. In our previous user test, some skilled users commented on the imbalance between these scaling parameters.

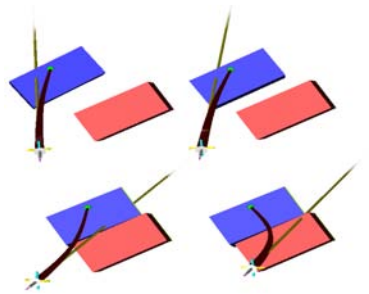

Figure 10: *Manipulating an object using the flexible ray. The blue (left) object is unable to move through the red (right) one, and so the ray must be flexible to remain connected to the blue object.*

They found that, for easy selection tasks in non-occluded, non-cluttered situations, a larger stickiness was hampering fast interaction. This effect was also noticeable in some elements of our test results: in the simplest selection task the time-dependent scoring technique was often outperformed by the snappier, time-independent version. As a contrast, the time-dependent scoring was preferred in selection tasks in tests where objects were moving and cluttered.

We take advantage of this observation by introducing specialized, per object, scoring behavior. This allows us to specify custom scoring response parameters for those objects which might benefit from this. For example, small and cluttered objects can be appointed higher stickiness to ease their selection and disambiguation. During local and precise interaction in this cluttered region the user generally slows down to select the intended object. However in regular interaction situations, tracking and hand jitter will make the necessary disambiguation between the small objects difficult. By increasing the stickiness of the objects, we effectively smooth their scoring values over time to compensate for this. The scoring mechanism which controls the snapping behaves as a low-pass filter. As a result, the bending ray will, albeit with a certain delay, snap to the highest ranking object and will remain snapped for certain longer period of time while the user is trying to hold the selection. The delayed selection caused by the high stickiness can provide the user sufficient opportunity to trigger the intended object. To illustrate this, Figure 11 shows the different accumulated scoring responses when a conic volume sweeps three adjacent objects at a constant speed. For the higher stickiness setting, the score is more spread out, and new objects take a long time before reaching the highest score.

From this observation and current informal experiments, we believe that, ideally, the filtering properties of the score accumulation function should match the context of the scene and the type of objects. We are currently investigating these filtering properties in more detail and, and we hope to discover to what extent automated tuning based on scene con-

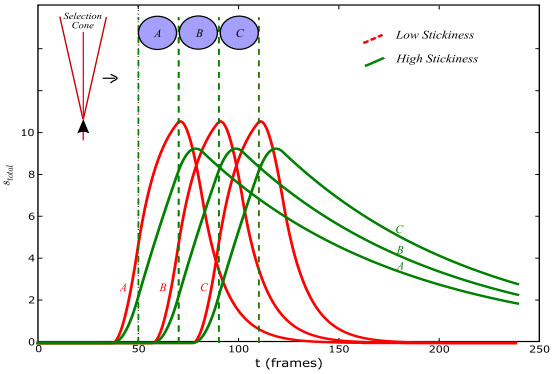

Figure 11: *Scoring response of three objects in line: The scoring response parameters influence the selection timing. The high stickiness setting spreads out the score over time, providing more time for user interaction.*

text can provide improvements in various selection scenarios.

4.5. Scoring redistribution

In most VR application a hierarchy of objects is used to create complex objects while maintaining flexibility in scene management. Often not all parts are useful for selection or manipulation, but only a specific object is. For selection, the nested bounding boxes of a tree of objects and sub- or child-objects might be used, where the parent object's bounding box contains all the bounding boxes of its children. In our original IntenSelect scoring metric we have not taken this object hierarchy into account, which can make the selection of sub-objects contained in other bounding boxes rather difficult. In addition, we want the selection of a parent object to be able to trigger the selection of a sub object. To facilitate this we provide scoring redirection. Here, the scoring value obtained by the parent object(s) in the tree can be redirected to some or all child objects. Especially in situations where large objects have only small selectable and manipulable sub objects, this redirection can be used to trigger the sub objects on selection of the parent object. A useful example of nested UI elements on which we applied redistribution, are our window and widget constructs, shown in Figure 12.

