Extended Pie Menus for Immersive Virtual Environments

Sascha Gebhardt, Sebastian Pick, Franziska Leithold, Bernd Hentschel, and Torsten Kuhlen

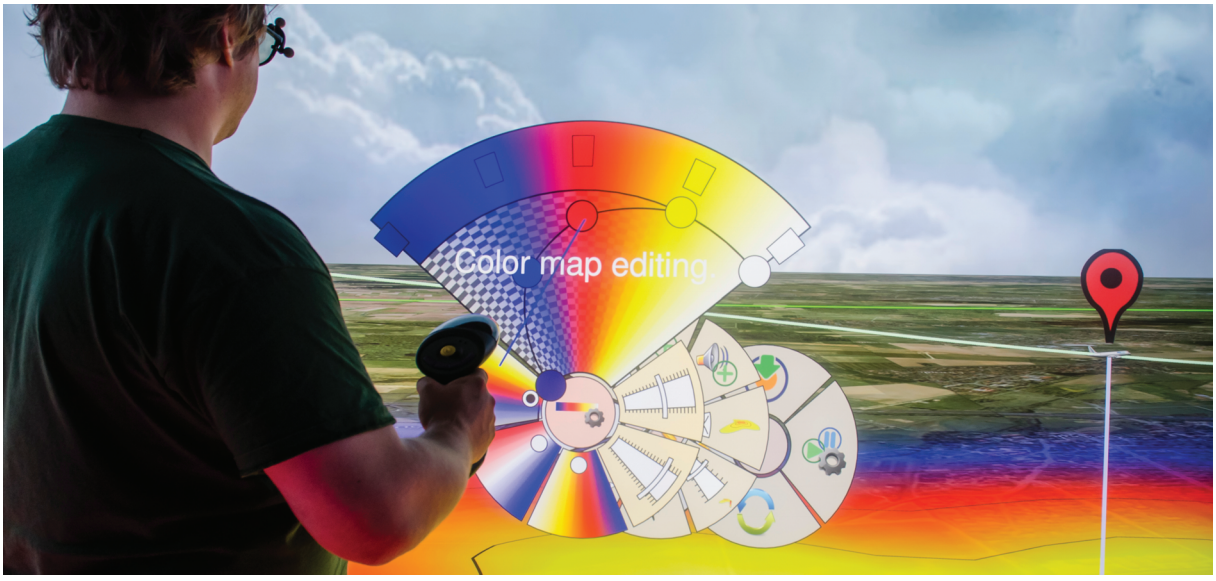


Fig. 1. Test application that was used for the expert review. A VR expert is testing the color map editor of the extended pie menus.

Abstract—Pie menus are a well-known technique for interacting with 2D environments and so far a large body of research documents their usage and optimizations. Yet, comparatively little research has been done on the usability of pie menus in immersive virtual environments (IVEs). In this paper we reduce this gap by presenting an implementation and evaluation of an extended hierarchical pie menu system for IVEs that can be operated with a six-degrees-of-freedom input device. Following an iterative development process, we first developed and evaluated a basic hierarchical pie menu system. To better understand how pie menus should be operated in IVEs, we tested this system in a pilot user study with 24 participants and focus on item selection. Regarding the results of the study, the system was tweaked and elements like check boxes, sliders, and color map editors were added to provide extended functionality. An expert review with five experts was performed with the extended pie menus being integrated into an existing VR application to identify potential design issues. Overall results indicated high performance and efficient design.

Index Terms—Pie menus, interaction, user interfaces, user study

1 Introduction

Configuration and operation of applications can be implemented in a broad variety of ways and a common example is the use of context menus. In classic desktop environments, context menus are usually realized as lists, where a certain command is assigned to each item. An alternative way to implement context menus is to realize them as *pie menus* [4]. These are circular menus with slice-shaped menu entries, all having an equal distance to the input device and an infinite target size. Thereby, they tend to be very efficient with regard to Fitts' law [7]. Though context menus are a good choice for issuing commands, more complex configuration tasks such as selection from item sets or setting scalar values require additional user interface (UI) elements like radio buttons or sliders.

So far, many studies on pie menus have been performed, but they focused mostly on two dimensional UIs and conventional input devices such as mice or pen tablets. However, pie menus have been used in various applications for immersive virtual environments (IVEs) which feature a different class of input devices and display systems. Only little research has been done on the usability of pie menus in this context. Therefore, it is worthwhile to formally evaluate the performance of pie menus in IVEs in order to compare several common variations to each other and to enhance them with extended functionality to facilitate their use for complex configuration tasks. To this end, we followed an iterative development process to design and implement a pie menu system with extended functionality for the use in IVEs that is operated with a six-degrees-of-freedom (6-DoF) input device. We first performed a pilot user study with classic hierarchical pie menus in order to identify the best performing selection method and also analyzed the secondary parameters layout, abort method, and dead zone size.

Based on the results of the pilot study, we enhanced the system by adding a variety of special-purpose menu elements for more complex

- *Sascha Gebhardt, Virtual Reality Group, RWTH Aachen University. E-mail: gebhardt@vr.rwth-aachen.de.*
- *Sebastian Pick, Virtual Reality Group, RWTH Aachen University. E-mail: pick@vr.rwth-aachen.de.*
- *Franziska Leithold, Institute for Information, Organization and Management, Munich School of Management, LMU München. E-mail: leithold@bwl.lmu.de.*
- *Bernd Hentschel, Virtual Reality Group, RWTH Aachen University. E-mail: hentschel@vr.rwth-aachen.de.*
- *Torsten Kuhlen, Virtual Reality Group, RWTH Aachen University. E-mail: kuhlen@vr.rwth-aachen.de.*

Manuscript received 13 September 2012; accepted 10 January 2013; posted online 16 March 2013; mailed on 16 May 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

configuration tasks such as check boxes, sliders, and color map editors. We evaluated the extended pie menus with an expert review to identify potential usability problems. Results indicate that the extended features are very useful, although smaller issues could be identified.

To summarize, the contribution of this paper is twofold. First, we formally evaluated interaction variations with pie menus in IVEs with focus on entry selection. Second, we presented a new way of performing complex configuration tasks in IVEs by adding extended functionality to pie menus.

The rest of the paper is structured as follows. After discussing prior research relevant to our system (sec. 2), the basic version of our pie menus and results from the pilot study will be outlined (sec. 3). The extended version of the menu will then be introduced (sec. 4) and outcomes of the expert evaluation will be discussed (sec. 5) before concluding this paper (sec. 6).

