Visual Menu Techniques

GILLES BAILLY, CNRS LTCI, Telecom ParisTech, Paris-Saclay University
ERIC LECOLINET, Telecom ParisTech, Paris-Saclay University, CNRS LTCI
LAURENCE NIGAY, University Grenoble Alpes, CNRS, LIG

Menus are used for exploring and selecting commands in interactive applications. They are widespread in current systems and used by a large variety of users. As a consequence, they have motivated many studies in Human-Computer Interaction (HCI). Facing the large variety of menus, it is difficult to have a clear understanding of the design possibilities and to ascertain their similarities and differences. In this article, we address a main challenge of menu design: the need to characterize the design space of menus. To do this, we propose a taxonomy of menu properties that structures existing work on visual menus. As properties have an impact on the performance of the menu, we start by refining performance through a list of quality criteria and by reviewing existing analytical and empirical methods for quality evaluation. This taxonomy of menu properties is a step toward the elaboration of advanced predictive models of menu performance and the optimization of menus. A key point of this work is to focus both on menus and on the properties of menus, and then enable a fine-grained analysis in terms of performance.

Categories and Subject Descriptors: D.2.2 [**Design Tools and Techniques**]—User Interfaces; H5.2 [**User Interfaces**]—Graphical User Interfaces (GUI)

General Terms: Human Factors

Additional Key Words and Phrases: Menu techniques, command selection, shortcuts

ACM Reference Format:
Gilles Bailly, Eric Lecolinet, and Laurence Nigay. 2016. Visual menu techniques. ACM Comput. Surv. 49, 4, Article 60 (December 2016), 41 pages.
DOI: http://dx.doi.org/10.1145/3002171

1. INTRODUCTION

Menus are widely used for exploring and selecting commands in interactive applications. They first appeared in 1968 with the AMBIT/G system (implemented at MIT's Lincoln Labs) and were popularized by the Xerox Star in 1981 and the Macintosh in 1984 [Myers 1998]. Today, menus are widespread in current applications and are used by a large variety of users. As a consequence, menus have motivated many studies in Human-Computer Interaction (HCI): more than 60 new menu techniques were defined during the last 2 decades. Despite the number of studies on menus, designing effective menus remains a major challenge in HCI.

Menu design can be considered as an optimization problem [Lee and Macgregor 1985; Bailly and Oulasvirta 2014]. An "optimal menu" is the particular menu design that best meets all usability goals (e.g., speed) while respecting relevant constraints (e.g., screen space). As an optimization problem, we identify three challenges: the need

Authors' addresses: G. Bailly and E. Lecolinet, Telecom ParisTech, 45 rue Barrault, 75013 Paris, France; emails: {gilles.bailly, eric.lecolinet}@telecom-paristech.fr; L. Nigay, Laboratoire LIG, Bâtiment IMAG, 700 av. Centrale, 38401 Saint Martin d'Hères, France; email: laurence.nigay@imag.fr.

DOI: http://dx.doi.org/10.1145/3002171

to (C1) **characterize the design space** by defining decision variables; (C2) **develop predictive models of user performance** to evaluate the quality of a given menu design for a given set of constraints, and (C3) **implement optimization methods** that systematically explore the design space.

In this article, we focus on the first challenge of characterizing the design space of menus, as the two other challenges depend on this first step. Despite the apparent simplicity of menus, it is still difficult to have a clear understanding of the design possibilities, as well as to ascertain the similarities and the differences of existing menu techniques. Indeed, menu design is much more than just ordering commands in a hierarchy but involves numerous dimensions such as item characteristics, visual cues, menu layout, temporal considerations, menu shortcuts, and so on. Moreover, menu design must also tackle subtle interactions between menu features and the context of use.

To characterize the design space of menus, we propose a survey of visual menu techniques. A key point of this work is to focus both on menus and on their properties. It is essential to consider these two levels of granularity to precisely compare menus as a menu can verify several properties and a given property can be shared by several menus.