5. Results: Integrating Interaction and Interface

The individual windows, dialogs and control elements are constructed from regular VR objects that work directly with our supported interaction mechanism. To improve the selection and manipulation behavior of our interface elements, we used the interaction extensions described above. In this section we describe the most notable integration results we obtained in several small test applications. We conclude with a description how the bulk of the material integrated into our Cloud Explorer application.

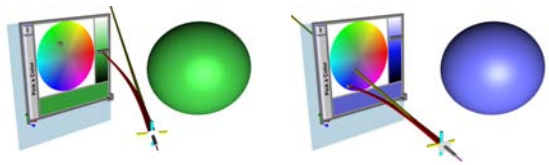

Figure 12: *Color picking on the PlexiPad: supported selection of the small controllers, while manipulation is dynamically constrained.*

5.1. VR System Characteristics

We will briefly overview the characteristics of our primary VR system used in this work. For hardware, we make use of a Infitec-based passive stereo Virtual Workbench setup with a 180×110 cm screen size and a resolution of 1400×860 pixels. For tracking we use a Polhemus FASTRAK electromagnetic tracker tracking 4 devices, each tracked interleaved at 30 Hz. Our system is powered by 2 dual Intel Xeon Linux-based rendering machines. On the software side of our implementation, we worked with OpenGL Performer and an in-house VR library. We make use of VRPN for monitoring our tracked devices, and we have implemented predictive filtering to help compensate for tracking latency.

5.2. Snapping and Constraints

In Figure 12, we illustrate the use of constrained manipulation in the case of the color picker dialog. As described in section 3.3, the dialog's two controller elements can be used to control the selected color value. The entire dialog and window can be repositioned and resized in 3D space, in this case on the PlexiPad, while the controllers' movements are actively constrained to their respective control regions. At the left of figure 12, the controller manipulation is limited to a 1D movement over the lightness bar. At the right of this figure, the second controller is manipulated and restricted to the circular hue/saturation 2D region. In both situations, if the controller movement is restricted by constraint boundaries, the ray is deformed to maintain the flexible connection. Figure 14 shows a user controlling these small dialogs on the PlexiPad. As the windows are fixed to a particular plane in space, in this case the PlexiPad, we use similar constraints on the windows during manipulation. Once the widget is being manipulated, the flexible deformed ray will again maintain its connection, also if stylus and PlexiPad are moved.

As discussed, for some interaction we need precise interaction on information panels and windows. To illustrate this we use ray-based snapping mode in the graph panel to extend the widget based control (see figure 13). Users can directly select a 2D position in the graph layout, which is used, in this case, for updating the current time slider.

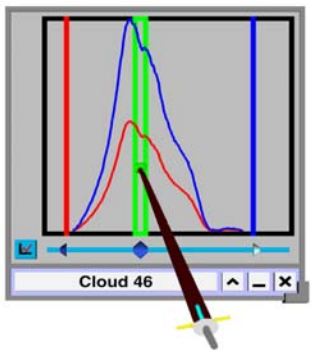

Figure 13: *Ray-based snapping point on the Graph window provides precise a 2D interaction point*

5.3. Selection and Readability

Due to tracker miscalibration, the passive haptic feedback provided by the PlexiPad and the Workbench display did not always work as expected. Although the supported direct manipulation reduced the severity of this problem, selection of small cluttered objects remained difficult due to tracker error. This was doubly the case if objects were placed on the PlexiPad, thus requiring two tracked devices to work together. To help solve this problem, we can use a higher stickiness setting for small individual widgets, such as the buttons in the numeric input dialog. Careful parameter selection provides a subtle smoothing of the jitter for in these scenarios.

IntenSelect permits scaling of widgets down to sizes of only a few screen pixels. On the Workbench we have scaled down the elements as far as text legibility allowed. In this configuration, the text height on the buttons was limited to 10 pixels, which corresponds to about 12mm on our screen, which is well readable. We must note that these dimensions are dependent on the screen resolution and viewing distance, but foremost the font type and anti-aliasing quality. Using these widget sizes, we experience no difficulty selecting the interface elements. Manipulation was also intuitive and easily accomplished, but accurate manipulation at a distance still relies, to some degree, on the user's manual dexterity.