2 Related Work

Callahan et al. [4] compare non-hierarchical pie to linear menus in a 2D desktop and mouse environment and state the superiority of pie menus with regard to speed and error rate. They explain their results with a better distance-to-target-size ratio of individual entries of pie menus. However, they do not find differences in user satisfaction among the tested menu types.

Hopkins [10] was the first one to mention the possibility for the user to 'mark ahead'. This means, that entries can be selected even before the menu is actually drawn onto the screen. This idea has been studied more intensely by Kurtenbach et al. by introducing marking menus, which are an extension to hierarchical pie menus [12]. In their application, the cursor leaves an ink trail—the mark—and the underlying pie menu is only presented if the user needs a hint on which selection choices are available. Thus a transition from novice to expert users is realized. Extensions or variations of marking menus are for example realized by Tapia and Kurtenbach, who presented design refinements, like showing only text labels or idealizing drawn marks [19]. Zhoa and Balakrishnan developed hierarchical marking menus, where multiple straight strokes are used for selection instead of one single zig-zag line [20], because the latter could introduce ambiguities. For example one zig-zag line going up and down could be either interpreted as up-up-down or up-down-down with both variants representing valid menu selections. Though marking menus outperform normal pie menus with regard to interaction time, this concept is not useful for our approach of extended pie menus, because some of the extended elements, such as sliders, must always be visible for interaction.

Mackay empirically compared different post-WIMP interaction techniques against each other in order to find out which technique fits best for which type of task [15]. Floating palettes, marking menus and toolglasses where subjects of her study. She found that marking menus are most efficient for object-oriented tasks where multiple modifications have to be carried out on the same object or the same modification has to be carried out on multiple objects. This supports our approach of using extended pie menus as context menus, especially for object manipulation.

When moving from 2D to 3D menus, Bowman et al. should be considered, where various mappings of input data to entries of linear lists for use in IVEs are discussed [2]. One such mapping is the 1-DoF input metaphor for which one dimension of a linear motion or rotation of a 6-DoF input device is used to browse through list entries. The reduction in DoF is supposed to increase selection performance. They further discuss the TULIP metaphor, for which gloves with touch-sensitive finger tips are used [3]. Here, menu entries are mapped to fingers which can then be selected by touching the respective finger with the thumb.

Ni et al. presented another freehand menu technique, the roll-and-pinch menu (rapMenu) [16]. Here, elements are chosen by focusing a group of four elements from a circular menu through wrist-rotation and then selecting the desired element by touching the corresponding finger with the thumb. Ren and O'Neil presented a 3D marking menu technique based on freehand gestures [18]. The menu's items are arranged on two overlapping circles, a horizontal and a vertical one, with eight menu items each. The TULIP metaphor, as well as the rapMenu and Ren's 3D marking menus are based on hand respectively finger tracking. This makes them hardly usable for settings where the scene is manipulated via a hand-held 6-DoF input device.

Komerska et al. compare different types of menus in a 3D fish tank virtual reality (VR) environment using a haptic input device [11]. Their study shows that pie menus with an exceed border technique and a planar assistive force are fastest. Adding a haptic border and a snap constraint slightly decreased speed but increased accuracy considerably. However these techniques do not work in settings where haptic feedback is missing.

Cao's and Balakrishnan's VisionWand [5] is an input device for interaction with large displays. A colored stick—the wand—is tracked three-dimensionally by two cameras. Thus 5-DoF input is realized (a sixth degree of freedom is missing, because rotation around the wand's own axis is not tracked). A pie menu has been implemented to be operated with the wand. Here, the selection of entries is realized by rotating the wand, what realizes a 1-DoF selection for the pie menu. We tested a similar metaphor in our pilot study.

In Grosjean's command and control cube (C3) [8] menu items are arranged in a $3 \times 3 \times 3$ cube. Single elements can be selected by moving the hand to the position of the desired element within the cube. We implemented a similar approach for the selection of menu items to be tested in our pilot study with the difference that we project the hand movement onto the two-dimensional menu plane.

Das and Borst compare pie menus against linear menus in a VR setting using a 6-DoF input device [6]. They also compare two different selection techniques for pie menus against each other: selection with a pick ray against pointer-attached to menu (PAM) selection. It was shown that pie menus were faster than linear menus and that pick ray selection was faster than PAM. They also tried different locations for the menus to appear, and showed that contextual menus were faster than menus with a fixed location. Their study shows that Callahan's results regarding the comparison of pie menus against linear lists are also true for IVEs. The finding that pick ray selection is more efficient than PAM makes it unnecessary for us to include PAM in our pilot study, however we compare pick ray to two other selection techniques. We intend to explore interaction with pie menus in VR further, introducing three different selection techniques for pie menus in VR settings and evaluating them with regard to several measures, thus extending the work of Das and Borst.

3 Basic Menu Design and Pilot Study

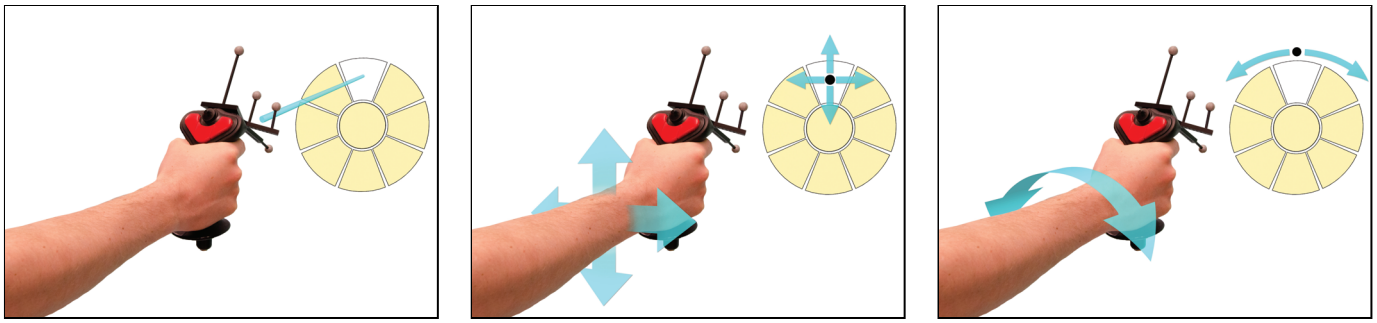
Before implementing pie menu extensions, we conducted a pilot study to investigate how pie menus are operated best in IVEs. We performed a user study, measuring task execution times, error rates and subjective ratings of users. In the following section we describe the basic menu design and the implemented variations, before presenting the study and the results.