The core contribution of this article is a taxonomy of menu properties, which have an impact on menu performance. The taxonomy is useful for both the analysis, the design, and the selection of menu techniques. More precisely, we analyze the field of menus at a fine level of granularity to provide a synthesis of research on menus during these last decades and to highlight open research questions. We also provide foundations for future research on modeling and menu optimization by identifying a set of properties and their impact on performance. This can also help menu designers during the design process by informing them about relevant design decisions. Finally, this survey can help application designers to discover and choose the most relevant menus for a given application. An interactive tool (www.gillesbailly.fr/menua/) lists most of these techniques with additional information resources.

This article is organized as follows: We first provide definitions related to menus and discuss menu usages to fix the terminology and delineate the scope of our study. We then focus on menu performance by refining it through a list of quality criteria and by reviewing existing analytical and empirical methods for quality evaluation. We then present our taxonomy of menu properties. The taxonomy is organized along three dimensions: Menu System, Menu, and Item. Each menu property is illustrated by menu techniques of the literature. In light of menu properties, we finally discuss under-researched areas and open research questions.

2. MENU TECHNIQUES

Although expressions such as “menu,” “menu system,” or “menu technique” are widely used, there is no consensual definition of these terms in the literature. Indeed, several definitions have been proposed [Lee and Raymond 1993; Norman 1991; Jackoby and Ellis 1992; Barfield 1993; Marcus et al. 1994; Helander et al. 1997; Shneiderman 1992; Foley et al. 1990; ISO 1991]. For instance, a menu is defined as “a set of selectable options” in ISO [1998]. Some authors define menus by opposition to command lines (CLI) because they do not force users to memorize complex command sequences [Barfield 1993; Lee et al. 1993; Marcus et al. 1994; Shneiderman 1992]. We quote two complementary definitions from the literature on menu techniques:

Menu selection is a mechanism for users to indicate their choices. The characteristics of menu selection are that a) the interaction is, in part, guided by the computer; b) the user does not have to recall commands from memory, and c) user response input is generally straight forward.

Psychology of Menu Selection [Norman 1991]

Menus are a set of options, displayed on the screen, where the selection and execution of one (or more) of the options results in a change in the state of the interface
Handbook of Human Computer Interaction [Helander et al. 1997]

However, these two definitions remain somewhat general for defining menus precisely. As stated by Helander et al. [1997], defining the notion of a menu accurately is challenging. We propose four key characteristics:

1. Menus allow the user to select commands from a bounded set of items [ISO 1991, Foley 1990].
2. Menus provide a structure for presenting items visually. Items are generally organized in hierarchical groups or categories, which may be delimited by separators. Items can be organized alphabetically, numerically, semantically or in accordance with their frequency of use. Items are laid out according to a geometrical structure (linear, circular, etc.) [Jackoby and Ellis 1992; Dachselt 2007] that helps users to find the desired commands.
3. Menus are *transient* [Jackobsen et al. 2007]. In transient visualizations, information is temporarily displayed and is easily dismissed. Menus do not require permanent screen space because they appear on demand and are closed immediately after the selection of an item.
4. Menus are *quasimodal* [Raskin 2000]. Quasimodes are modes “that are kept in place only through some constant action on the part of the user” [Raskin 2000]. When the user activates a menu, the application enters into a specific mode until the end of the selection process.

In order to delineate the scope of our study, we will take the aforementioned four key characteristics for defining **menus**. Therefore, we will not consider certain interactors sometimes considered as menus, such as *comboboxes* and *option menus,* because they do not strictly adhere to our definition. These interactors are in fact hybrid objects that combine two components: an “anchor interactor,” which is permanently visible and a menu attached to this anchor. Moreover, a *menubar* is not a menu but an object that serves to open menus. While *palettes* are sometimes considered menus because they allow the selection of commands, we will not take them into account because they are not transient. Finally, in this article, we focus on command menus, thus excluding information search menus such as web catalogues and decision trees as in recommender systems. However, we believe that many aspects of this work might be applied to these interactors.