5.4. Integration with Cloud Explorer

The use of our new hybrid interface and interaction technique have made a significant difference in the UI for Cloud Explorer. Figure 15 shows a comparative illustration between the old UI and the new UI. While most of the basic elements from the old UI remain, they are now much smaller and easier to use. The size is critical because Cloud Explorer is still in its infancy, and, yet, the interface is already quite cluttered. With the new UI, irrelevant portions of the interface may be hidden, and various portions can

Figure 14: *Windows on the PlexiPad: Two-handed interface control (see colorplate)*

be moved to accommodate the user's preferences (e.g. left-handed users). With the improved interaction, the buttons, sliders, and clouds are all easier to use. We were surprised at how easily the sliders on the graph windows, being around 15×15 pixels, could be selected and manipulated at distances of over one meter. Direct interaction with UI elements is simple and intuitive in spite of a persistent tracker miscalibration. The addition of the numeric input dialog also addresses a common user complaint for selecting clouds that have died out and are no longer visible. Furthermore, the transition between interacting with the UI elements and the clouds themselves is seamless.

6. Conclusions and Future Work

The use of hybrid interfaces is important in our application area: scientific visualization. We have demonstrated two techniques we developed and integrated to address some of the limitations of hybrid interfaces.

We implemented a new hybrid interface, which offers users a less cluttered and more flexible UI. It integrates command functionality and enriching information in an intuitive manner through the use of the familiar windows paradigm. We make use of constructs such as tooltips and dialogs to maintain a lightweight UI without alienating the novice user.

The new extensions to our IntenSelect method provide the user with more intuitive and easy to use tools to interact with the VE. The use of generic scoring metrics and filtering provided a flexible framework for implementing special interaction behavior. Direct selection and improved object scoring makes it easier for the user to select objects of interest in

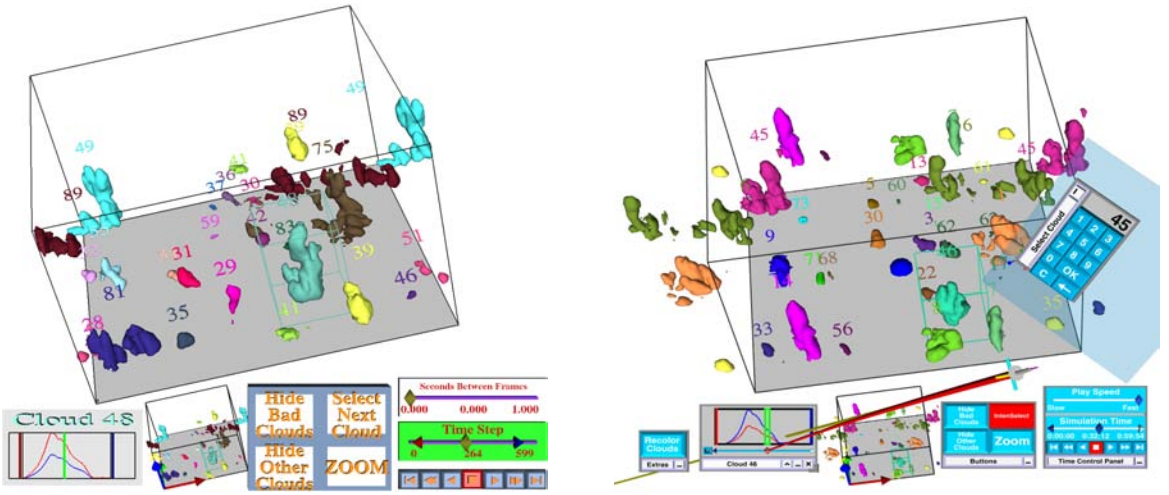

Figure 15: *A comparative illustration between the old Cloud Explorer interface (left) and the new one (right, see colorplate).*

various scenarios. New object snapping and object manipulation techniques allow the user to effect meaningful changes in the virtual environment with less effort.

