The Effect of Dwell Time on Swipe-based Pie-Menu Navigation Efficiency

https://doi.org/10.3991/ijoe.v16i15.17763

Alen Salkanovic, Matija Stojkovic, Sandi Ljubic (✉)
University of Rijeka, Rijeka, Croatia
sandi.ljubic@riteh.hr

Abstract—In this paper we present and analyze pie-menu implementation for touchscreen mobile devices which supports swipe-based navigation through deep hierarchical menu configurations. We propose a solution that synergistically encompasses important design features for a given context: radial menu visualization, semi-transparency, manual repositioning, occlusion awareness, and marking menu concept. From the menu navigation efficiency standpoint, dwell time represents one of the most influential factors in such a design. Reducing the time needed to automatically activate submenus could potentially provide faster navigation through the pie-menu hierarchy; however, short dwell times could make swipe-based interaction more error-prone, as unintentional item selections are then more probable. Therefore, we specifically investigate the effect of dwell time on navigation efficiency when using the proposed solution. A remotely administered experiment involving 36 participants was carried out in order to empirically evaluate different dwell time values. The interaction style, i.e. the way of holding a smartphone device, was also taken into account as an independent variable. We present the results of this empirical research, additionally tackling the interaction workload, usability attributes, and overall design considerations of the proposed solution. According to the obtained results, we argue that highly adaptable pie-menus could provide interaction benefits in the mobile touchscreen domain.

Keywords—Pie menu, swipe-based interaction, dwell time, touchscreen mobile devices, remote experiment

1 Introduction

The concept of hierarchical menus originates from software applications implemented for desktop operating systems. Hierarchically organized menus allow navigation through different levels of the underlying menu structure. A hierarchical layout supports an increasing amount of items being presented, as menus are stepped through (e.g. from options to sub-options, or from categories to sub-categories). Common instances of hierarchical design are typically found in application menu bars with the corresponding pull-downs, and in websites within their navigation link structure. Various menu implementations are usually designed and optimized for mouse and/or keyboard interaction. However, as contemporary mobile devices became powerful computing platforms able to run mobile versions of popular desktop applications, multi-level menu implementations turned out to be a necessity in the mobile touchscreen domain as well. In the Human-Computer Interaction (HCI) field, designing a multi-level menu for mobile devices continues to pose a challenge, and can be considered as a specific effort in broader research on navigation *within* information systems [1]. When compared to desktop screens, mobile touchscreens are considerably smaller. Consequently, the limited screen size cannot easily accommodate many different subcategories in the mobile menu navigation.

Touchscreen smartphone devices face specific issues in menu interaction, such as occlusion, delay-based context menu activation, insufficient accuracy, and lack of shortcuts [2]. Present-day menus for mobile devices are designed in diverse shapes and forms, here including sequential menus (like navigation drawer), floating action buttons (FABs), accordions (e.g. hamburger menu), section menus (tabs), and bottom bar menus. Generally, these mobile menu designs are bounded to the underlying application. Menu items from a single submenu are typically displayed on the screen, thus leaving the content of the application predominantly hidden. Furthermore, activation points are usually predefined, often located on the top or bottom of the screen. In all cases, a trade-off persists between menu sub-navigation efficiency and screen real-estate occupation. A range of benefits and drawbacks can be found for all menu formats, and, in that sense, implementation of a particular menu design may or may not be appropriate for a given application context. Finding a suitable way for providing fast and easy navigation in mobile applications is therefore very important, as it directly leads to a higher level of mobile usability [3].

In this paper we investigate alternative menu design for the mobile domain, combining some of the previously introduced concepts (e.g. radial-shaped menus, marking menus) and touchscreen interaction modalities (e.g. swipe gestures instead of tapping sequence). Our solution refers to the circular visualization of the context menu, also known as the *Pie-menu*. The proposed implementation is completely detached from the mobile application itself and, thus, acts as a floating widget service. This feature provides the possibility to use the same menu structure in different target applications.

We present a pie-menu wherein continuous swipe gesture is utilized in order to navigate across the (potentially deep) hierarchical menu configurations. By involving gesture-based interaction, we purposely avoided the need for recurrent tapping. Our design additionally supports semi-transparent visualization and manual repositioning of the menu pivotal position – according to the users' individual preferences. These features contribute to better screen utilization, as more space and visibility can be dedicated to the underlying application content.

To invoke a new submenu within the proposed design, a particular swipe gesture should involve the touch-dwelling concept. Namely, a subsequent menu level can be invoked by retaining the finger over the specific menu item for a given period of time (*dwell time*). While three specific dwell time values are offered as default options, the custom value can be defined as well. Our main research goal is to investigate how different dwell times affect the navigation efficiency of the proposed menu solution. Therefore, the execution of swipe-based pie-menu navigation tasks was empirically analyzed, targeting altered dwell times, as well as two contexts of use: single-handed and cradling (the case wherein one hand is holding the device, while the other – usually the dominant one – is performing swipe gestures).

2 Related Work

Pie menu, or radial menu, represents a design wherein the menu items are visualized along the circumference of a circle at equal radial distances from the center. In general, circle-based formats can gain over traditional linear menus by reducing target seek time, lowering error rates, and minimizing the drift distance after target selection. A number of empirical comparisons between linear and pie menus have been made in the desktop domain. One of the early studies showed that interacting with pie menus can be about 15% faster than with linear menus, with a much lower error rate [4]. Following the technology improvement and the expansion of available interaction modalities, pie menus found their way from desktop systems to touchscreen devices, regardless of their form factor. Namely, radial menu implementations can be successfully applied on large tabletops, as well as on contemporary tablets and smartphones.

Floating Pie Menus [5] combine the dynamic aspect of pie menus and the persistence of floating palettes. They are transparent and can be repositioned by the user, thus allowing to see the underlying application content while accessing the menu more easily. Two additional features are provided within this solution: lock-mode and dynamic target. While the lock-mode enables the menu to stay open at a certain hierarchy level, the dynamic target allows the user to attach a previously locked menu to another object, thus providing the possibility to repetitively apply menu options in different contexts without the menu closing in-between.