3.1 Menu Design

In IVEs, capabilities for designing real 3D UIs are provided via 3D view, which is achieved by combining stereoscopic display systems with user-centered projection in combination with tracked 6-DoF input devices that can be used to design three-dimensional interaction metaphors [2]. In order to find the best way to realize hierarchical pie menus in IVEs, we implemented multiple variants of different aspects of pie menus, namely, *selection method*, *hierarchical menu layout*, *abort method* and *dead zone size*. The implementation is based on the platform-independent VR toolkit ViSTA [1], which provides abstractions for display systems and input devices, thereby allowing for quickly adapting applications to different target systems.

3.1.1 Entry Selection

The entry selection task is composed of two substeps: *focusing* the entry that is to be selected and selection *confirmation* by issuing a command. Focus is indicated by increasing an element's radius by 10% and highlighting it with a different color. We evaluate three common selection variants: *pick ray*, *hand projection* and *hand rotation* (Fig.



a) pick ray b) hand projection c) hand rotation

Fig. 2. The different selection methods that were tested in the pilot study.

2). All three methods have in common that they use tracking data of a 6-DoF input device and that selection is confirmed by pressing a designated button. To avoid ambiguities as described by Zhao [20], we decided not to realize confirmation without button press.

Menus are always positioned centered at the pointing direction of the input device and at a fixed distance to it. Consequently, submenus will open with their center at the location at which selection was confirmed in the parent menu. In the following, the different selection methods will be explained in more detail.

Pick Ray (S-PR) With this selection method, entries are focused by pointing at them (Fig. 2a). A ray is cast from the center of the input device. The intersection point between the pie menu's plane and this ray determines the focused element. A visual representation of the ray is displayed as feedback to the user.

Hand Projection (S-HP) The input device's position is used to determine the focused menu entry (Fig. 2b). It is similar to the selection in the C3 [8], but the device position is projected into the pie menu plane, making the selection two-dimensional. A small sphere indicates the projection point.

Hand Rotation (S-HR) This selection technique captures the rotation of the user's hand around the axis of the forearm to determine the focused menu entry (Fig. 2c). It is a form of one-dimensional input and therefore, S-HR is not a realization of a pie menu, but of a 1-DoF circular menu as it is dicussed in [2] and was previously similarly realized in [5]. Nevertheless, we chose to include this method in our evaluation as it feels natural and fits the visual representation of pie menus well. To determine the focused menu entry, the hand's rotation is mapped to a radial position in the menu and the respective pie menu element is focused. To reduce rotation strain, the measured angle is multiplied by a constant amplification factor of 2.5 (comparable to a steering wheel in a car), enabling the user to comfortably focus all menu entries. This factor has been determined by informal testing prior to the study. In order to clarify the relation between hand posture and focused entry, a small sphere indicating the current radial position is displayed outside of the pie menu (Fig. 2c).

3.1.2 Hierarchical Menu Layouts

In order to avoid visual clutter, the different layers of hierarchical pie menus should be arranged with a structured menu layout. When defining a layout strategy, a trade-off has to be found between the amount of occlusion caused by the layout and the amount of information it provides about the menu hierarchy, because both of these parameters could influence performance.

To take this into account, we designed three layout strategies, namely *linear in-plane offset*, *depth offset* and *in-plane offset* (Fig. 3).

Depth Offset (L-DO) Here, parent menus get pushed away from the user whenever a submenu is opened (Fig. 3a). This way, the visual footprint of the layout is kept small to minimize occlusions of scene objects. The layout still provides information about the menu hierarchy, even though parent menus are partly occluded by their children (self-occlusion).

In-plane Offset (L-IO) This layout strategy pushes parent menus to the opposite direction of the selected entry (Fig. 3b). This yields a

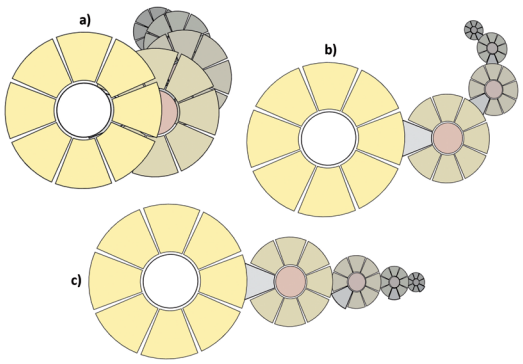


Fig. 3. The different hierarchical menu layouts that were tested in the pilot study: a) depth offset, b) in-plane offset, c) linear in-plane offset.

layout with similar shape as the depth offset layout, but has the advantage that parent menu entries are not occluded. Parent menus touch their immediate child with the selected entry. As a result, this strategy offers more context information than L-DO by showing the selection history. However, the layout footprint becomes bigger, leading to a higher potential for occlusion of scene objects. To partly reduce this effect, parent menus are scaled down as the user moves deeper into the hierarchy, but stay in the same plane. Additionally, this layout does not completely resolve self-occlusion issues. When the user selects an entry that lies roughly in the opposite direction of the entry that was selected from its parent menu level, the newly opened menu fully occludes its grandparent menu.

Linear In-plane Offset (L-LO) To avoid self-occlusions, we introduce this third layout strategy. It is a variation of L-IO, in which the active menu is touched by its parent with the selected entry. All other menus align linearly to it (Fig. 3c). Consequently, self-occlusions cannot occur anymore, although other drawbacks are introduced. All ancestor menus must be rearranged upon opening a sub menu, what can lead to a lot of movement, which could distract the user. Additionally, the potential for occlusion of scene objects is rather high, as menus stretch far into one direction. Also no selection history is provided beyond the very last selection.

3.1.3 Dead Zone Sizes and Abort Methods

The dead zone is an inactive area in the center of the menu, which guarantees that no element is selected when the menu gets opened until user interaction occurs. We embedded three predefined dead zone sizes for the user study. For the use with the S-HR, the dead zone size is set to 0, thus completely removing the dead zone. For the other selection methods two different sizes are available, a large one, covering about 40% of the visible radius of the pie menu and a small one, which covers about 15%.

As for any other hierarchical menu, it is also necessary to provide a method to abort the current menu interaction in order to ascend one level in the menu hierarchy. For this purpose, three different alterna-

tives have been realized: *dead zone*, *pie menu slice* and *button*. No matter which abort method is used, the active menu closes without triggering any action when an abort command is issued. If the closed menu had a parent, this gets re-centered on the currently focused point in the closing menu and becomes active again—just as though it had been newly opened. The rest of the menu hierarchy is updated according to the active layout strategy.