We have used our Cloud Explorer application as an example of the kind of UI that this integration made possible. The environment is filled with an array of 3D objects and 2D UI widgets, and the transition between interacting with each is seamless. The UI affords more room for the 3D clouds, while also giving the user more flexibility to arrange it to his or her own taste. Less relevant elements can be positioned at the edges of the Workbench without presenting any difficulties to the user if he wishes to read or interact with them.

As the need for more abstract and complex interaction and information panels grows in our exploration applications, we can extend our interface elements to control its specific interaction behavior in more detail. We believe that the integrated solution of flexible interfaces and interaction support allows us to stretch the complexity limit of interface scenarios in VR, without usability issues exploding.

We would like to continue to extend and enhance our hybrid interface in two ways. First, we would like to develop new and easier to use widgets, and offer more intelligent interface management and placement. The latter is useful due to the erratic behavior of overlapping 2D elements. Secondly, we would like to use the interface in various scenarios such as multiple-linked views. Here, the need for a convenient method for managing global state variables through bi-directional interface elements will be necessary.

We continue to extend and develop the (mathematical) foundations of IntenSelect towards improved scoring behavior in various situations. As described earlier, we hope to discover to what extent automated tuning based on scene context can provide improvements in various selection scenarios. Furthermore, we plan to extend our scoring and snapping mechanisms to facilitate cooperative interaction for multiple users and interaction devices.

To strengthen our statements on usability and limitations of our techniques, we plan to fine-tune parameters and perform user tests on a wide variety of VR systems, of which a Personal Desktop VR system and a large CAVE-like system are candidates.

7. Acknowledgements

This research was supported by the Netherlands Organisation for Scientific Research (NWO) and the Dutch BSIK/BRICKS-project.

References

[AS95a] ANGUS I. G., SOWIZRAL H. A.: Embedding the 2D interaction metaphor in a real 3D virtual environment. In *Proceedings of Stereoscopic displays and virtual reality systems II* (1995), pp. 282–293.

[AS95b] ANGUS I. G., SOWIZRAL H. A.: VRMosaic: WEB access from within a virtual environment. In *Proceedings of the 1995 IEEE Symposium on Information Visualization* (1995), p. 59.

[BHB98] BOWMAN D. A., HODGES L. F., BOLTER J.: The virtual venue: User-computer interaction in information-rich virtual environments. *Presence: Teleoperators & Virtual Environments 7*, 5 (1998), 478–493.

[BWH99] BOWMAN D. A., WINEMAN J., HODGES L. F.: The educational value of an information-rich virtual environment. *Presence: Teleoperators & Virtual Environments 8*, 3 (1999), 316–331.

[CRC04] CUPPENS E., RAYMAEKERS C., CONINX K.: VRIXML: A user interface description language for virtual environments. In *Proceedings of Developing User Interfaces with XML: Advances on User Interface Description* (2004), pp. 111–117.

[CRF97] CONINX K., REETH F. V., FLERACKERS E.: A hybrid 2D / 3D user interface for immersive object modeling. In *Proceedings of the 1997 Conference on Computer Graphics International* (1997), p. 47.

[Dan05] DANG N.-T.: The Selection-By-Volume Approach: Using Geometric Shape and 2D Menu System for 3D Object Selection. In *Proc. of the IEEE VR Workshop on New Directions in 3D User Interfaces* (2005), Bowman D., Fohlich B., Kitamura Y., Sturzlinger W., (Eds.), pp. 65–68.

[dHKP02] DE HAAN G., KOUTEK M., POST F. H.: Towards intuitive exploration tools for data visualization in vr. In *Proceedings of the ACM symposium on Virtual reality software and technology* (2002), pp. 105–112.

[dHKP05] DE HAAN G., KOUTEK M., POST F.: IntenSelect: Using Dynamic Object Rating for Assisting 3D Object Selection. In *Proceedings of the 9th IPT and 11th Eurographics VE Workshop (EGVE) '05* (2005), Kjems E., Blach R., (Eds.), pp. 201–209.

[Dyk94] DYKSTRA P.: X11 in virtual environments: Combining computer interaction methodologies. *The X-Resource 9*, 1 (1994), 195–204.