Occlusion-aware Menu [6] is a radial menu design proposed for multi-user interaction on digital tabletops. The menu has an open side where no items are located, which avoids occlusions caused by the user's hand. This adaptive menu placement in addition takes the user's handedness into account. Multi-user interaction scenarios are inherently supported by multi-touch and pen tracking capabilities. *Stacked Half-Pie menu* [7] represents another solution specifically designed for the tabletop's domain, but in this case, a semicircular design was utilized instead. It allows visualization of an unlimited number of hierarchical menu items, as well as touch-based navigation and item selection. However, the menu position is fixed at the bottom of the screen and can't be moved, which can be considered as the major drawback.

ArchMenu and ThumbMenu [8] are designs adapted for interaction with small handheld touchscreen devices in mobility conditions. They apply semi-circular layouts wherein all menu items are easily accessed with the simple thumb movements. Nevertheless, as the layout shape increases with every new triggered submenu, there is a significant constraint on the number of items that can be displayed on the small screen.

One way to support exploration and navigation in data hierarchies on mobile devices is to use *Wavelet Menus* [9]. This solution represents an adaptation of the Wave menu, which is commonly used for browsing multimedia content on iPhone mobile devices. Efficient menu navigation is supported by pre-visualization, a special feature which allows user to see the submenu corresponding to a certain menu option before it is actually activated. Both circular and linear representations are provided, the latter being used when there are many items needed to be displayed.

The Swiss Army Menu (SAM) [10] is another solution that utilizes a radial menu design for small screen interfaces. Its main advantage is hierarchy navigation based on small thumb movements. A pointer controlled by the finger is used to select a certain menu item in order to avoid the occlusion problem. Four different types of menu items are allowed in this design, which correspond to the typical usage of buttons, checkboxes, sliders, and scrollbars. Similar to the Wavelet Menu, SAM also implements the preview feature which improves submenu navigation efficiency. Although it is possible to activate the SAM from any location on the screen, it always appears at the same predefined position.

Some of the existing menu solutions for touchscreen devices successfully apply the well-known marking menu concept. This concept involves a specific interaction design wherein items are selected using a straight stroke gesture. In general, it enables an easy transition from novice to expert user, according to the two different modes it imposes [11]. In the novice mode, the menu is visible to the users while making the selections, and it disappears from the screen once a menu item is selected. However, in the expert mode, the menu is completely hidden from the user. Therefore, previously memorized stroke gestures can be made without the need for the menu to appear on the screen. *X-O Arch Menu* [12] and *M3 Gesture Menu* [13] can be singled out as representative solutions that successfully combine the (semi)circular menu visualization and the marking menu concept.

X-O Arch Menu allows the users to annotate precise positions in large information spaces and is optimized for use with limited screen space. There are two interaction modes available: novice and expert. While both provide a drag-and-drop option for menu repositioning, the expert mode additionally enables the use of swipe gestures to reduce the time it takes to get deeper in the menu hierarchy. In order to save screen real-estate, the submenu items are always displayed in place of the previous menu level. *M3 Gesture Menu* is a result of more recent work in which the traditional marking menu concept is applied for mobile touchscreens. It utilizes a persistent screen space and contains menu items arranged in a grid instead of a circular layout. Predefined swipe gestures, based on related item locations inside the menu, are used for selection, and when a certain option is activated, the same space is being occupied by its submenu items. A shape matching algorithm provides a robust recognition of swipe gestures.

Pie menus have also been considered for in-vehicle information systems (IVIS). A *pie Touch* [14] menu is proposed so as to reduce the drivers' attention needed to operate such systems and ensure road safety. The adaptation of pie menus, serving as a gesture visualization, reduces the user's cognitive load and allows to achieve an almost blind interaction with the IVIS after just a couple of repetitions. Using the proposed solution, navigation through lists and context menus can be performed significantly faster than with standard point-and-touch systems.

Further use of radial designs can be found in gaze-based and multimodal (gaze and speech) menu selection. In [15], the authors demonstrated the benefits of semi-radial layouts over the linear and full-circle ones in the specific context of gaze-based and multimodal menu interaction. Linear menus are densely spaced and require complex tunnel-steering, which leads to errors, time waste, cognitive load, and uncomfortable use. On the other hand, the major drawback of the full-circle menu, according to the test subjects, is its widespread, ungrouped, and therefore confusing arrangement of menu items.

From the abovementioned menu solutions, several key design features can be identified: visualization type, transparency, repositioning, occlusion awareness, and marking menu concept. None of the existing menu implementations incorporate all the key features for a target domain. In this sense, the proposed solution fills the gap, as we tried to address all important design attributes. Table 1 summarizes related work overview, emphasizing the adoption of different design features in various menu implementations.

Table 1. Design features incorporated in different menu solutions

Menu solution	Target domain	Visualization	Transparency	Repositioning	Occlusion awareness	Marking menu
Floating Pie Menus [5]	Desktop	Radial, in-creasing	Yes	Yes	No	No
Occl - aware Menu [6]	Tabletops	Radial	No	No	Yes	No
Stacked Half-Pie [7]	Tabletops	Semi-radial, increasing	No	No	No	No
Arch Menu & Thumb-Menu [8]	Mobile	Semi-radial, increasing	No	No	Yes	No
Wavelet Menus [9]	Mobile	Linear / Radial	No	No	No	No
Swiss Army Menu [10]	Mobile	Radial	No	No	Yes	No
X-O Arch [12]	Mobile	Semi-radial	No	Yes	Yes	Yes
M3 Gesture Menu [13]	Mobile	Grid	Yes	No	No	Yes
Pie Touch [14]	IVIS	Semi-radial	No	No	No	No
Swipe-based floating pie-menu (this)	Mobile	Radial	Yes	Yes	Yes	Yes

3 Floating Pie-Menu: Implementation and Interaction Design

The pie-menu proposed in this paper allows swipe-based navigation through deep hierarchical menu configurations applied within mobile applications. It is implemented as a circular-shaped floating widget for Android-running mobile devices. Design choices were motivated by the aforementioned related work, as well as by new ideas about possible enhancements in menu navigation on touchscreen mobile devices. While modeling the menu widget and the corresponding interaction, we have utilized some of the previously introduced concepts. Namely, our solution is employing consistent radial layout which does not increase in size and allows swipe gestures (for commonly used menu options) to be easily memorized and subsequently executed faster. As such, the provided design also enables a smooth transition from novice to expert user.