Dead Zone (A-DZ) With this first method, aborts are issued by selecting the dead zone. The dead zone abort method works only in conjunction with the S-PR and S-HP selection methods, not with S-HR. This is because the dead zone cannot be mapped to an angular position with S-HR and thus cannot be focused.

Pie Menu Slice (A-PS) With this method, an additional abort pie slice is inserted at the bottom of the pie menu which triggers abortion when clicked. It can be selected like any other slice, thus making it part of the main selection task while being colored differently for better distinction.

Button (A-BU) The last method uses an additional input device button for aborting. This way, the pie menu's functionality does not have to be augmented to provide an abort mechanism.

3.2 Experimental Evaluation

The basic pie menu system was evaluated with a user-centered evaluation. In the user study, both quantitative and qualitative data were gathered using performance measures, questionnaire data and free comments of participants. Rather than to prove existing theory, our aim was to provide informed propositions for current and future research. For this purpose we used an explorative approach by applying a within-subject design in which each participant had to perform different tasks using all variants of the interface. Data was recorded using log files, questionnaires and observational notes. We measured task execution time and error rate as dependent variables while using *selection method* as independent variable. Other attributes of the menu (*hierarchical menu layout*, *abort method*, and *dead zone size*) were varied as well and included in the latter analysis. Experimental evaluation results will be presented and discussed accordingly.

3.2.1 User Study Setup

For the quantitative evaluation, user performance with regard to task execution times and error rates was measured. Qualitative data was recorded using usability and user experience questionnaires. Prior to the study, users were informed of the anonymous and confidential use of their data and their right to quit the study at any time. Regarding right- and left-handed participants, all users were free to use the input device with their preferred hand. All of them had normal or corrected to normal vision. During the study, participants were instructed by one supervisor, while one observer transcribed all user comments and noted problems and irregularities.

Users had to perform two different task types during the study, using varying parameter combinations: *object manipulation tasks* (OMT) and *number entry tasks* (NET). These synthetic tasks were designed for the purpose of measuring system performance under different conditions. Every parameter combination was tested with both, OMTs and NETs.

In OMTs, the appearance of an object (color, shape and texture) had to be changed by the participants. Therefore, a menu with two levels (parameter—modification of parameter) was provided which opened when the user clicked the object that was to be modified. For example, to change an object's color to green, it was necessary to click the object, choose *color* in the first and *green* in the second menu level (Fig. 4). An OMT concluded after all sub tasks had been performed correctly. Errors were measured by counting erroneous selections.

In NETs, users had to enter a five-digit number into a text box. The menu consisted of five levels, one for each digit, each of them containing all digits from 0 to 9. A NET concluded after the fifth digit was selected and the overall number was entered correctly. Errors were measured as during OMTs.

Participants completed a basic questionnaire about demographics and prior experiences with VR and were then introduced to the VR hardware (see sec. 3.2.2). During the introduction, the study application was running in sandbox mode, meaning all parameters of the pie menu could be changed freely. The parameter variations were explained to the participants who could familiarize themselves with the system. They were told to accomplish the tasks as normally as possible, thus, neither trying to hurry nor to play around with the system.

The user study was divided into three blocks, each covering one selection method. Altogether, 18 tasks had to be performed, resulting in six tasks per block. Throughout one block, the *selection method* remained constant, while the other parameters—*hierarchical menu layout*, *abort method*, and *dead zone size*—were pseudo-randomly changed from task to task. Every block contained both OMTs and NETs, which appeared in pseudo-randomized order to every participant. Before each task, participants could familiarize with the configuration for the upcoming task. After each finished block, participants had to fill out usability and user experience questionnaires with regard to the last-used selection method. At the end of the study, subjects had to fill out a final questionnaire in which they could rate all variants of the tested parameters.

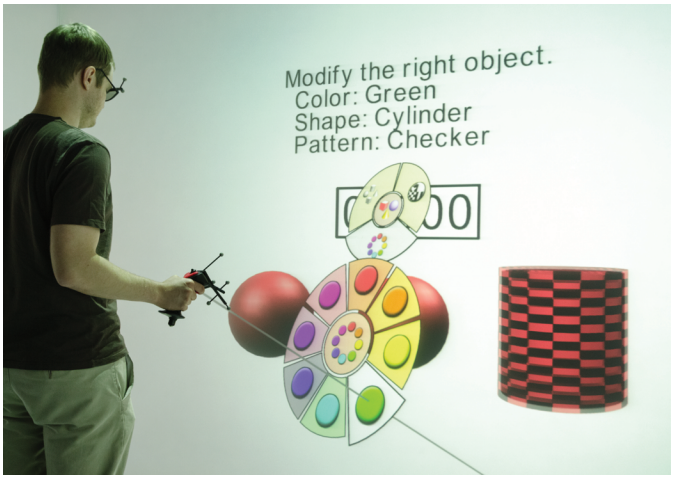


Fig. 4. User study setup in task mode. The user is about to change the color of the right object to green.

3.2.2 Hardware Setup

The user study was performed in a 5-sided CAVE-like display system, but the back wall was open and disabled for the user study. The walls are back-projected, while the floor is top-projected. For the side walls the resolution is 1200×1200, otherwise it is 1600×1200 pixels. Passive stereo is used. The CAVE's dimensions are $3.6m \times 2.7m \times 2.7m$ and it is equipped with A.R.T. infrared optical tracking with an update rate of $60Hz$ and an end-to-end latency of $\leq 100ms$. An A.R.T. Flystick was used as 6-DoF input device.

3.2.3 Results

Overall, 24 participants completed the user study (5 female, 19 male) with 13 participants already having experience with VR systems. The average age of participants was 29 years. 22 persons were right-handed and 2 left-handed, so explanatory power of the results might be stronger for right-handed people. Study results can be separated in four major areas: performance on tasks (task execution time and error rate), usability questionnaire results, concluding questionnaire results and voluntary comments of participants. The sample contained 378 measurements in general (21 valid log files with 18 tasks per participant, 3 log files could not be used due to data corruption).