[FK05] FREES S., KESSLER G.: Precise and Rapid Interaction through Scaled Manipulation in Immersive Virtual Environments. In *Proc. of IEEE Virtual Reality Conference* (2005), pp. 99–106.

[FMHR86] FISHER S. S., MCGREEVY M., HUMPHRIES J., ROBINETT W.: Virtual environment display system. In *Proceedings of the 1986 workshop on Interactive 3D graphics* (1986), pp. 77–87.

[FMHS93] FEINER S., MACINTYRE B., HAUPT M., SOLOMON E.: Windows on the world: 2D windows for 3D augmented reality. In *Proceedings of the 6th annual ACM symposium on User interface software and technology* (1993), pp. 145–155.

[GPK*05] GRIFFITH E. J., POST F. H., KOUTEK M., HEUS T., JONKER H. J. J.: Feature Tracking in VR for Cumulus Cloud Life-Cycle Studies. In *Proceedings of the 9th IPT and 11th Eurographics VE Workshop (EGVE) '05* (October 2005), Kjems E., Blach R., (Eds.), pp. 121–128.

[KP01] KOUTEK M., POST F.: Spring-based manipulation tools for virtual environments. In *Proc. Immersive Projection Technology and Eurographics Virtual Environments '01* (2001), pp. 61–70.

[LB03] LARIMER D., BOWMAN D. A.: VEWL: A framework for building a windowing interface in a virtual environment. In *Proceedings of INTERACT: IFIP TC13 International Conference on Human-Computer Interaction* (2003), pp. 809–812.

[LG94] LIANG J., GREEN M.: JDCAD: A highly interactive 3D modeling system. *Computers & Graphics 18*, 4 (1994), 499–506.

[LSH99a] LINDEMAN R. W., SIBERT J. L., HAHN J. K.: Hand-held windows: Towards effective 2D interaction in immersive virtual environments. In *Proceedings of IEEE VR* (1999), p. 205.

[LSH99b] LINDEMAN R. W., SIBERT J. L., HAHN J. K.: Towards usable vr: an empirical study of user interfaces for immersive virtual environments. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1999), ACM Press, pp. 64–71.

[LST01] LINDEMAN R. W., SIBERT J. L., TEMPLEMAN J. N.: The effect of 3D widget representations and simulated surface constraints on interaction in virtual environments. In *Proceedings of IEEE VR 2001* (2001), pp. 141–148.

[Osa05] OSAWA N.: Enhanced Hand Manipulation for Efficient and Precise Positioning and Release. In *Proceedings of the 9th IPT and 11th Eurographics VE Workshop (EGVE) '05* (October 2005), Kjems E., Blach R., (Eds.), pp. 221–222.

[SES99] SCHMALSTIEG D., ENCARNAÇÃO L. M., SZALAVÁRI Z.: Using transparent props for interaction with the virtual table. In *Proceedings of the 1999 symposium on Interactive 3D graphics* (1999), pp. 147–153.

[SG97] SZALAVÁRI Z., GERVAUTZ M.: The Personal Interaction Panel: A Two-Handed Interface for Augmented Reality. In *Computer Graphics Forum 16(3)* (1997), pp. 335–346.

[SP04] STEED A., PARKER C.: 3D Selection Strategies for Head Tracked and Non-Head Tracked Operation of Spatially Immersive Displays. In *8th International Immersive Projection Technology Workshop(IPT2004)* (2004).

[SRH05] STEINICKE F., ROPINSKI T., HINRICHS K.: VR and Laser-Based Interaction in Virtual Environments Using a Dual-Purpose Interaction Metaphor. In *In IEEE VR 2005 Workshop Proceedings on New Directions in 3D User Interfaces* (2005), pp. 61–64.

[vTRvdM97] VAN TEYLINGEN R., RIBARSKY W., VAN DER MAST C.: Virtual data visualizer. *Transactions on VIsualization and Computer Graphics 3*, 1 (1997), 65–74.

[WDC99] WATSEN K., DARKEN R. P., CAPPS M.: A handheld computer as an interaction device to a virtual environment. In *Proceedings of the International Projection Technologies Workshop* (1999).