Our pie-menu initially acts similar to the floating action button (FAB), placed on top of mobile applications. Being implemented as a service, the menu's UI is completely independent of the underlying application's layout. Instead of the usual navigation model wherein the typical menu widget occupies the entire screen or significant portion of the screen area, the proposed pie-menu supports better visibility of the application content. Its semi-transparency feature furthermore helps in this matter. Additionally, in order to minimize the occlusion effect, all menu items are being dynamically organized upon every sub-menu invocation. Specifically, menu item arrangement on a particular level is determined with respect to the current finger position on the screen, with the main goal being to place menu options on the positions not covered by the finger itself. The typical interaction pattern utilized in the proposed pie-menu is visualized in Fig. 1, and explained in detail in the following text.

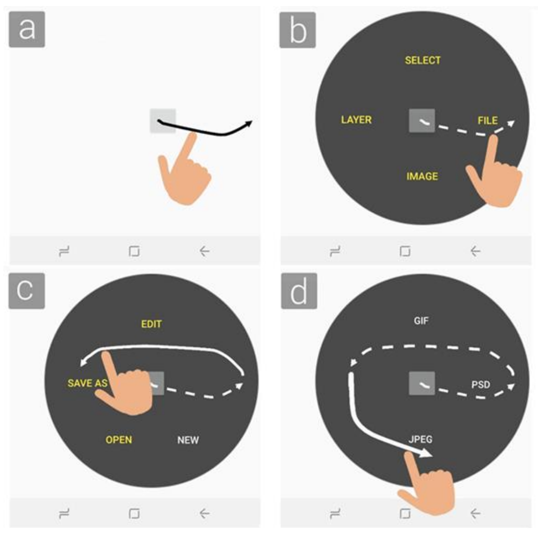

Fig. 1. Typical interaction pattern within the proposed pie-menu solution (a→b→c→d)

Two states of operation are enabled when the pie-menu is minimized and visualized as FAB only. Namely, the menu activation button can be either *active* or *locked*. When the FAB is in its *active* state, menu navigation is completely disabled, and the button is draggable. Hence, the *active* state allows the user to change the menu pivotal position, with respect to current needs and individual preferences. When the menu activation button is positioned to the desired location on the screen, it has to be *locked* in order to enable swipe gestures for hierarchical navigation. Switching between *active* and *locked* modes is enabled via long tap (i.e. touch-and-hold gesture).

Visualization of the menu starts in a previously customized pivotal position. Placing the finger on FAB in a *locked* state, and subsequently starting a swipe gesture in any direction (Fig. 1a), creates a circular container around the starting point, with all items from the top-level menu (Fig. 1b). To ensure that all menu items stay within the accessible screen area, changing the menu location is programmatically restricted. Consequently, the menu widget itself cannot exceed the screen boundaries, nor it can be furthermore expanded. Instead of using repetitive tap-based selections, menu hierarchy navigation, including the final item selection, is performed with a single continuous finger movement across different submenus. In order to select/activate a particular menu option, the finger should be dragged to the specific menu item (e.g. FILE in Fig. 1b). Highlighted items (yellow) indicate that there are additional submenus available. If the finger is retained over the highlighted item for a certain period of time (i.e. dwell time), all present menu options are replaced with the new ones from the corresponding submenu (Fig. 1c). The occlusion problem is tackled at this step, as new item arrangement is calculated in a way so as to avoid locations within the menu widget which are most probably covered by the current finger position. This navigation process can be repeated as long as there are options accessible in the hierarchical menu configuration. In case of an unintentional navigation error, the user can return to the previous submenus by hovering (touch-dwelling) over the menu activation button (in the center of the widget). The final selection is invoked by ending the swipe gesture once the finger is dragged across (or passed through) the particular menu item (e.g. JPEG in Fig. 1d). Once this occurs, the pie-menu is automatically minimized to its FAB form.

We believe the described approach provides several interaction benefits: (i) saving valuable space on small touchscreens, (ii) allowing hierarchical navigation within the same application activity, (iii) item selection by making use of a single swipe gesture, and (iv) retaining the visibility of application content while handling the menu at the same time. Similar to the marking menu concept, the provided solution could provide an easy transition from beginner to expert user. The effect of enhancing visibility, by involving a semi-transparent floating widget on top of the application content, is demonstrated in Fig. 2.

Adaptable features of the provided solution include menu repositioning (which can become especially convenient while changing the way of holding mobile device) and using different dwell times for swipe-based menu navigation. Our implementation utilizes three specific and predefined dwell time values (250 ms, 500 ms, and 1000 ms). However, an arbitrary custom dwell time value may also be defined according to the user's preference. In the following empirical evaluation, we investigated how exactly users utilize available customization options. Along with checking the preferred menu positions in different contexts of use, we specifically focused on the appropriate dwell time which can guarantee a higher level of menu navigation efficiency.

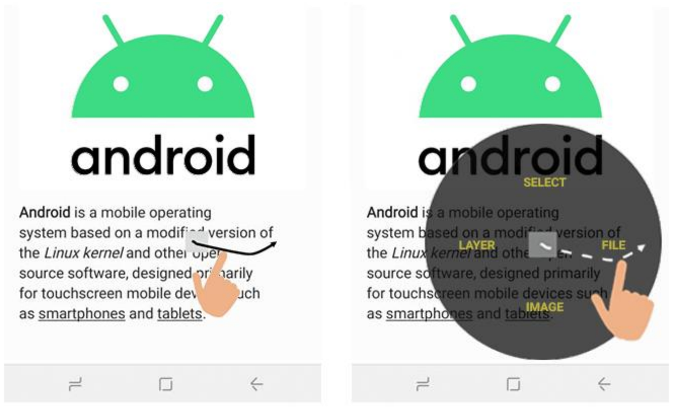

Fig. 2. Enhancing the visibility of underlying application content by utilizing semi-transparent floating pie-menu widget (instead of typical full-screen linear-based menu)

4 Empirical Evaluation

In this section, we provide information about the conducted experiment and address the obtained results. Due to the COVID-19 pandemic, and the related restrictions at the time of our research activities, recruiting the participants to carry out a fully controlled experiment *in-situ* was unfeasible. Consequently, the empirical evaluation of the implemented pie-menu solution occurred in a remotely administered fashion. The subjects were given detailed instructions on how to setup the mobile application (with pie-menu service) on their own smartphone devices. In addition, they were instructed, both by digital documents and video tutorials, how to execute menu navigation tasks in different contexts of use. The complete experiment procedure, including consent form signing, providing step-by-step guidelines, collecting application logs, as well as filling out post-study questionnaires, was managed online.