Task Execution Times Results on task performance in general showed a significantly shorter task execution time for NETs in comparison to OMTs ($t=5.46$; $p < .001$). This result can be explained by the study design, as a two-layer menu had to be opened three times for the OMTs, requiring a total of 9 clicks to complete the task while only one five-layer menu had to be opened in the NETs, requiring 6

clicks for task completion. The average task execution time for NETs over all participants was 10.15 seconds, while for OMTs it was 12.93 seconds. The error rate was highest for the S-HR selection method and object manipulation which could also have had an influence on the higher average task execution times for OMTs.

To find statistically significant differences in task execution times, analysis of variance (ANOVA) was used. Average task execution time for S-PR was 10.39 seconds, for S-HP 12.11 seconds and for S-HR 12.12 seconds. Accordingly, we found a highly significant difference in task execution times between selection methods (F=4.85, p=.008). Comparing the different input methods, analysis resulted in no significant differences between S-HP and S-HR (t=-.003, p=.997). However, highly significant differences in task execution times between S-PR and S-HP (t=2.67, p=.008) and even higher differences between S-PR and S-HR (t=2.96, p=.003) could be observed. These results show clearly that participants could manage the tasks most efficiently using the S-PR selection method. Average task execution times (mean and standard deviation) for selection methods are shown in figure 5.

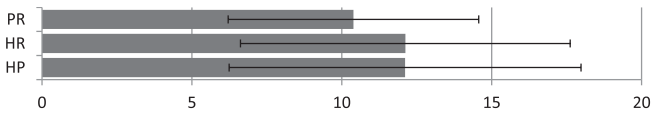


Fig. 5. Average task execution times (in seconds) per selection method. Error bars show standard deviation.

Analysis of task execution times with regard to menu layout showed no significant differences between the three layouts (F=1.72, p=.180). Results showed that L-DO displayed minor superiority in comparison to L-IO (t=1.43, p=.155) and almost significant superiority to L-LO (t=1.91, p=.056). Average times are shown in figure 6. Results showed no major difference in task execution times with regard to different dead zone sizes (F=1.409, p=.246). Average task execution time for the large dead zone was 11.46 seconds while it was 11.04 seconds for the small dead zone.

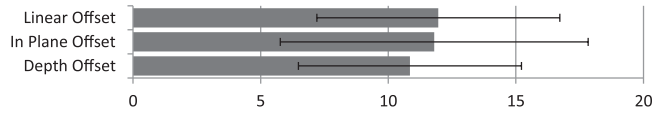


Fig. 6. Average task execution times (in seconds) per layout. Error bars show standard deviation.

Two-Way ANOVA demonstrated a highly significant influence of *selection method* on task execution time (F=4.83, p=.008) whereas *hierarchical menu layout* did not have a significant influence (F=1.75, p=.175). *Hierarchical menu layout * selection method* did not have a significant influence on task execution time (F=.484, p=.747). No significant influence on task execution time appeared for *selection method * dead zone size* and *selection method * abort method* as well as other combinations of the former.

Error Rates Error rates were low in general. Over all 36 tasks, the ratio of erroneous to correct executions was 0.05. 71% of the errors were made during OMTs (17 errors vs. 7 errors during numeric tasks). With regard to the selection method, 46% of the errors were made using S-HR, 29% using S-PR and 25% using S-HP.

Usability Questionnaires To measure user experience and usability of the different input methods, the Computer Systems Usability Questionnaire (CSUQ) [14], the AttrakDiff [9] and the User Experience Questionnaire (UEQ) [13] were used. UEQ and AttrakDiff (semantic differentials) were used on a 5-point scale, CSUQ on a 7-point Likert scale. In the CSUQ, four subscales provide detailed information on usability aspects of the system. AttrakDiff and UEQ have four and five subscales respectively and were combined to obtain broad and comprehensive information about the users' experience of the system. Participants were asked to fill out all three questionnaires after each test block in order to observe differences in usability and user experience for the different selection methods.

In accordance with task execution times and error rates, results showed a clear superiority of S-PR selection with regard to user preference in all questionnaires. It scored highest on all scales of all questionnaires, except novelty in UEQ and hedonic quality–stimulation in AttrakDiff. These two lower scores could be explained by users' general familiarization to pointing, in comparison to the other two—more abstract—selection methods. S-HP scored lowest on all scales of all questionnaires, except INFOQUAL in CSUQ and hedonic quality–identiy in AttrakDiff.

Concluding Questionnaire Results After the test session, participants were asked to fill out a concluding questionnaire, comprising summarizing questions about the tested parameters. In accordance with preceding results, S-PR selection method scored best, followed by S-HR and S-HP. Dead zone as abort method was clearly preferred by most of the participants. L-DO was marginally superior in comparison to the other layouts.

Free Comments of Participants Nearly all participants named S-HP as the least convenient selection method. Coordination and physiological stress issues of arm and hand were mentioned as reasons. This was the case for S-HR, too, though opinions of participants were mixed and the need for practice with S-HR was stated. However, three participants liked S-HR the best, naming the swiftness of this method as particularly positive. The majority of participants named the S-PR as the most convenient, simple and intuitive selection method. This stands in accordance with task execution times, error rates and usability questionnaires. However, four participants had difficulties with S-PR, being confused by the ray penetrating the menu. L-LO was named inconvenient by five participants, two participants explicitly named L-DO as most convenient, which again stands in accordance with the other results. No certain conclusions from free text comments concerning the dead zone sizes could be drawn as namings were few and varied strongly.

3.2.4 Summary and Discussion

Results demonstrated that S-PR is the best selection technique, as demonstrated by time measurements as well as the usability questionnaires and free text comments of the participants. We assume that the reason for these results is a low amount of hand movement and the fact that direct pointing with S-PR is less abstract than the other two techniques. Especially, the amount of strain with regard to wrist rotation in S-HR and strain on the arm in total in S-HP might have led to higher user satisfaction with S-PR.

Among the tested layout strategies, L-DO produced the lowest average task execution times. It causes the least movement of menus and we assume that it therefore causes the lowest user distraction what would be beneficial for task performance. Participants strongly voted for using the dead zone to abort the current menu. No certain conclusions can be drawn about the optimal size of the dead zone.

4 Extended Pie Menus

Pie menus have proven to be a good way to realize menu functionality in IVEs, thus making them easier to operate and configure [6]. In the last section we showed how they should be operated best in IVEs. Nevertheless, classic pie menus only offer the possibility to issue commands. Other tasks like adjusting a scalar value or selecting individual entries from a list can hardly be covered. Obviously, it would for example be possible to adjust a scalar value by opening a menu, clicking on $+/-$ elements and repeat this several times, but this is neither intuitive nor efficient.

