# Wave Menus: Improving the Novice Mode of Hierarchical Marking Menus

Gilles Bailly[1,2], Eric Lecolinet[2], and Laurence Nigay[1]

[1] LIG University of Grenoble 1, Grenoble, France
[2] GET/ENST – CNRS UMR 5141, Paris, France
{gilles.bailly, laurence.nigay}@imag.fr, {gilles.bailly, eric.lecolinet}@enst.fr

**Abstract.** We present Wave menus, a variant of multi-stroke marking menus designed for improving the novice mode of marking while preserving their efficiency in the expert mode of marking. Focusing on the novice mode, a criteria-based analysis of existing marking menus motivates the design of Wave menus. Moreover a user experiment is presented that compares four hierarchical marking menus in novice mode. Results show that Wave and compound-stroke menus are significantly faster and more accurate than multi-stroke menus in novice mode, while it has been shown that in expert mode the multi-stroke menus and therefore the Wave menus outperform the compound-stroke menus. Wave menus also require significantly less screen space than compound-stroke menus. As a conclusion, Wave menus offer the best performance for both novice and expert modes in comparison with existing multi-level marking menus, while requiring less screen space than compound-stroke menus.

**Keywords:** Marking menus, Wave menus, novice mode.

## 1 Introduction

In recent studies, much attention has been devoted to marking menus [7, 8] and their extensions [15, 16]. Marking menus are a combination of pop-up radial menus and gesture recognition. They provide two interaction modes: novice and expert modes. On the one hand, if a user presses and waits a fraction of a second, radial menus pop up and items can be selected in a manner similar to common linear menus: this is the novice mode. On the other hand, the user can select an item without waiting for the menus to pop-up, by drawing a mark that leaves an ink trail corresponding to a selection path through the hierarchical menus: this is the expert mode. The key property of marking menus is that they support a seamless transition from novice to expert mode [6]. In that sense, marking menus can be seen as pseudo-gestural interaction techniques since they make it possible to learn gesture interaction from a menu system. Several user studies have revealed their efficiency over linear and pie menus [1]. Alternate designs have also been proposed to improve their efficiency when menus contain a large number of items [15, 16].

C. Baranauskas et al. (Eds.): INTERACT 2007, LNCS 4662, Part I, pp. 475–488, 2007.

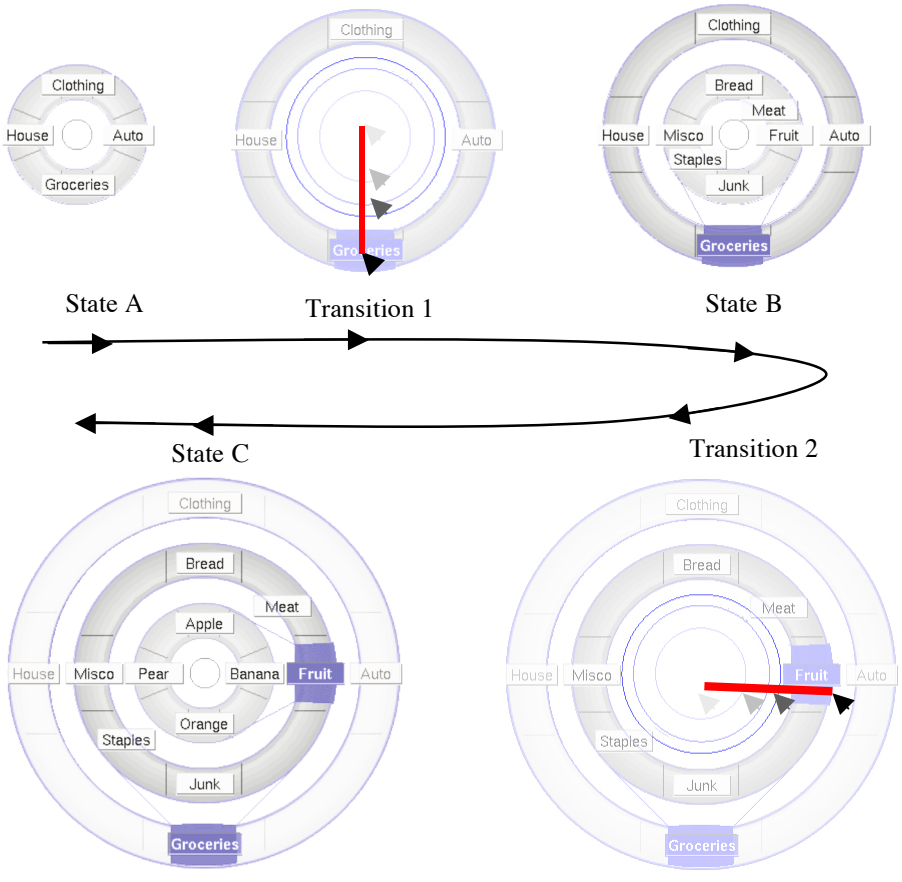


**Fig. 1. WaveMenus.** (State A) The menu appears centered around the cursor. By drawing a stroke towards a desired item (Transition 1), the first level menu is enlarged to leave room for the submenu (State B). With a second stroke (Transition 2), the third level appears at the center and the first and second levels are enlarged (State C).