4.1 Participants, apparatus, and the procedure

Thirty-six participants have been involved in the empirical research (6 females and 30 males), their age ranging from 20 to 41 with an average of 25.3 years (SD = 4.6). In the pre-study online questionnaire, subjects have reported their smartphone models and corresponding screen sizes. Additionally, users were asked to specify their dominant hand (four of them were left-handed), as well as their preferred hands posture while holding a smartphone device (depicted in Fig. 3).

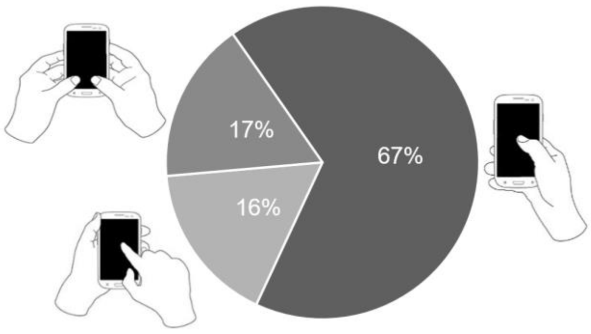

Fig. 3. The majority of the recruited participants generally prefer one-thumb single-handed interaction with a smartphone device

Since our pie-menu solution has been implemented for Android operating system, all participants used Android smartphones running OS version 5.1 (Lollipop) or newer, with display size (screen diagonal) ranging from 4.6 up to 6.7 inches. In order to get familiar with the menu widget and the corresponding interaction, subjects were engaged in a short practice session. At this point, no data was collected whatsoever.

Besides the general instructions sent via text manuals, the participants were required to watch a video demonstrating how to utilize the swipe-based menu navigation, as well as how to appropriately hold the smartphone device in different contexts of use (single-handedly using thumb, and cradling with forefinger).

In the actual experiment, users were requested to complete a total of 400 menu navigation tasks arranged in 40 cycles. Here we considered two independent variables: interaction style and dwell time. Regarding the interaction style, we are referring to the way of holding the smartphone device while performing swipe-based pie-menu navigation. Specifically, we were interested whether the particular hand posture (single-handed vs cradling) makes a difference in executing menu navigation tasks. Different dwell times were also utilized across the given navigation tasks. While three specific values have been predefined (250 ms, 500 ms, and 1000 ms), participants were additionally asked (at a given stage of testing) to apply individually preferred values for each interaction style. This way we wanted users to self-assess the dwell time value they considered to be the most suitable for efficient menu navigation. The main objective was to investigate how exactly dwell time affects the swipe-based pie-menu navigation efficiency.

Experiment tasks have been generated based on the available menu configurations with distinctive hierarchy structures. More precisely, five menu configurations with different breadths and depths were specified. Intending to provide users with a fairly familiar menu navigation tasks, we designed configurations similar to the ones that can be found in popular text editors, photo editors, and integrated development environments. The related tasks included both shorter and longer navigation paths, while the maximum depth of the menus was set to six submenu levels. The single task cycle encompassed 10 different task instances for a particular menu configuration, certain interaction style, and currently active dwell time. Therefore, a total number of

400 menu navigation tasks can be structured as follows: 2 interaction styles × 4 dwell times × 5 menu configurations × 10 navigation paths. The order of the menu navigation tasks was properly counterbalanced (enforced by the testing application), in order to compensate for possible learning effects. A repeated measures (i.e. within-subjects) experiment design was utilized. The single task was displayed on the smartphone screen as the required navigation route (e.g. File → Save as → JPEG).

The testing application has been developed for logging all relevant interaction events, while running the pie-menu service at the same time. It utilizes configuration files, containing a human-readable description of the menu structures, from the internal memory of a mobile device. From these configuration files, menu navigation tasks can be specified either manually or automatically. Consequently, we provided support for tweaking the experiment settings in an easy way. In this manner, introducing a new menu hierarchy and new tasks in the experiment requires preparing the corresponding configuration file and utilizing the available task generator. Once generated, all navigation tasks are stored in the SQLite database and ready to be presented in the counterbalanced order.

The time taken to complete the required menu navigation task is considered to be the interval between a first touch inside the FAB (in its *locked* state) and a finger lift-off event which ends the associated swipe gesture. This task execution time is measured by the application itself, by making use of a built-in monotonic clock which is tolerant to power saving modes and is the recommended basis for general-purpose interval timing on Android devices. Participants were instructed to turn off all network-based services on their mobile devices during the experiment.

If the testing application is closed during the particular task cycle, the corresponding data will not be saved to the output log file, and the same cycle will be reloaded once the application is restarted. Hence, the application itself ensures the proper conducting of the experiment procedure, by managing the current state of the testing process. Consequently, the participants were allowed to take breaks only when a single 10-task cycle was performed completely. If target selection was not successfully accomplished, the corresponding task was not repeated, and error details were logged along the task execution time.

When changing between interaction styles (e.g. from single-handed to cradling or vice-versa), the subjects were allowed to select a custom menu pivotal position. To ensure that all menu items remain visible inside the accessible screen region, menu repositioning was programmatically constrained. In other words, pie-menu is visualized either in the user's preferred position or in the closest point that allows full widget view. Pie-menu itself does not change in size, as submenu levels do not require additional container space.

After the participants completed all 400 menu navigation tasks using both interaction styles and four dwell times, they were required to complete post-study questionnaires made available via Google Forms. The first questionnaire was based on the rating part of the NASA-TLX (Raw-TLX format). Individual opinions about perceived workload had to be estimated on a 21-point Likert scales for five factors: mental demand, physical demand, frustration, performance, and effort. Participants were thus asked to comparatively assess single-handed and two-handed pie-menu navigation. Finally, users' impressions about the proposed solution's usability features and general design choices were collected using the concluding surveys based on 7-point Likert scales.