In classic 2D user interfaces, tasks like the aforementioned ones are realized by special UI elements like sliders or check boxes. In order to bring this functionality to IVEs without having the need of displaying toolboxes or other widgets, which could disturb immersion and cause occlusion, we chose to realize several classic UI elements as parts of pie menus. The idea is to make IVEs configurable freely by only using one context menu system that has minimal impact on the immersion as it is only displayed on demand.

After having performed our pilot study, we implemented extensions to the pie menu functionality, thereby considering the study's results. Afterwards we evaluated these extensions in an expert review, more precisely a cognitive walkthrough with five participants. We realized the following basic elements as parts of pie menus: *check boxes*, *radio buttons*, *sliders*, hue/lightness *color pickers* and *color map editors*. Based on a hue/lightness color picker and sliders for red, green, blue, alpha and saturation we also added a pre-designed *color chooser menu* to our pie menu system (Fig. 7).

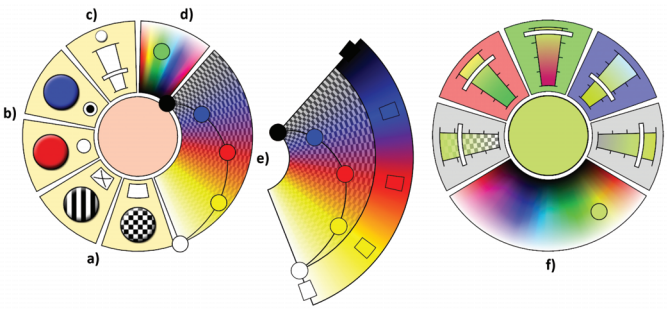


Fig. 7. Pie menu extensions: a) check boxes, b) radio buttons, c) slider, d) color picker, e) color map editor, f) color chooser menu.

4.1 Common Concepts

Basic menu interaction is configured according to the results of the pilot study (see sec. 3.2.3). Selection in extended pie menus is realized with a pick ray and menus are closed by selecting the dead zone, whose radius is 38.2% of the menu's radius (this corresponds to the large dead zone radius in the pilot study). The layout for menu hierarchies is *depth offset*.

All new elements share a set of common attributes, most of them taken from the original pie menu elements of the pilot study stage, which will be referenced as *buttons* from now on. All elements can be assigned an icon. Due to additional drawing space that is needed for most of the new elements, icon placement must be individually solved, according to the visual representation of the element. In addition to the icon, we added captions for menus and elements. The menu's caption is displayed above the menu, while the caption of the focused element is displayed below the menu. We chose to only display the caption of the currently focused element to reduce text cluttering.

While implementing the extension type elements, we recognized, that some of them entailed certain requirements. Those are for example, high radial precision (e.g. value sliders for setting a value) or high angular precision (e.g. color map element for placing control points). To increase precision, two parameters can be tuned when configuring a menu. The first one is a radial increase when an element is in focus. This value defaults to 110%, meaning that an element gets resized to 110% as visual feedback when focused. This value can be changed for each element individually in order to achieve higher radial precision. For example, it is 150% for sliders by default, which showed to be a good trade-off between occlusion of scene elements and precision.

The second parameter is layout scaling. By default all elements have a layout scaling of 100%, which makes them take the same angular space in a menu. This value can also be changed individually for each element. For example, in a menu where a color map editor and two sliders for the minimum and maximum values of the color map editor are present, it would be useful to have the color map editor larger than the sliders, because it is the most important element in this setup and also needs more angular precision to place and modify control points. The results of modifying this parameter can be seen in both menus in figure 7, where the color map editor in the left menu and the color picker in the right menu are each scaled to 300%, thus taking three times the space of the other elements.

Especially when using menus that do not close upon interaction, users could tend to gaze between the pie menu and other scene elements. Hereby a context switch could occur, which could make the user forget that she is actually operating the pie menu, and thus try to interact with other scene elements. Due to the infinite target property of pie menu elements, clicking anywhere in the scene would trigger an action in the pie menu. In order to avoid this, we implemented *extending pie elements*, that extend, when the intersection point of the ray and the menu leaves the element's area. That means, the radius of the element is dynamically increased in a way that the pick ray always intersects the selected pie menu element (Fig. 8).

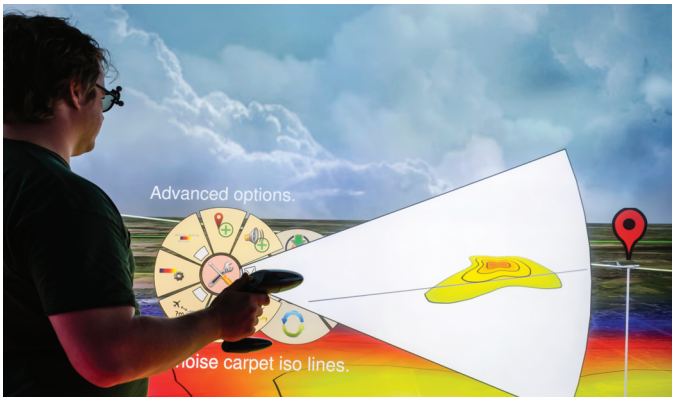


Fig. 8. A pie menu element is extending with cursor to indicate that the menu is active and that no other scene interaction is possible at the moment.

4.2 Check Boxes

Check boxes are commonly used to enable or disable different application properties, like showing a certain item or using a special mode. They store a boolean value and whenever clicked their value changes. In our implementation, we tried to mimic the look and feel of classic 2D check boxes as close as possible, while seamlessly integrating them into pie menus (Fig. 7a). Therefore they are realized as buttons, that are enriched with a box at the inner border. Depending on the value, this box either contains a cross or is empty. Apart from these changes, check boxes are identical to buttons.

4.3 Radio Buttons

Radio buttons are similar to check boxes, but have a slightly changed behavior. Like check boxes they store a boolean value, but the difference is that they work as a group in which exactly one element is checked at a time. That means, whenever an unchecked radio button gets checked all other radio buttons in the same group get unchecked.

Within our pie menu system, the visual representation of radio buttons is very similar to that of check boxes (Fig. 7b). As check boxes, they are based on buttons, but they have a circle at their inner border instead of a box. It indicates the current state of the radio button element by being empty when unchecked or containing a solid circle when checked. Regarding behavior, all radio buttons within one single menu are grouped per definition. If more than one group of radio buttons is needed in a menu hierarchy, these groups must be put into different sub menus.