In addition to the term menu, we define the following terms:

—The **menu structure** is the graph of commands, which is generally a tree [Norman 1991].
—A **menu system** is a set of linked menus, such as hierarchical menus.
—The **current menu** is the menu with which the user is currently interacting.
—A **submenu** is a menu that can be accessed from an item of the current menu of a hierarchical menu system.
—The **super-menu** (or parent menu) is the menu that contains the item, which opens the current menu.
—A **menu panel** is the graphical user interface (GUI) widget that visually presents commands on the screen.
—A **menu technique** denotes an interaction technique [Appert 2004], which, combined with an menu panel, allows navigating in the menu structure and selecting commands in the graphs.

Having defined this terminology, we now discuss three factors affecting the use of a menu technique: user characteristics, menu behavior, and the context of use.

2.1. User Characteristics

How users interact with a menu depends on their perceptual, motor and cognitive abilities. In particular, it depends on their knowledge and the goals on which they focus on. Let for instance consider two extremes cases: On the one hand, an inexperienced user will have to navigate in a menu system and perform several ups and downs in the menu hierarchy before reaching the desired command. On the other hand, an experienced user will use keyboard shortcuts (hotkeys) to directly access frequent commands. To analytically study such usages, several models have been proposed to describe user knowledge and user goals when interacting with a graphical interface, as for instance Card [1983] and Rasmussen [1983]. Taking into account former work in this domain [Norman 1991; Waterworth and Chignell 1991; Grossman and Fitzmaurice 2009], we propose two orthogonal axes for the specific case of menus. The first axis focuses on user knowledge, and the second axis on the precision of the user's goal.

2.1.1. User's Profile. Three levels of user's profile can be defined considering their knowledge of the menu system [Shneiderman 1992]:

- *Neophyte users* do not know how to use an interaction technique and need to understand how it works. A technique must be easy to learn; otherwise, users may reject it, even if it is efficient after sufficient training.
- *Inexperienced users* know how the technique works, but they do not know the organization of the menu system. They are often occasional users who only use a limited number of commands (typically the most frequent ones).
- *Experienced users* are familiar with the interaction technique and the organization of the menu system. They typically are professionals that need rapid response time and brief, undistracting feedback. Even if they do not know the location of a new command in the hierarchy, thanks to experience, they are able to find it quickly.

2.1.2. Target Orientation. Target orientation is related to the cognitive state of users [Waterworth et al. 1991] who may or may not have a definite target in mind. In the context of menu selection, one can further distinguish between functionality and command search [Norman 1991].

- *Absence of a definite target:* this case occurs when users browse a menu system. For instance, when they discover and scan the menu hierarchy of a new application.
- *Functionality search*: users must find a functionality that satisfies their needs, but they do not know if this functionality exists nor its name.
- *Command search*: the target is clearly defined in the user's mind, who knows its name but ignores its location in the menu hierarchy.

2.1.3. Menu Usages. The user's profile and the target orientation of the user lead to different usages of the menus. Such usages correspond to different paths in the menu hierarchy to achieve goals [Howes 1994; Catledge and Pitkow 1995]. We distinguish three types of paths:

- A *roundabout path* is characterized by intensive exploration and navigation in the hierarchy with several ups and downs before the desired command is found. It generally occurs when users do not know the location of the desired functionality or command.
- A *straight path* is a path without detour: the user does not visit accidental wrong submenus. It generally occurs when users know the location of the desired command

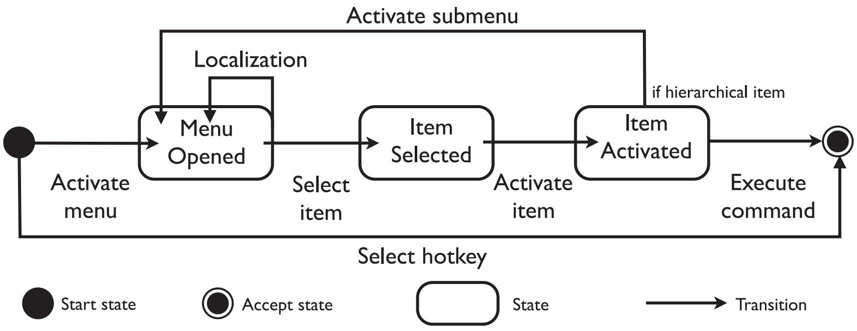

Fig. 1. Command execution process in a menu system.

or when they do not know this location but the menu titles allow them to find the desired command.