4.2 Results and discussion

After the experiment, each participant sent us the log file, which was generated and stored internally on the target mobile device. 36 participants accomplished altogether 14400 menu item selections. By averaging data from the obtained log files, a total of 288 menu navigation performance records were acquired: 36 participants × 2 interaction styles × 4 dwell times. Fig. 4 depicts the average times taken to complete the required pie-menu navigation tasks by utilizing provided swipe-based interaction modality.

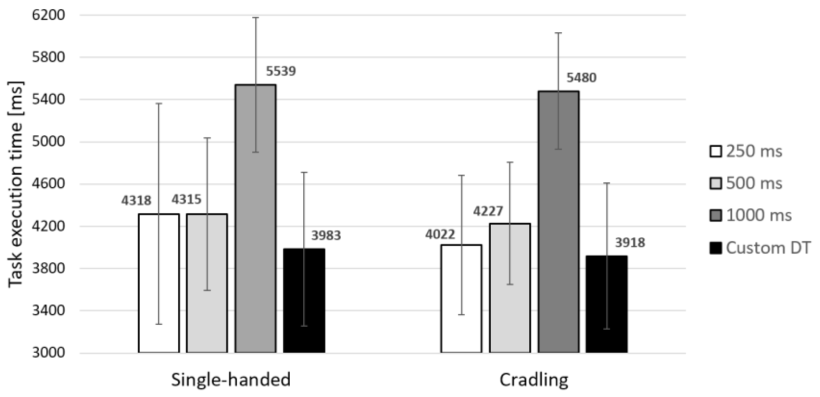

Fig. 4. Pie-menu navigation times (mean values and standard deviations), obtained both single-handedly and via cradling for different dwell time values

In order to analyze the obtained data, a two-way repeated-measures ANOVA was used, with *Dwell time* (250 ms, 500 ms, 1000 ms, custom) and *Style* (single-handed, cradling) being the within-subjects factors. The analysis revealed a significant effect of *Dwell time* on menu navigation time ($F_{3, 105} = 198.666$, $p < .001$). On the other hand, the effect of *Style* (i.e. the way of holding a mobile device) was not found statistically significant ($F_{1, 35} = 1.893$, $p = 0.178$).

As for the pairwise comparisons, the differences in menu navigation efficiency caused by different dwell time values showed to be as expected. Namely, when the longest dwell time was used (1000 ms), it yielded the slowest menu navigation:

- T_{1000} vs T_{250}: (5.50 ± 0.09 s) vs. (4.17 ± 0.126 s), $p < .001$
- T_{1000} vs T_{500}: (5.50 ± 0.09 s) vs. (4.27 ± 0.096 s), $p < .001$
- T_{1000} vs T_{CUSTOM}: (5.50 ± 0.09 s) vs. (3.95 ± 0.105 s), $p < .001$

The mentioned outcome seems very reasonable, as continuous waiting for a whole second before each submenu invocation certainly contributes to the longer menu traversing. On the contrary, menu navigation was convincingly the most efficient when custom dwell time was applied:

- T_{CUSTOM} vs T_{250}: (3.95 ± 0.105 s) vs. (4.17 ± 0.126 s), $p < .05$
- T_{CUSTOM} vs T_{500}: (3.95 ± 0.105 s) vs. (4.27 ± 0.096 s), $p < .001$
- T_{CUSTOM} vs T_{1000}: (3.95 ± 0.105 s) vs. (5.50 ± 0.09 s), $p < .001$

The observed differences, although statistically significant, may at first seem negligible within the real-life usage. However, it has to be noted that menu navigation tasks in mobile applications are performed very often, so the provided time gain can easily become considerable.

As shown in Fig. 5, when asked to set up and apply preferred dwell time, the majority of participants had chosen values just in between the default provided ones (250 ms and 500 ms), both for single-handed and cradling-based usage. This reaffirmed our choice not to set the default lower threshold below 250 ms, as considerably short dwell times can make the interaction with the proposed pie-menu quite difficult.

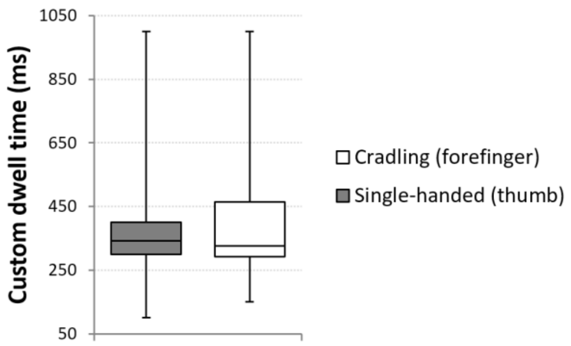

Fig. 5. Custom dwell time values chosen by the experiment participants (box plots indicate the minimum, the maximum, the sample median, and the first and third quartiles)

When it comes to the errors while performing swipe-based pie-menu navigation, log records revealed that most of the mistakes were made on the last submenu level, when the non-target item was unintentionally selected by ending the swipe gesture on the wrong place. Nevertheless, error rates were rather low in all experiment conditions. Fig. 6 depicts error rates obtained within menu navigation tasks, executed both single-handedly and via cradling, while utilizing different dwell time values.

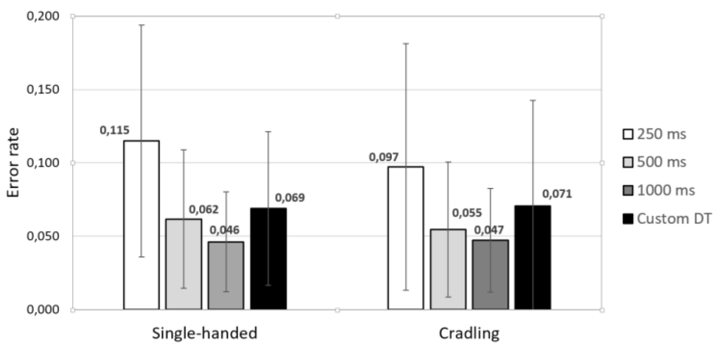

Fig. 6. Error rates within pie-menu navigation tasks (mean values and standard deviations), obtained both single-handedly and via cradling for different dwell time values

Just like with the menu navigation time, two-way repeated-measures ANOVA revealed a significant effect of *Dwell time* on error rate (ER) as well ($F_{1.949,\ 68.230}$ = 15.658, p < .001). The Greenhouse–Geisser correction (ε = 0.65) for the violation of sphericity was applied in this test instance. Contrariwise, the effect of *Style* was not found statistically significant ($F_{1,\ 35}$ = 0.929, p = 0.342).