4.4 Sliders

Adjusting scalar values is vital in a broad variety of applications, like setting minimum and maximum values for color maps or setting the day time for architectural walkthroughs. Therefore, we decided to add sliders to our pie menu system. They work very much like sliders known from classic 2D user interfaces.

In our implementation, sliders look like button elements, but have a slice-shaped axis inside, with the minimum at the inner and the maximum at the outer end. Additionally, a handle is present to indicate the currently selected value (Fig. 7c). Clicking in the area of the element triggers a change of the sliders value and also makes the handle follow the projection of the currently selected point onto the slider's axis. The slice-like shapes of the containing element and the slider axis are visual hints to where the minimum and maximum ends of the slider are.

An optional texture can be applied to the axis of the slider. This feature can be used, for example for displaying a color map to adjust the value of an iso-surface or for previewing color selections like shown in section 4.7 and figure 7f).

4.5 Color Pickers

Many VR applications have the need for selecting or adjusting colors. We offer a simple and quick way for color picking in our pie menu system by integrating a hue and lightness color picker (Fig. 7d). It displays a texture with hues on the angular axis and lightness values on the radial axis of a pie menu element. Colors can be selected by clicking or dragging to a certain location. To offer the possibility to select color from the full hue, saturation, and lightness (HSL) color space, the saturation value of the color picker can be set externally, for example by a slider element.

4.6 Color Map Editors

Color maps are used to colorize items like volume renderings or particle traces. To support their dynamic creation and adjustment, we implemented a color map editor that integrates with our extended pie menus as shown in figure 7e). As seen here, the color map editor has two modes: a passive mode (left) when being out of focus and an active mode (right) when being focused. The passive mode just displays the currently configured color map and the control points on top of it. The active version is used to edit the color map. Here, an arc is additionally shown at the outer border of the element that contains rectangles that correspond to a control point, each. The background of this arc is a fully opaque version of the color map. This helps in configuring mostly transparent color maps. To adjust the color map, control points can be clicked and dragged freely within the color map element, with the angular position denoting the scalar value while the radial position maps to the alpha value of the control point's color. Clicking a rectangle in the outer arc opens a color choosing menu to adjust the corresponding control point's color (this menu will be described in detail in the next section). New control points can be added by clicking at an empty position in the editor and control points can be deleted by dragging them to the dead zone.

4.7 Color Choosing Menu

A very good example of how individual elements can be combined to complex user interfaces is the color choosing menu (Fig. 7f). It is a predefined menu that can be used freely within the extended pie menu system, by adding it as sub menu. It consists of three sliders for setting red, green and blue (RGB) values, a slider for the alpha value and the saturation respectively and a color picker for picking a color in hue/lightness space. This way colors can be specified in HSL and RGB color spaces simultaneously. All sliders show a gradient texture that indicates the influence of a value change and gets updated instantly upon interaction. The dead zone of the color choosing menu shows the currently selected color and clicking it confirms selection. No other interaction closes the menu, neither dragging sliders, nor dragging in the color picker. At first this behavior seems to break with the concepts of automatic closing and aborting, but informal user tests in the development phase indicated that it did not cause usability issues. Additionally, the succeeding expert review did not show any potential issues with the color choosing menu.

4.8 Limitations

Due to a limited number of elements that can be efficiently used, creation of elements like list or combo boxes with many entries cannot be realized and also text entry elements can hardly be realized with our extended pie menus. In contrast to 2D user interfaces, UI elements cannot be freely arranged. This is not necessarily a disadvantage, because elements always get automatically arranged, which generates a harmonic look-and-feel. In response to these limitations, extended pie menus should not be seen as substitution to classic 2D user interfaces or systems that mimic their behavior in IVEs, but as an extension to context menus, which provide an easy way to configure VR applications in many, but not all, situations.

5 Testing the Design with an Expert Review

After finishing the implementation of the extensions, an expert evaluation in form of a cognitive walkthrough [17] was performed with five VR experts without domain-specific knowledge regarding the test application. They were told to assume the role of a user, in order to identify possible usability issues. We chose to evaluate the current state of the system with this method in order to be able to improve it by reacting on identified problems and by finding potential alternatives to certain implementation aspects. To perform the review we included our extended pie menu system in an existing application, which enables users to explore airplane noise emissions in the vicinity of airports. For this, visualization and auralization approaches are combined with various interaction techniques for navigation and data analysis. Context menus previously played a central role in granting access to the application's functionality, and several other UI elements, like independent slider widgets, were also used for configuration. All this was replaced by extended pie menus and all elements of the menu system are used in the application, thereby making it a suitable use case for their evaluation.

We prepared a standard scenario on how a user would possibly interact with the application and asked the experts to go through the scenario by successively performing defined tasks. Before starting, they were explicitly told that the extended pie menu system was object of the study and not the application it was integrated into. The study was conducted by a supervisor, who guided the experts through the scenario and an observer who transcribed the comments of the experts. Prior to the review, experts were informed of the anonymous and confidential use of their data and their right to quit the review at any time. All of the experts were right-handed and had normal or corrected to normal vision. They were told to think aloud about everything they were doing and additionally, questions were asked by the supervisor that sometimes led to discussions about certain menu elements. After finishing the walkthrough they were asked to fill out a final questionnaire where they had the possibility to leave additional comments.

5.1 Findings

Findings were gathered by evaluating transcripts of the conversation between the supervisor and the participating expert as well as the analzying comments that were given in the questionnaire. Results from both were grouped and will be presented in the remainder of this section. First we will present more general findings, before having a closer look at single menu elements.

General Comments All expert could easily use the features of the extended pie menus. In the test application the functions of menu opening and selections making were implemented by using different buttons in order to imitate left and right click usage of desktop systems. Experts suggested either to use one button for opening the menu and making selections or, when using different buttons, to make the opening button also close the menu.

Automatic Closing of the Menu Usually, pie menus are automatically closed after an entry has been selected. With regard to the newly introduced extensions, this behavior must be revised, because closing the menu upon each interaction could irritate the user. In the test application, we tried to implement an intelligent menu design, where we decided for each element if the menu closed upon interaction or not.

Even though this worked well in the sense that the menu mostly reacted as expected, distractions occurred in certain situations. To account for this, two solutions came up in the review. The first one is that automatic closing should only happen on button-like elements, namely the button, check box, and radio button elements, while the menu should stay open upon interaction with every other element. The second solution is to introduce a lock mode, to give the user the option to keep a menu opened when needed. Here, an additional element like a ring around the dead zone, is added to the menu, which enables the lock mode when selected. With lock mode enabled, the menu does not close automatically upon interaction and thus stays open when the

user decides she needs to perform successive interactions on the same or different menu elements.