However, most of the recent studies were driven by the efficiency of the expert mode. The key point of this article is that the efficiency of the novice mode of marking menus is crucial for user acceptance of the menuing technique. Indeed:

- *The novice mode is the first interaction mode* with the menu and *an unavoidable step* before expert mode. The users must be able to learn how to use the most used commands in expert mode. The underlying principle of marking menus is that the users will learn the expert mode implicitly by activating these commands repeatedly in novice mode. This feature obviously highlights the crucial role of the novice mode: if the user interaction in novice mode is not efficient or too cumbersome (in terms of speed and accuracy), the users may reject marking menus immediately.

- The *novice mode never disappears*. Many menu items are seldom used by users. For instance, a study showed that 90% of Microsoft Office commands are rarely used [5]. This implies that many commands will be activated in novice mode, even by expert users that are familiar with the application. Besides, users with limited physical and/or cognitive skills (including handicapped and aged users) may never use the expert mode: this is an important consideration when designing applications to ensure widespread user adoption.
- *The novice and expert modes co-exist*. As pointed out in [7], transition from novice to expert mode is not one way. Switching back and forth between novice and expert modes is an important feature of marking menus. For example after a long lay-off period, the expert users return to the novice mode to refresh their memory of the layout of the menu.

In this paper, we focus on the improvement of the novice mode of marking menus by presenting a new hierarchical marking menu technique, called **Wave Menu** (Fig. 1), that combines the advantages of two previous marking menu techniques (compound-stroke [8] and multi-stroke [15]):

- In *novice mode*, Wave menus provide better performance than multi-stroke menus and similar performance to compound-stroke menus.
- In *expert mode*, Wave menus preserve the efficiency of multi-stroke menus that have been shown to outperform the compound-stroke menus.

The first section presents previous work and criteria for characterizing marking menus. These criteria are then used as the basis for the design of our two variants of the Wave Menu. We finally present the results of a controlled experiment that compares the novice mode of our two variants of the Wave menu with that of existing marking menus while highlighting the difficult and nearly unexplored problem of experimentally evaluating the novice mode of a menu.

## 2 Criteria for Characterizing Marking Menus

For the sake of clarity, we first briefly describe the existing hierarchical marking menus before presenting criteria for characterizing marking menus and especially their novice mode.

### 2.1 Marking Menus Compared

**Compound-Stroke Menus (CS menus)** [8], are based on *spatial composition*: users make a compound mark that results from the combination of the elementary strokes needed to activate items in the menu hierarchy in novice mode. The gesture is performed continuously without releasing the mouse button. An important feature is that these menus are scale invariant in expert mode, so that the user does not have to look at the screen to check the location of the mouse release. As for traditional menu systems, the novice mode provides *previsualization*: users can browse submenus by dragging the mouse on their parent items or by tapping on them.

**Multi-Stroke Menus (MS menus)** [15], a variant of CS menus is based on *temporal composition:* the elementary strokes are drawn in quick succession, and each stroke

(which is inflexion free) is completed when the user releases the mouse. A delay is needed to determine if a stroke belongs to the previous sequence, or if it initiates a new interaction sequence. In contrast to CS menus, the marks can overlap.

**Zone and Polygon Menus** [16], two variants of MS menus, consider both the relative position and orientation of elementary strokes. These menus improve MS menus by extending the breadth of menu to 16 items (CS and MS being limited to breadth-8) while preserving efficiency.

### 2.2 Criteria

The presented criteria are inspired by previous work on marking menus [6, 15, 16]. However, while most previous studies mostly focused on expert mode, we here put a special emphasis on the novice mode. The first two criteria presented correspond to system design constraints: the screen space that is required by the menuing technique and its supported number of items. The following criteria then cover the three aspects of usability [14]: Efficiency, Learnability and Satisfaction.

**Screen space.** The screen space required for displaying hierarchical marking menus in novice mode is a major limitation. This problem is critical for handheld devices as well as for contextual "pop-up" menus. Indeed, circular menus are at least twice as large as linear menus because items are laid out on both sides around the center point. Items must also be centered around the cursor that made them appear. These contrast with linear menus which can either appear on the left or the right side of the mouse press, depending on the space to the screen borders remaining. A 3-level CS menu hence requires more horizontal space than 10 linear menus to display its leftmost and rightmost branches (Fig. 2). This makes CS menus bad candidates for small screens or for being used as contextual menus, even on large screens: if the selected object is too close to the screen border, the menu will be partially hidden or it will not be centered around the mouse cursor, thus making user interaction cumbersome.