As expected, pairwise comparison tests (utilizing Bonferroni adjustment) proved that participants made significantly more selection errors when the shortest default dwell time was applied (250 ms):

- ER_{250} vs ER_{500}: (10.6 ± 1.3 %) vs. (5.8 ± 0.6 %), p = 0.001
- ER_{250} vs ER_{1000}: (10.6 ± 1.3 %) vs. (4.7 ± 0.5 %), p < .001
- ER_{250} vs ER_{CUSTOM}: (10.6 ± 1.3 %) vs. (7.0 ± 0.9 %), p < .05

When custom dwell time value was utilized, participants made significantly more navigation errors only when compared with the longest default value (1000 ms):

- ER_{CUSTOM} vs ER_{250}: (7.0 ± 0.9 %) vs. (10.6 ± 1.3 %), p < .05
- ER_{CUSTOM} vs ER_{500}: (7.0 ± 0.9 %) vs. (5.8 ± 0.6 %), p = 0.719 (ns)
- ER_{CUSTOM} vs ER_{1000}: (7.0 ± 0.9 %) vs. (4.7 ± 0.5 %), p < .05

This brings to a conclusion: when compared to “direct competitors” (250 ms and 500 ms values), custom dwell time significantly improves menu navigation efficiency, without notable degradation in selection accuracy.

The statistical analysis also revealed that the *Style* factor does not impact the menu navigation efficiency whatsoever. Apparently, both the menu navigation time and the error rate are not affected by the way of holding a mobile device. These quantitative indicators are additionally supported by the obtained results from the post-experiment TLX questionnaire, which are shown in Fig. 7.

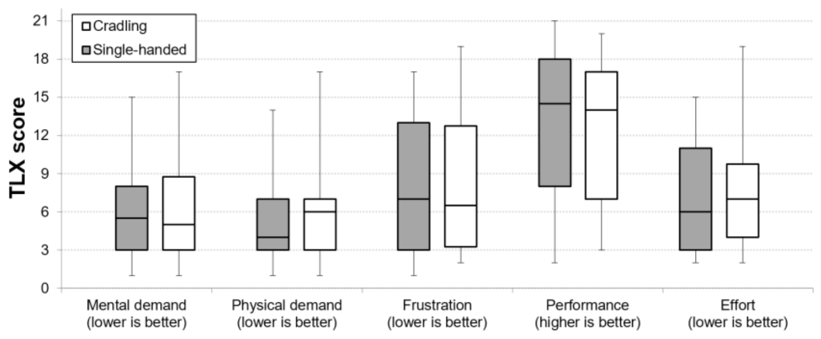

Fig. 7. Users' comparative ratings on perceived workload between different interaction styles (single-handed, cradling) for the proposed pie-menu solution

Individual opinions about perceived workload had to be estimated on a 21-point Likert scales for five TLX factors, in order to qualitatively assess both single-handed and two-handed pie-menu navigation. The Wilcoxon signed-rank tests were subsequently used to statistically analyze obtained TLX scores for each considered factor: mental and physical demands, level of frustration, perceived performance, and overall effort invested in experiment tasks. The tests revealed no significant differences in perceived workload between single-handed and two-handed menu navigation. Indeed, as interaction style showed to be a non-significant effect on pie-menu navigation efficiency (from both the speed and error rate standpoint), it comes by no surprise that participants do not have a clear preference towards a particular way of holding their smartphone while interacting with the proposed pie-menu solution. Interestingly, custom dwell time values that participants selected for the single-handed context of use (mean: 386.83 ms) do not much differ from the custom values chosen for cradling interaction (mean: 382.58 ms). This may further explain the lack of significant differences in menu navigation efficiency with respect to the observed interaction styles.

Usability attributes of the pie-menu are inspected using simple 7-point Likert scales. Namely, participants had to rate pie-menu implementation against the utility (its usefulness), ease of use, learnability, overall satisfaction, and engagement (willingness to recommend the proposed solution to other users). The obtained results are presented in Fig. 8.

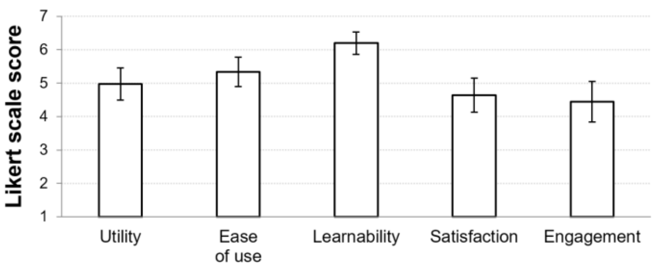

Fig. 8. Usability attributes of the proposed menu solution (mean values and 95% confidence intervals are presented)

As a higher score on the Likert scale means a higher level of the corresponding usability attribute, it can be concluded that pie-menu's overall usability is perceived as quite good. Users especially find the pie-menu easy to learn (mean rating: 6.19), as well as easy to use (mean rating: 5.33), which means that proposed interaction design, based on utilizing swipe gestures and dwelling, comes as both straightforward and natural concept.

Finally, the last part of the post-study survey tackled participants' attitudes towards the main design features incorporated within the proposed solution. Specifically, we wanted to investigate the extent to which users think the pie-menu is appropriate for deep hierarchical structures, and whether swipe-based interaction is indeed a good choice for performing menu navigation. Furthermore, users reported their opinions about the pie-menu adaptability, i.e. the possibilities to customize menu pivotal position and dwell time according to the individual preferences. Considerations on semi-transparency visualization and occlusion-awareness were also a part of this concluding questionnaire. All the mentioned design features were assessed using 7-point Likert scales, and Fig. 9 presents the corresponding outcomes.