- A *direct path* occurs when users select a command without going through submenus. This is the case for keyboard or stroke shortcuts,[1] as they provide a direct access to commands.

2.2. Menu Behavior

Facing the different user characteristics and thus the different usages of a menu, we now describe the behavior of a menu technique. Its features (appearance, layout, etc.) will be discussed in the taxonomy of Section 4.

The behavior of a menu technique can be described by a flow chart (Figure 1). The first transition, named *activation*, consists of opening the menu from an object of interest such as a menu bar or another interactor. Once the menu is opened, the user *localizes* the desired item visually then *selects it* by moving the mouse or by pressing arrow keys. The selection is generally highlighted. Finally, the user *activates* this item and executes the corresponding command, by clicking the mouse or pressing the space key. For the case of a cascaded menu (i.e., when the item has an attached submenu), the user continues to navigate in the menu system and the automaton comes back to the "Menu Opened" state. All these steps are involved when using the novice mode of the menu. They are bypassed in expert mode as shown in Figure 1 (bottom). We now detail both modes.

Novice mode. The novice mode is based on "recognition" [Lee and Raymond 1993]: Users can explore the menu and recognize the desired command. The interaction is generally performed with the mouse or the arrow and space keys of the keyboard. Some graphical toolkits also provide *mnemonics* (e.g., on Windows): The users just have to type the underlined letter of one item to activate it when the menu is opened (Figure 2).

Expert mode. The menu does not appear on the screen in this mode, which is based on "recall," hence forcing users to make some effort to learn how to activate commands [Lee and Raymond 1993]. Its purpose is to provide faster interactions that let users focus on their task, as the menu content is not displayed. Depending on the type of menu, items are either activated by keyboard shortcuts (also called hotkeys) or gestures

[1]Hierarchical Marking menus [Kurtenbach et al. 1993] require users to draw an inflected path (one inflexion for each menu level). This path is not a direct path from a graphical point of view, but it can be interpreted as a whole and integrated as a unique "chunck"/action by users [Buxton 1995; Zhao et al. 2004]. However, this assertion is harder to defend for some variants such as Multi-Stroke menus [Zhao et al. 2004] where users must perform a series of simple marks.

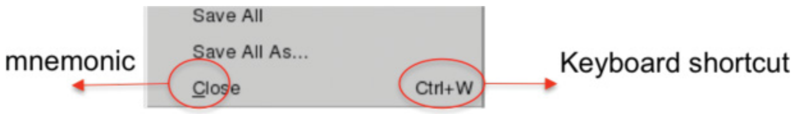

Fig. 2. Mnemonics and keyboard shortcuts in linear menus.

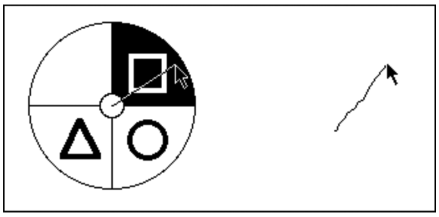

Fig. 3. Novice mode (left) and expert mode (right) of **Marking menus** [Kurtenbach et al. 1991] © ACM.

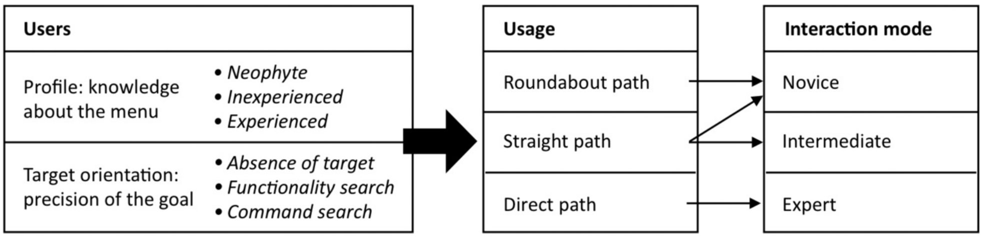

Fig. 4. Modes of a menu according to users' profile and their goals.