Other designs, such as MS menus and their variants (i.e. Zone and Polygon menus) still require more space than linear menus but are much less space consuming than CS menus because their submenus are superimposed. Moreover in expert mode, MS menus also require less space than CS menus because the performed simple marks can overlap [15].

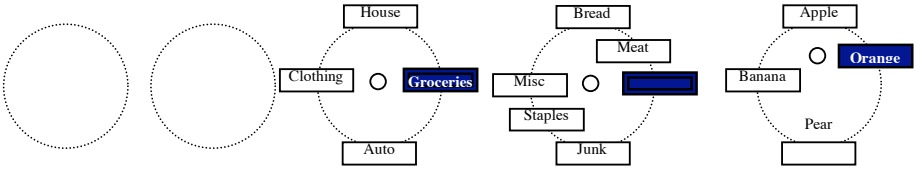


**Fig. 2.** Space requirements in breadth for three-depth Compound-Stroke menus

**Number of items.** Many applications now contain a very large number of commands (for instance, 642 commands were available in Microsoft Word 6 [5]). Traditional menu systems may have some submenus that contain a rather large number of items

(typically more than 10 or 15 items) as opposed to CS and MS menus for which previous studies [6, 15] have shown a breadth equal to 8 to guarantee good performance in expert mode. Note that for multilevel CS menu, the breadth is even smaller than 8 because certain compound marks are ambiguous due to the scale invariance of the marks (for example a 3-level CS menu can allow 400 items instead of 512=8x8x8 items). The breadth limitation of CS and MS menus may imply awkward groupings of items [16] as well as an increased menu depth; two consequences that may lead to (a) a decrease in performance in expert mode (especially for CS menus, as shown in [15]) and (b) an increased decision time and possible disorientation in novice mode as the user needs to navigate in a larger number of submenus [10]. Zone and Polygon menus [15] have been introduced as a way to solve this problem by extending the menu breadth up to 16 items. However, it is important to notice that the time to select an item amongst N not only depends on the depth and breadth of the menu but also relies on various factors such as the type of menu task and the order of items. Empirical testing is thus often needed to define an optimal breadth and depth combination.

**Speed and Error rate.** *Speed* measures the time needed by users to activate appropriate items. *Errors* occur when the user selects an erroneous command instead of the one s/he was supposed to activate. The speed and error rate criteria are obviously related as users are more likely to make errors when they try to interact faster. The speed and the error rate also depend on the layout of the menu, especially the depth and the breadth of the menu, as explained in the previous section [4, 10] and the orientation of items in the circular submenus [8, 15]. We distinguish two cases depending on if the menu is used in expert or novice mode.

*Expert mode*. MS menus are more efficient than CS menus in expert mode [15]. MS menus allow for slightly faster selections and significantly reduce the error rate for large hierarchies. Their accuracy remains approximately constant (about 93%) for 2 and 3-level menus and does not decrease for diagonal directions. A possible explanation is that the spatial complexity of strokes does not vary with MS menus: users must always draw simple marks (but multiple times) while CS menus require more complex continuous compound gestures. Zone and Polygon menus are slower than MS menus for a given depth because the user must "tap" before drawing the stroke. However, as they provide a larger breadth, their depth tends to be smaller, so that the global performance of a menu system is likely to be equivalent or even better.

*Novice Mode.* Studies have shown that Pie menus were 15-20% faster than linear menus in novice mode [1]. Still, to the best of our knowledge, the marking menu techniques have not yet been compared in novice mode even though novice mode efficiency is a key issue for user acceptance (section 1).

The notion of "speed" is harder to define in novice mode than in expert mode. It can be viewed as the mean time needed to activate items that the user does not already know, and thus, that s/he cannot select in expert mode. Nevertheless the notion of a novice usage corresponds to different kinds of user behaviour and strategy [10]. A user can for instance search for a "concept" he has never or rarely used before. For example, the user of a new text editor will probably search the various commands for changing the text properties (such as the font size, the font style, etc.). These

commands may be located in different submenus at different levels in the hierarchy. A complete novice user may explore the whole menu system to find all commands that are related to text properties. A more experienced user may approximately know where these commands are located. An even more experienced user may know the appropriate submenu but not the precise location of the command in this submenu. This last case corresponds to an *intermediate mode* [3] where the user could start her/his search in expert mode (by performing a gesture that will open the right submenu) and complete her/his action in novice mode (by selecting the appropriate item in this submenu). This intermediate mode highlights the need for supporting the switch between novice and expert mode in menu techniques (section 1). Besides, even a novice user is unlikely to perform her/his search in a completely "blind" way: s/he will probably use the semantics of the menu labels to constrain her/his search. This property makes a search task in a menu system quite different from an exploration task in an arbitrary tree where item labels may convey little or no semantic information. A search in a menu system is always *directed* by a priori user knowledge (which can be domain knowledge, knowledge from previous uses of the application or of similar applications).