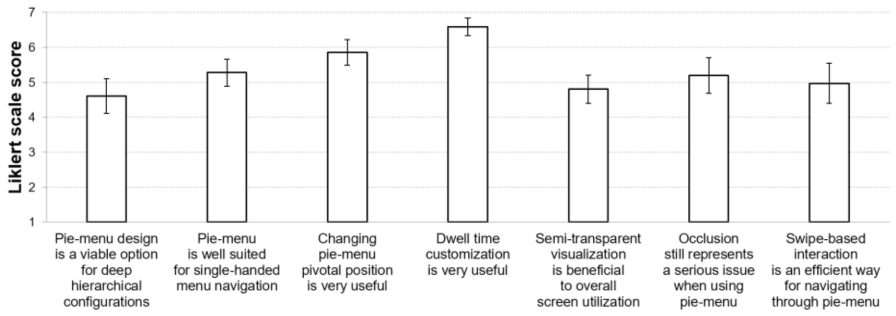

Fig. 9. Participants' attitudes towards the pie-menu characteristics and design features (mean values and 95% confidence intervals are presented)

It can be seen that all design choices are well accepted among participants, except for occlusion tackling which obviously requires certain improvements. Users agree that occlusion still represents a burden while navigating within the pie-menu (mean value: 5.19), although actions have been taken to mitigate the effects of this problem. In this context, extending the area of the pie-menu widget wherein new submenu items will not be presented (i.e. area under the finger position) seems like a reasonable enhancement. For example, in upcoming implementations we could restrict the available space up to 75%, assuming that a quarter of the pie-menu widget will be visually covered with a finger.

As expected, users particularly favor the provided adaptability features. The usefulness of both the dwell time customization (mean value: 6.58) and the menu repositioning (mean value: 5.86) was rated with the highest scores. This is an understandable outcome, as custom dwell time proved to be a "game-changer" when it comes to menu navigation efficiency. As for menu repositioning, the participants were allowed to change the pie-menu pivotal position two times within the experiment: before starting the single-handed task sequence, and prior to the cradling-based one. The distributions of the chosen screen locations are illustrated in Fig. 10.

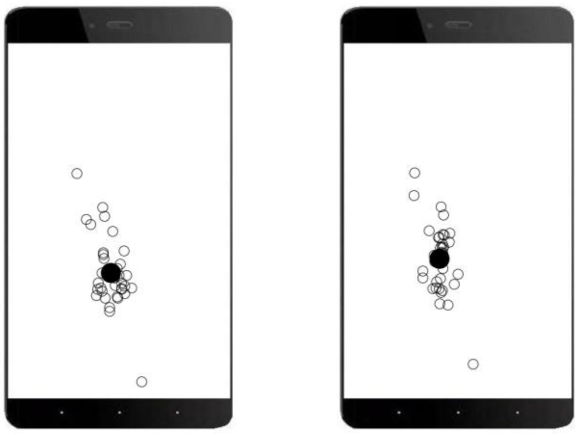

Fig. 10. Pie-menu custom positioning for single-handed usage (left) and cradling context (right)

As already mentioned, all menu navigation tasks were executed on users' personal devices, with screen resolutions ranging from 540 × 960 up to 1440 × 2960 pixels. We acquired all customized pivotal positions from the obtained log files, normalized them, and calculated the related centroids. As shown in Fig. 10, participants generally tend to place the pie-menu widget roughly in the middle (horizontally) and more to the bottom part (vertically) – both when interacting single-handedly and via cradling. We expected centroid for the single-handed usage to be more to the right side of the screen, seeing that only 11% of the participants were left-handed. However, it seems that the central horizontal position of the pie widget still allows all menu items to be easily reachable, even when larger smartphone models are being used.

5 Conclusion

In this paper we introduced a specific pie-menu implementation for mobile touchscreen devices, which allows swipe-based navigation through deep hierarchical menu configurations. The proposed solution encompasses several important design features such as circular layout, floating widget form, semi-transparent visualization, marking menu concept, occlusion awareness, and adaptability options.

Unlike typical mobile application menus, which usually occupy the entire screen or significant part of it, our pie-menu widget floats on top of the underlying content and acts as an accompanying service. This, along with the incorporated semi-transparency feature, provides more effective utilization of available screen space and better visibility of the application content.

The most important characteristic of the proposed solution is its interaction modality which relies on swipe gestures and touch dwelling, rather than utilizing repetitive tap-based selections. Namely, menu navigation is performed by making use of a single swipe gesture, with touch-and-hold events being dedicated to submenu invocations. In such a design, the dwell time represents a key factor for the efficiency of menu navigation. This was confirmed by conducting the empirical evaluation of the proposed pie-menu involving 36 participants. The results obtained from the experiment revealed that dwell time indeed has a significant effect on both the menu navigation time and the number of errors being made. Specifically, users were significantly faster when individually preferred dwell time was applied (mean value: 0.38 s), as opposed to the competing values initially provided by default (0.25 s, 0.5 s, and 1 s). The related gain in menu navigation speed comes without noteworthy degradation in accuracy (mean error rate: 7%). As expected, the longest default dwell time value (1000 ms) showed to be the least efficient option for swipe-based pie-menu navigation, although it yielded a lower amount of selection errors.

In addition, empirical research also tackled two different contexts of use: single-handed and cradling-based interaction. Data analysis revealed that the corresponding factor (interaction style) does not have a significant effect on the menu navigation efficiency, as task execution speed and the number of errors are not affected by a way of holding the smartphone device. This outcome is furthermore corroborated by the post-study TLX questionnaire, wherein participants comparatively assessed the perceived workload with no major preference towards a certain interaction style.

Overall usability of the proposed pie-menu is perceived as quite good, with learnability and ease of use being the best-rated attributes. Hence, the incorporated interaction design, which utilizes swipe gestures and touch dwelling, can be considered as a practical way to traverse through deep hierarchical menu configurations. Among all the features involved in the pie-menu design, the provided adaptability is the one the most praised among the participants. Namely, the possibilities to change dwell time and menu pivotal position showed to be the most influential aspects from the usefulness point of view. This comes as no surprise, knowing that highly adaptable user interfaces generally attract most of the nowadays users. In that sense, we argue that our pie-menu design provides interaction benefits in the mobile touchscreen domain.