Giving Hints on Being in the Menu We let the experts compare menus with and without elements that extend with the cursor. They overall liked the concept of extending elements, but ideas for two different refinements came up. The first was that an element would extend up to a certain radius before the pick ray would slip out of it and could thus be used for interaction with the scene. Though this idea is promising, it would remove the infinite target property of pie menus and thus could cause longer interaction times. Here, further research on the topic is needed. The second idea was to make the menu follow the pick ray, when a certain maximum radius for element extension is reached. This would maintain the infinite target property and additionally introduce a method to move the menu around in the scene, thus making it possible to avoid scene occlusions.

Buttons, Check Boxes, Radio Buttons and Color Pickers Regarding button elements the sole comment was, that an indicator whether a button triggers an action or opens a submenu should be introduced. Apart from this no issues could be identified with regard to the button element. Check boxes, radio buttons and color pickers reacted as expected by the experts and no problems regarding usability could be identified.

Sliders In some cases the accuracy of sliders was perceived to be too low, a fact that could be overcome by optimizing the radial increase of the slider's radius upon focus and by including jitter reduction. It was also mentioned that the current value of the slider as well as minimum and maximum values should be drawn in text form; at least when hovering over the slider element. This request is very challenging because of limited space within the menu and limited resolution of many VR devices.

Color Map Editors All experts stated that the color map editor was well-realized and easy to use. Some of the experts would have liked the value of the currently dragged control point to be displayed directly next to it, in order to avoid gazing between the caption and the control point. Another remark was that intuitive understanding upon first contact could be improved by displaying a trash can in the dead zone while dragging a control point, to give a hint that it can be deleted by dragging it there.

Color Choosing Menus All of the experts liked the color choosing menu and no interaction problems came up. One expert remarked that the menu was useful for expert users, but that a possibility to reduce complexity - by only presenting elements for adjusting the color in either RGB or HSL color space - could increase the usability for novice users without knowledge of color spaces.

6 Conclusion

We presented a formal evaluation of hierarchical pie menus in the context of IVEs and tracked 6-DoF input devices. For this, we implemented a pie menu system and evaluated it in a quantitative and qualitative within-subject user study. The study focused on different entry selection methods but also covered other aspects of pie menus. The pick-ray-based entry selection metaphor was found to work best and for the other considered aspects clear preferences could be derived as well.

Considering the findings of the pilot study, we implemented extended pie menus for IVEs that still contain the basic functionality of pie menus but are enriched with additional elements that are known from 2D user interfaces. Using the extended menus makes it possible to configure even complex VR applications.

Acknowledgments

The depicted research was funded by the German Research Foundation DFG as part of the Cluster of Excellence "Integrative Production Technology for High-Wage Countries".

We want to thank Marc Hassenzahl for providing the short version of the AttrakDiff2 questionnaire and Joachim Herber for writing the pilot study application.

References

[1] I. Assenmacher and T. Kuhlen. The ViSTA Virtual Reality Toolkit. In *Proceedings of the IEEE VR 2008 Workshop Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, pages 23–28. Shaker Verlag, 2008.

[2] D. Bowman, E. Kruijff, J. LaViola, and I. Poupyrev. *3D User Interfaces: Theory and Practice*. Addison-Wesley Professional, 2004.

[3] D. Bowman and C. Wingrave. Design and Evaluation of Menu Systems for Immersive Virtual Environments. In *Proceedings of the Virtual Reality 2001 Conference*, page 149. IEEE Computer Society, 2001.

[4] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. An Empirical Comparison of Pie vs. Linear Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 95–100. ACM, 1988.

[5] X. Cao and R. Balakrishnan. VisionWand: Interaction Techniques for Large Displays Using a Passive Wand Tracked in 3D. In *Proceedings of the 16th annual ACM symposium on User Interface Software and Technology*, pages 173–182. ACM, 2003.

[6] K. Das and C. Borst. An Evaluation of Menu Properties and Pointing Techniques in a Projection-based VR Environment. In *IEEE Symposium on 3D User Interfaces*, pages 47–50. IEEE, 2010.

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

[8] J. Grosjean, J.-M. Burkhardt, S. Coquillart, and P. Richard. Evaluation of the Command and Control Cube. In *Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces*, pages 473–478. IEEE, 2002.

[9] M. Hassenzahl, M. Burmester, and F. Koller. AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualitt. In *Mensch & Computer 2003: Interaktion in Bewegung*. Teubner, 2003.

[10] D. Hopkins. The Design and Implementation of Pie Menus. *Dr. Dobb's Journal*, 16(12):16–26, 1991.

[11] R. Komerska and C. Ware. A Study of Haptic Linear and Pie Menus in a 3D Fish Tank VR Environment. In *Proceedings of the 12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 224–231. IEEE Computer Society, 2004.

[12] G. Kurtenbach, A. Sellen, and W. Buxton. An Empirical Evaluation of Some Articulatory and Cognitive Aspects of Marking Menus. *International Journal of Human-Computer Interaction*, 8(1):1–23, 1993.

[13] B. Laugwitz, T. Held, and M. Schrepp. Construction and Evaluation of a User Experience Questionnaire. In *Proceedings of the 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for Education and Work*, pages 63–76. Springer-Verlag, 2008.

[14] J. Lewis. IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction*, 7(1):57–78, 1995.

[15] W. Mackay. Which Interaction Technique Works When? Floating Palettes, Marking Menus and Toolglasses Support Different Task Strategies. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 203–208. ACM, 2002.

[16] T. Ni, R. McMahan, and D. Bowman. Tech-note: rapMenu: Remote Menu Selection Using Freehand Gestural Input. In *IEEE Symposium on 3D User Interfaces*, pages 55–58. IEEE, 2008.

[17] P. Polson, C. Lewis, J. Rieman, and C. Wharton. Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies*, 36(5):741–773, 1992.

[18] G. Ren and E. O'Neil. 3D Marking Menu Selection with Freehand Gestures. In *IEEE Symposium on 3D User Interfaces*, pages 61–68. IEEE, 2012.

[19] M. Tapia and G. Kurtenbach. Some Design Refinements and Principles on the Appearance and Behavior of Marking Menus. In *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*, pages 189–195. ACM, 1995.

[20] S. Zhao and R. Balakrishnan. Simple vs. Compound Mark Hierarchical Marking Menus. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, pages 33–42. ACM, 2004.