As a conclusion, the time to select an item amongst N in a menu system depends on various factors, such as the menu layout as explained in the previous section. Hence, novice mode efficiency cannot be evaluated in the same straightforward way as for expert mode. As an attempt to solve this problem, we have defined a "typical novice task" to compare the relative performance of different marking menu designs. This task, which can be parameterized in order to control the complexity of the search, will be detailed in section 4.

**Learning time and retention over time.** *Time to learn* evaluates how fast users are able to acquire the physicals skills needed to operate the menu. This involves three different aspects: a) the time to understand how the menu works; b) the time to understand how items are organized; and c) the time needed for the transition from the novice to the expert mode.

*Retention over time* estimates "how well users maintain their knowledge after an hour, a day, or a week. Retention may be related to time to learn. Frequency of use also plays an important role." [14]. These criteria have only been studied for 1-level CS menus [7]: it has been found that novice users would eventually become experts using the mark-ahead feature 90% of the time. This study also highlighted that the time needed to reacquire expert mode after an idle period decreases with the amount of experience. Zone and Polygon menus may lead to significantly different results because of their more sophisticated design. This is especially true for Polygon menus where users must draw marks that do not start from a central point and that activate different items according to the direction of the mark. The time to understand how the menu works may thus be longer than for MS and CS menus. Nevertheless an informal study of these menus [6] suggests that their increased breadth (16 items instead of 8 items) may favor the learnability of the overall organization of the menu items.

**Subjective satisfaction.** By estimating how much users liked using various aspects of the menu system, "subjective satisfaction may be the key determinant of success" [14]. It can depend on many factors, from design to subjective efficiency. The novice mode plays an important part in the evaluation of satisfaction because it is the first

mode used. If the novice mode does not satisfy the user, there is little chance that s/he will continue to use it even if the expert mode is very good.

## 3 Wave Menus and Inverted Wave Menus

Our rationale for designing the Wave menus is guided by the above criteria. Considering the speed and error rate criterion, for expert mode, MS menus and other variants based on temporal composition, outperform CS menus. We chose to improve MS menus, rather than Zone or Polygon menus because their more complex design may increase the time to learn and no experimentation has yet been done to study this important aspect. We then focus on the novice mode of MS menus and propose alternate designs for improving the novice mode while preserving the efficiency of the expert mode.

For novice mode, MS menus require less screen space than CS menus since a submenu is displayed on top of its parent menu. However this approach may imply problems for exploration of the hierarchical menu and cause user disorientation as the navigation path is not visible. This lack of contextual information although minimizing the required screen space criterion may therefore have a negative impact regarding the time to learn criterion. As people acquire display-based knowledge, the graphical display is crucial to learn the structure of the menu. The context in which each action is performed helps the user to recall the methods used (i.e. the sequence of actions). Moreover MS design does not allow previsualization. Previsualization is a common feature of linear and CS menus that allows the user to browse the submenus of a given menu by dragging the mouse over its items. Previsualization is a proactive feedback [13] that is important for exploring menu systems because it provides a means for a fast inspection of possible commands [12].

To sum up, the rationale for designing Wave menus corresponds to an alternate design of MS menus that improve their novice mode while preserving the efficiency of their expert mode. Its graphical design is geared towards requiring less space than CS menus while offering previsualization of submenu items and path visualization. As a consequence, Wave menus work exactly the same way as MS menus in expert mode. Fig. 1 shows an example of a 3-level 8-breadth menu that illustrates the new graphical design we designed for novice mode. In Fig. 1, *State A* shows the root menu, which is centered around the cursor. As for MS menus, drawing a stroke towards an item of the root menu opens the corresponding submenu. However, this submenu is not displayed on top of its parent but at the center of the menu system. In order to remain visible, its parent menu is then enlarged to surround this submenu (*State B*). The same effect occurs again if another stroke is made (*State C*). Submenus move outwards from the center, recalling the propagation of waves. The outmost menu is always the root menu while the inmost menu is always the deepest submenu in the hierarchy that is currently visible on screen (as deeper submenus may exist but may still be closed).

For backing up a level in the hierarchy, the user clicks in the center area of the inmost submenu. The inmost submenu then disappears and the menus that surround it are shrunk. For instance, clicking twice on the center of the menu system shown in Fig. 1 in *state C* would make it appear in *state B* then in *state A*. An important

consequence is that the user mostly interacts with the inmost menu. The cursor remains in the center area of the layout thus reducing the time for pointing (length movement) and the focus of attention remains at the center. The fact that interaction location and focus of attention remain stable at the center is likely to improve performance.