(stroke shortcuts). A label reminding the shortcut is often displayed close to the item name, as shown in Figure 2.

Stroke shortcuts were introduced in Marking menus (Figure 3) [Kurtenbach et al. 1991]. **Marking menus** have a circular layout. They are only displayed (novice mode) if the user waits for a fraction of a second (about 100–300ms) once the menu is activated. An important feature of this technique is that gestures are similar in novice and expert modes. This property contrasts with keyboard shortcuts that require users to use a different modality in expert mode (see Section 4.5.1)

Intermediate mode. Hierarchical circular menus can allow an intermediate mode called "combined" or "mark confirmation" mode [Kurtenbach 1993]. This case happens when users start in expert mode and finish in novice mode because they do not recall the gesture corresponding to a desired command. Some linear menus allow a similar property: users first press the CTRL key to display a visual help, then select the proper hotkey (**ExposeHK** [Malacria et al. 2013]).

As shown in Figure 4, the three modes of a menu (novice, intermediate, expert) enable us to support the different usages defined by the user's profile and the precision of her/his goal (Section 2.1). Indeed, the novice mode allows a *roundabout interaction path*: the novice and intermediate modes support a *straight path,* while the expert mode provides a *direct path*.

2.3. Context of Use

When choosing a menu, the designers should consider the context of use and in particular the types of tasks of the application as well as the set of devices where the applications will run.

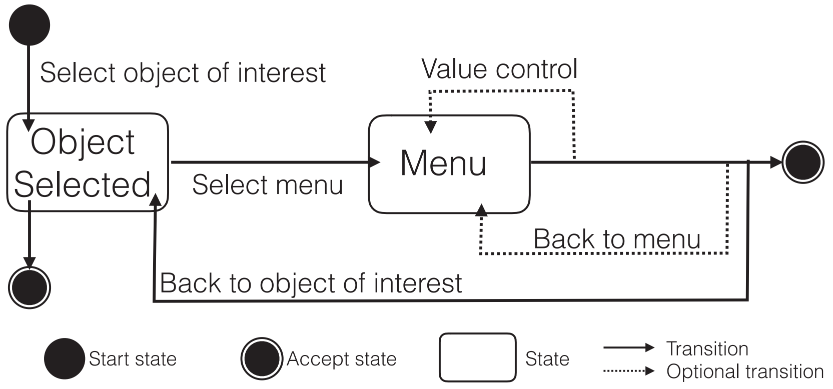

Fig. 5. Command selection process between the menu and the objects of interest. We now detail these transitions.

2.3.1. Task. Menus are not the main objects of the task—they are instruments intended to help users to accomplish a goal. It is thus important to understand how the menu is integrated into the context of the primary task. To accomplish such tasks, users generally select one object of interest, open the menu, select commands, and control their parameters. The performance of a menu thus depends on this interaction sequence [Appert et al. 2004; Mackay 2002]. Dillon et al. [1990] claim that "the true cost of command selection includes both the movement to the menu and the movement back to the main task." We thus propose a model with two supplementary transitions (Figure 5):

- Menu selection (object-to-command transition)
- Back to object of interest (command-to-object transition)
- Back to menu (command-to-command transition)
- Value control (command-to-value transition)

Menu selection. Menu selection (also called "object-to-command transition" [Cance et al. 2006]) corresponds to the distance (or time) for accessing the menu from the object of interest. This transition generally consists of pointing to an anchor such as a menu bar to activate the menu. In contrast to menu bars,[2] context menus (also called "in-place menus" [Hinckley et al. 1999]) can directly be manipulated on objects of interest.