Within the conducted research, all experiment procedures were completely administered online. COVID-19 pandemic, along with the restrictions imposed at the time of our research activities, made the fully controlled in-situ testing unfeasible. However, we demonstrated how typical HCI experiment, if planned thoroughly, can be successfully carried out remotely as well.

As for future work directions, occlusion problem still remains to be tackled in a way to deliver the even better user experience. Also, in order to get a detailed insight into the effect of the marking menu concept, a longitudinal study should be carried out wherein memorizing the swipe trajectories (for frequently selected menu items) could provide additional benefits for end users of the proposed pie-menu solution.

6 References

[1] T. Vainio, “Review of the Navigation HCI Research During the 2000’s,” International Journal of Interactive Mobile Technologies (iJIM), vol. 4(3), pp. 36–42, 2010.

[2] G. Bailly, E. Lecolinet, and L. Nigay, “Visual menu techniques,” ACM Comput. Surv., vol. 49(4), pp. 60:1–60:41, 2017. https://doi.org/10.1145/3002171

[3] R. V. Nacheva, “Standardization Issues of Mobile Usability,” International Journal of Interactive Mobile Technologies (iJIM), vol. 14(7), pp. 149–157, 2020. https://doi.org/10.3991/ijim.v14i07.12129

[4] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman, “An Empirical Comparison of Pie vs. Linear Menus,” In: Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI’88), pp. 95–100, 1988. https://doi.org/10.1145/57167.57182

[5] J. M. Rubio, P. Janecek, “Floating Pie Menus: Enhancing the functionality of Contextual Tools,” In: Proc. ACM Symp. User Interface Software and Technology (UIST’02), pp. 39–40, 2002.

[6] P. Brandl, J. Leitner, T. Seifried, M. Haller, B. Doray, and P. To, “Occlusion-Aware Menu Design for Digital Tabletops,” In: Proc. Extended Abstracts on Human Factors in Computing Systems (CHI EA’09), pp. 3223–3228, 2009. https://doi.org/10.1145/1520340.1520461

[7] T. Hesselmann, S. Flöring, and M. Schmitt, “Stacked Half-Pie Menus: Navigating Nested Menus on Interactive Tabletops,” In: Proc. ACM Int’l Conf. Interactive Tabletops and Surfaces (ITS’09), pp. 173–180, 2009. https://doi.org/10.1145/1731903.1731936

[8] S. Huot and E. Lecolinet, “ArchMenu et ThumbMenu: contrôler son dispositif mobile «sur le pouce»,” In: Proc. Conf. l'Interaction Homme-Machine (IHM’07), pp. 107–110, 2007. https://doi.org/10.1145/1541436.1541457

[9] J. Francone, G. Bailly, L. Nigay, and E. Lecolinet, “Wavelet menus: a stacking metaphor for adapting marking menus to mobile devices,” In: Proc. Int’l Conf. Human-Computer Interaction with Mobile Devices and Services (MobileHCI’09), pp. 49:1–49:4, 2009. https://doi.org/10.1145/1613858.1613919

[10] D. Bonnet and C. Appert, “SAM: the Swiss Army Menu,” In: Proc. Conf. l’Interaction Homme-Machine (IHM’11), pp. 5:1–5:4, 2011. https://doi.org/10.1145/2044354.2044361

[11] G. Kurtenbach, and W. Buxton, “The limits of expert performance using hierarchic marking menus,” In: Proc. Conf. Human Factors in Computing Systems (CHI’93), pp. 482–487, 1993. https://doi.org/10.1145/169059.169426

[12] F. Thalmann, U. von Zadow, M. Heckel, and R. Dachselt, “X-O Arch Menu: Combining Precise Positioning with Efficient Menu Selection on Touch Devices,” In: Proc. ACM Int’l Conf. Interactive Tabletops and Surfaces (ITS’14), pp. 317-322, 2014. https://doi.org/10.1145/2669485.2669539

[13] J. Zheng, X. Bi, K. Li, Y. Li, and S. Zhai, “M3 Gesture Menu: Design and Experimental Analyses of Marking Menus for Touchscreen Mobile Interaction,” In: Proc. Conf. Human Factors in Computing Systems (CHI’18), pp. 249:1–249:14, 2018. https://doi.org/10.1145/3173574.3173823

[14] R. Ecker, V. Broy, A. Butz, and A. De Luca, “pieTouch: A Direct Touch Gesture Interface for Interacting with In-Vehicle Information Systems,” In: Proc. Int’l Conf. Human-Computer Interaction with Mobile Devices and Services (MobileHCI’09), pp. 1–10, 2009. https://doi.org/10.1145/1613858.1613887

[15] Y. Kammerer, K. Scheiter, and W. Beinhauer, “Looking My Way through the Menu: The Impact of Menu Design and Multimodal Input on Gaze-based Menu Selection,” In: Proc.

Symp. Eye Tracking Research & Applications (ETRA’08), pp. 213–220, 2008. https://doi.org/10.1145/1344471.1344522

7 Authors

Alen Salkanovic is a PhD student at the University of Rijeka, Faculty of Engineering, Croatia. He currently works as a teaching assistant at the Department of Computer Engineering and is a member of AIRI: Center for Artificial Intelligence and Cybersecurity at the University of Rijeka. His research activities primarily cover mobile-HCI topics.

Matija Stojkovic currently works as a teaching assistant at the Department of Computer Engineering (University of Rijeka, Faculty of Engineering, Croatia). His main research interests are in the field of mobile robotics, distributed algorithms, and their application to ad-hoc and wireless sensor networks.

Sandi Ljubic currently holds the position of Assistant Professor at the University of Rijeka, Faculty of Engineering Department of Computer Engineering, Croatia. He leads the Human-Computer Interaction Lab at AIRI: Center for Artificial Intelligence and Cybersecurity. His research interests span a range of topics in the HCI field, including mobile interaction, predictive modeling and evaluation, universal access, and text entry methods. Email: sandi.ljubic@riteh.hr

Article submitted 2020-08-14. Resubmitted 2020-10-11. Final acceptance 2020-10-13. Final version published as submitted by the authors.