Users can open submenus without releasing the mouse button. Menus start being expanded when the cursor crosses an item border of the inmost menu. A fast animation is then performed to make the corresponding submenu pop-up in the center of the menu system. This effect, that is related to crossing-based interaction [2, 11] simulates the idea of "pushing the menu outwards". It only requires small movements to avoid affecting performance. Symmetrically a fast animation occurs when the user clicks on the center for closing a submenu to produce a "collapsing effect". The role of these animations is to help users to understand layout changes when the menu is expanded or collapsed. We have informally observed during experimentation that "crossing" (i.e. interacting when reaching item borders) seemed to be quite "natural" for users. A possible explanation is that drawing a stroke that crosses an item's border affords the idea of stretching this object in order to make its content (i.e. its submenu) appear. Moreover, as submenus appear in the center, it makes sense to drag their parent items outwards to "make place" in the center.

Another important characteristic of Wave menus is that users can interact with *all* visible items by clicking or dragging the mouse on them. This makes it possible to explore different branches without closing menus and to previsualize submenus. Previsualization can be performed continuously by dragging the mouse over the items of any parent menu. The current path in the menu tree is always visible as the ascendants of a submenu appearing in the center are necessarily displayed (by construction of the Wave menu). Wave menus thus offer submenu previsualization and path visualization, two features of traditional menus that increase learnability as explained in section 2.

Finally, with regards to required screen space criterion, Wave menus require the same *minimum* amount of screen space as MS menus because they can be operated in the same way, by interacting only with the inmost submenu. Some operations, such as direct interaction with outer menu items require more space. But Wave menus can still be used, even if it is in degraded mode, when they are close to the screen borders. This property contrasts with CS menus. Besides, their layout also provide better screen space efficiency than CS menus because submenus do not appear on the side of their parents but are equally distributed around the same central point **(Fig. 1** & **2)**.

**Alternate design.** A major feature of Wave menus, which may be surprising at first, is that the children of the menu tree appear in the center of the graphical representation while the parents (and the root nodes) are located on the outmost rings. Such a way of displaying (and interacting with) trees is somewhat unconventional. This layout scheme also differs from the marking and pie menus [1]. Hence, we also developed a variant called *Inverted Wave menus* (WM2) where tree nodes are displayed in the opposite order. With this variant, the root menu appears in the center of the graphical representation, its submenus appear around the root and so on.

The main drawback of this representation is that it requires more space to be usable. In contrast with the standard Wave menu, it cannot be used in "degraded mode" when the activation point is close to the screen borders. Besides, it also leads

to more changes of focus as the user cannot focus on the center area most of the time, as is the case with our previous design. But we found it interesting to experiment with these dual designs and to compare them with some of the most promising marking menu techniques proposed so far.

## 4 Experimental Comparison of Novice Mode Efficiency

We conducted a controlled experiment to compare the efficiency of the *Multi-Stroke* (MS), *Compound-Stroke* (CS), *Wave* (WM) and *Inverted Wav*e (WM2) menus in novice mode. As pointed out in section 2, novice mode usage can be seen as a kind of a search that is guided by the semantics of the labels and a priori user knowledge. The key point here is that users will generally not explore the entire menu tree but only a subpart of it, whose size depends on user experience. Moreover, they may want to find all commands that are related to a given topic in order to decide which of them is more appropriate.

As the knowledge across subjects is different, a search actually guided by semantics is likely to introduce bias in the exploration task. In order to avoid this effect, we gave the same name (e.g. *Name1*) to all the "possible" items in the root menu. This means that the subject must only open the submenus that correspond to *Name1* items. The number of *Name1* items (and their corresponding submenus) controls the difficulty of the task. This design modelizes a situation where the user needs to open several menus (the larger the number, the more of a novice he is).

Similarly, we gave the same name (e.g. *Name2*) to the items that must be found by the user in the submenus. Some submenus contains one occurrence of *Name2*, some other submenus do not contain it. Again, the number of *Name2* items is a fixed parameter of the task. *Name2* items are all terminal items: they do not open other submenus. While our design could easily be extended to 3-level menus we only considered 2-level menus to avoid too long experiments, that is to say less than 1 hour per subject for testing the four techniques.

The proposed task is thus as follows: the subject must activate **all** *Name1* items in the root menu in order to open the corresponding submenus and s/he must search for **all** *Name2* items in these submenus. A key factor is that the user must consider **all** *Name1* and *Name2* items. This makes it possible to control the difficulty of the task and thus to avoid excessive variability. A naive design would consist in asking the user to find and select the **first** *Name2* occurrence (instead of **all** *Name2* occurrences) in the set of possible submenus. This would make it difficult to control the task complexity as the number of submenus that users would have to open and scan for finding *Name2* would depend on chance. As the number of submenus explored is unpredictable it cannot be easily counterbalanced with trials. Hence, we preferred a design where the complexity is known in advance. In our experiments, the number of *Name1* items is fixed and set to 4, a value that corresponds to a user that is partially, but not completely, novice. The number of *Name2* items varies from 2 to 4 in a way that is counterbalanced among the techniques, so that complexity is the same for all the techniques. This number varies to force users to open all the 4 possible submenus (if it was fixed and inferior to 4, users could perform the task without opening all the submenus in some trials and this would cause inopportune variability; if it was always

equal to 4, all submenus would contain the *Name2* item, a situation that is less realistic than the proposed model).We performed this experiment to check the following hypotheses:

- **H1:** Multi-Stroke (MS) menus are slower than Compound-Stroke (CS) menus in novice mode as they do not provide *previsualization* and *visible path availability*.
- **H2:** Wave menus and CS menus have similar performance in novice mode.
- **H3:** Wave menus have similar performance to their "Inverted" variant: the order of the hierarchy does not matter.
- **H4:** There is no significant effect on the number of errors and the number of missed items between menu techniques.

### 4.1 Experimental Design

**Participants.** Twelve volunteers participated in the experiment (5 women, 7 men) from 24 to 35 years (mean 26). They were all accustomed to using computers and to interaction techniques such as linear multilevel menus. None of them had prior experience with the tested techniques, nor knew which was the one we had designed.

**Apparatus.** The experiment was conducted on a laptop with a 2 GHz Pentium 4 processor, with a 17" display. All menus were implemented by using the Ubit [11] GUI Toolkit (version 5.0) with OpenGL acceleration. In contrasts with many prior studies, we did not use a pen interface but a Logitech (TM) mouse as an input device. The primary reason is that most of our users had no experience with using pen tablets. Much longer training time would have been required to avoid possible bias due to pen interaction learning. Besides, our study focuses on user acceptance of marking menus for common use, and the mouse is much more commonly used as an input device.

**Task and Stimulus.** As explained above, the task consists in finding and selecting the multiple occurrences of Name2 items in all Name1 submenus (i.e. submenus that correspond to Name1 items in the root menu). Name1 appears 4 times among the 8 items of the root menu so that the user must open 4 submenus that may contain Name2. The number of Name2 occurrences, which varies from 2 to 4, is counterbalanced over techniques and submenu breadths. As would be the case in a real task, Name1 and Name2 are existing words. They are respectively a continent and a city of this continent. To avoid bias due to different subject knowledge both names are displayed during the trial. The names contained in the menus are changed at the end of each trial. We only performed experiments with 2-level menus in order to keep their duration within reasonable limits (about 1 hour). Each submenu is filled with 4, 6 or 8 items where the position is counterbalanced in order not to favor a specific direction.

**Procedure.** The experiment was divided into 4 blocks, one per menu technique, counterbalanced across subjects using a Latin square design. For each block, we let users discover the menu technique for three minutes without giving indications. Participants were then instructed about how to use the menu for five minutes. Then, they had to execute 18 randomly-chosen trials for each block. Each trial works as follows: 1) the participant presses the mouse in the center of the screen to start the experiment; 2) the continent (Name1) and the city (Name2) appear on the screen; 3)

the participant opens the menu by pressing the mouse; 4) he can then search and select the multiple occurrences of the city name in the menu hierarchy; 5) he presses the space bar when he thinks he has found all occurrences; 5) a new screen appears with a button to start the next trial. Participants were allowed to take breaks between trials. Breaks were enforced between different techniques. In summary, the experiment (excluding training) follows this design:

12 subjects x 4 techniques (MS, CS, WM, WM2) x
3 submenu breadths (4-6 or 8 items) of 6 selections = 864 menu selections

The dependent variables are: the reaction, execution and total times, the number of errors and the number of missed items. Reaction time is measured as the interval between the appearance of the stimulus and the mouse press to open the menu. This interval represents the time needed by participants to understand stimuli before making a selection. Execution time is measured as the interval between this mouse press and when the user presses the space bar to finish the trial. The total time is obtained by adding up the reaction time and the execution time. The number of errors is the number of wrong selections. The number of missed items is the number of city name occurrences that were present in the menu but not selected by the user.

### 4.2 Results

The two-way analysis of variance indicates no significant effect for menu techniques on **reaction time**, thus suggesting there is no difference in mental preparation time. The **execution time** is calculated without taking into account trials with missed items. Analysis of variance reveals a significant main effect for technique on execution time ($F_{3,33}$= 47.25, p< .0001). A post hoc Tukey test with 1% alpha level shows that the Wave and CS menus are significantly faster than the MS and Inverted Wave menus. As expected, our results also reveal a significant main effect for submenu breadth ($F_{2,22}$= 68.69, p< .0001). There is also a significant submenu-breadth x technique interaction ($F_{6,66}$ = 2.75, p< .001), indicating that breadth changes affected menu

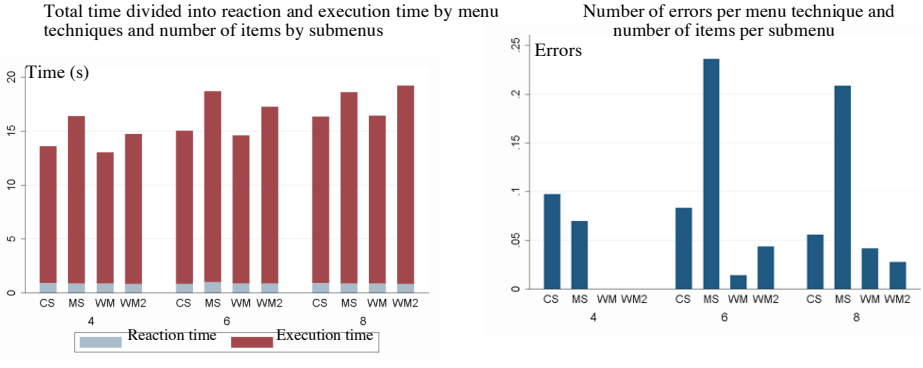


**Fig. 3.** Performance of the Compound–Stroke menu (CS), Multi-Stroke menu (MS), Wave menu (WM) and Inverted Wave Menu (WM2) with 4, 6 or 8 items by submenu

techniques differently. Pairwise means comparisons (Tukey test with 1%) indicate that there is no significant time difference between 6 and 8 items for the MS menu. It also indicates that the WM2 is significantly slower with 6 and 8 items. A possible explanation is that the distance between items is much larger in this design that takes longer to read all items. Finally, the analysis of **total time** (Fig. 3) shows the same significant effect as for execution time (as there is no effect for reaction time).

There is a significant effect for techniques on the **numbers of errors** ($F_{3,33}$= 13.59, p< .0001) as shown on Fig. 3. A post hoc Tukey test with 1% alpha level shows that the number of bad selections is significantly higher for MS menus than for other techniques. Wave menus produce 8 times fewer errors than the MS design. This can be explained by the fact that users have difficulty selecting the center of the menu when they want to come back one level up. The accuracy for different *submenu-breadths* is statistically significant (but the Tukey test with 5% indicates that the differences are not significant). There is also a significant submenu-breadth x technique interaction ($F_{6,66}$ = 2.75, p< 0.1). Pairwise mean comparisons (Tukey test with 1%) indicate that MS menus generate significantly more errors for 6 and 8 items. *Techniques* and *submenu-breadth* have no significant effect on the number of **missed items**.

After the experiment, we asked participants to rank-order each of the four menu techniques.

- 6 participants chose CS menus as their favorite technique and 5 of them chose Wave menus. 9 participants ranked MS menus as their least favorite technique.
- 5 participants judged CS menus as the easiest to learn, 3 chose Inverted Wave menus and 2 standard Wave menus. Conversely, 6 participants considered MS menus as the hardest technique to learn and 4 Wave menus.
- For 7 participants MS menus was the most tiring technique, for 2 participants it was the CS and Inverted Wave techniques. In contrast, 7 participants found CS as the least tiring technique, 2 chose WM and MS.
- 8 participants considered that MS menus made it hard to perceive the hierarchy and 3 found this to be the case for the Inverted Wave menus. Conversely, 7 participants said that CS menus were the best technique to perceive the hierarchy, and 2 chose WM and WM2.
- 7 subjects were not bothered by the inverse hierarchy used in Wave menus, 3 said they were uncomfortable only at the beginning and 2 during the whole session.

### 4.3 Discussion

We can now revisit and attempt to confirm the hypothesis posed earlier:

***H1:*** *Multi-Stroke (MS) menus are slower than Compound-Stroke (CS) menus in novice mode*. Our results clearly show that CS menus are faster than MS menus in novice mode (18% faster). As seen in previous sections, *previsualization* and *visible path availability*, two features that MS menus do not provide, seem to have a major effect in improving the navigation in the menu system. In fact, 11 subjects said they felt they could easily get lost in MS menus because they were unable to remember which submenus they had already visited.

***H2:*** *Wave menus and CS menus have similar performance in novice mode*. This hypothesis is confirmed by our results and can be explained in the same way as for H1: as they provide the same previsualization and visible path availability features as CS menus, they also share the same performance in novice mode.

***H3:*** *Wave menus have similar performance to their "Inverted" variant: the order of the hierarchy does not matter*. Our results show that Wave menus are actually faster that their inverted variant. This may seem surprising but can be explained by a larger distance between the "most important" items. An attractive feature of the normal Wave menu is that the deepest submenu of the visible part of the hierarchy is always located in the center of the menu. This makes it possible for users to focus on this zone and to avoid multiple changes of attention, as is the case with the Inverted Wave menu. This fact was also pointed out by several users who complained about the excessive distance between the items they had to read.

Hence, it is somewhat difficult to conclude on the effect of the order of the hierarchy on performance. However, the fact that the Wave and the CS menu share similar performance tends to prove that this order does not really matter. Besides, most users (10/12) said they were not bothered by the reversed hierarchy after sufficient learning time.

***H4:*** *There are no significant effects on the number of errors and the number of missed items by menu techniques.* We found no significant difference between techniques for the number of missed items. But the number of errors is surprisingly high for Multi-stroke menus, which produces 7 times more errors than other techniques. While observing participants, we noticed that many of them had difficulty clicking in the center zone of the menu to come back to the previous level, and often selected a wrong item. This problem does not occur in expert mode where users never come back to the previous level.

## 5 Conclusion

Motivated by a criteria-based analysis of marking menus, we have presented Wave menus, a variant of multi-stroke marking menus designed for improving their novice mode. A user experiment focusing on the novice mode showed that compound-stroke marking menus and Wave menus are significantly faster (18%) and more accurate (7 times fewer wrong selections) than multi-stroke menus. Wave menus have the same performance as the novice mode of compound-stroke marking menus and as the expert mode of multi-stroke marking menus. By always displaying the parent menus and allowing interaction on all the displayed items, Wave menus also offer previsualization and path visualization, two features that increase the browsability and the learnability of menus by supporting exploration and menu knowledge acquisition. Even though the parent menus are displayed, Wave menus require less space than compound-stroke marking menus and can be used in "degraded mode" with restricted-space conditions (proximity to screen borders). In our experiment, we used a mouse because we are focusing on user acceptance of marking menus for common use. In a future comparative study, we plan to experiment with Wave menus with a stylus, which is an adequate device for marking interaction and cross-based

interaction. Finally in our experimental study we did not consider Zone and Polygon menus. Further research must be carried out to study their novice mode.

## References


1. Callahan, J., Hopkins, D., Weiser, M., Shneiderman, B.: An empirical comparison of pie vs. linear menus. In: ACM CHI Conference on Human Factors in Computing Systems, pp. 95–100 (1988)
2. Guimbretière, F., Martin, A., Winograd, T.: Benefits of merging command selection and direct manipulation. ACM Trans. Comput.-Hum. Interact, 460–476 (2005)
3. Howes, A.: A model of the acquisition of menu knowledge by exploration. In: ACM CHI Conference on Human Factors in Computing System, pp. 445-451 (1994)
4. Kieger, J.I.: The depth/breadth tradeoff in the design of menu driven interfaces. International Journal of Man-Machine Studies 20, 201–213 (1984)
5. Linton, F., Joy, D., Schaefer, A.: Building user and expert models by long term observation of application usage. In: Conference on User Modeling, vol. 3. pp. 129–138 (1999)
6. Kurtenbach, G. P.: The Design and Evaluation of Marking Menus. Doctoral Thesis. UMI Order Number: UMI Order No. GAXNN-82896., University of Toronto (1993)
7. Kurtenbach, G., Buxton, W.: User learning and performance with marking menus. In: ACM CHI Conference on Human Factors in Computing Systems, pp. 258–264 (1994)
8. Kurtenbach, G., Buxton, W.: The Limits of Expert Performance using Hiearchical Marking Menus. In: ACM CHI Conference on Human Factors in Computing Systems, pp. 35–42 (1993)
9. Lecolinet, E.: A molecular architecture for creating advanced GUIs. In: ACM UIST Symposium on User interface Software and Technology, pp. 135–144 (2003)
10. Norman, K.: The Psychology of Menu selection: Designing Cognitive Control at the Human/Computer Interface. Ablex Publishing Corporation (1991)
11. Pook, S., Lecolinet, E., Vaysseix, G., Barillot, E.: Control menus: excecution and control in a single interactor. In: ACM CHI Extended Abstracts on Human Factors in Computing Systems, pp. 263–264 (2000)
12. Rekimoto, J., Ishizawa, T., Schwesig, C., Oba, H.: Presense: Interaction techniques for finger sensing input devices. In: ACM UIST Symposium on User interface Software and Technology, pp. 203–212 (2003)
13. Sellen, A., Kurtenbach, G., Buxton, W.: The prevention of mode errors through sensory feedback. Journal of Human Computer Interaction 7(2), 141–146 (1992)
14. Shneiderman, B.: Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley Longman Publishing Co., Reading (1986)
15. Zhao, S., Agrawala, M., Hinckley, K.: Zone and polygon menus: using relative position to increase the breadth of multi-stroke marking menus. In: ACM CHI Conference on Human Factors in Computing Systems, pp. 1077–1086 (2006)
16. Zhao, S., Balakrishnan, R.: Simple vs. compound mark hierarchical marking menus. In: ACM UIST Symposium on User interface Software and Technology, pp. 33–42 (2004)