Back to object of interest. Users generally need to come back to the location of the selected object(s) of interest after activating a command [Cance et al. 2006]. For context menus, this location generally corresponds to the activation point. Two strategies have been proposed to reduce the average distance between items and the activation point. The first one consists of using a stack layout (**Multi-Stroke menus** [Zhao et al. 2004]) that prevents the cursor from moving too far away from the activation point during the exploration of the menu. As in **Slippery menus** [Cance et al. 2006], another solution consists in *sliding* the menu under the cursor instead of moving the cursor over the menu so that the cursor remains on the object of interest at the end of the interaction of the user.

Command–to-command transition. Users often need to apply several commands to one object or one command to several objects: This implies many "object-to-command" and "command-to-object" transitions. Reducing these transitions improves

[2]An exception with the MacOSX menu bar that uses the border of the screen as an *impenetrable border* (see Section 4.2.1) in order to quickly access menus.

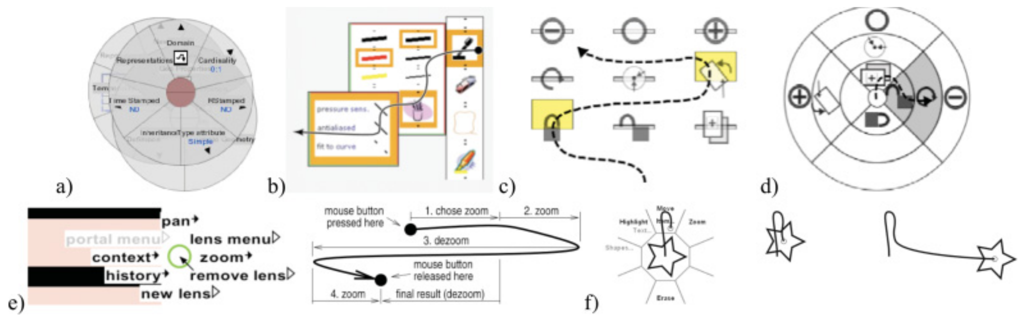

Fig. 6. (a) Users can make the submenus persistent with the Floating Pie menus [Rubio and Janecek 2002]. (b) CrossY [Apitz et al. 2004], (c,d) Grid gates and Polar gates [Sulaiman et al. 2008] allow users to compose several commands by performing a single gesture. (e) Control menus [Pook et al. 2000] allow users to select a command and control a parameter in the same gesture. Users must go through a threshold distance from the activation point to start controlling parameters. (f) Flow menus [Guimbretiere et al. 2000] rely on a similar principle except that leaving the central zone activates the command while re-entering the central zone starts the operation.

interaction. For instance, palettes let users select a mode or a command that can then be applied multiple times without having to return to the palette. However, mode persistence [Raskin 2000] tends to be error prone because users may forget to change the current mode when performing an action a while later. Alternately, submenus can be made persistent by "detaching" them (**Tear-off menus**) or keeping them opened after activation (**Floating Pie menus** [Rubio and Janecek 2002]). Moreover, some menus allow users to activate several commands in sequence by performing a single gesture (Figure 6). They leverage specific menu layouts, which can be linear (**CrossY** [Apitz et al. 2004]), according to a grid (**Grid Gates** [Sulaiman et al. 2008]) or circular (**Polar Gates** [Sulaiman et al. 2008]).

Value control. Some commands require specifying parameters (command-to-value transition). This is typically done by opening a dialog box once the command is selected. In contrast, some menus allow selecting and controlling a command in a single gesture, a characteristic called "merging" in Guimbretiere et al. [2000] Parameter control starts by moving further than a certain threshold from the activation point with **Control menus** [Pook et al. 2000]; Figure 6(e), by (re-)entering a specific area with **FlowMenus** [Guimbretiere et al. 2000]; Figure 6(f), or by tilting the pen with **Tilt menus** [Tian et al. 2008].

2.3.2. Devices. This section reviews various devices and modalities that can be used for interacting with a menu.

Input modalities. Menus are often manipulated by using a relative and indirect 2D input device such as a mouse or a touchpad (or a keyboard in the case of mnemonics and hotkeys). Figure 8 shows examples of menu techniques relying on other input modalities such as pressure (Figure 8(a)), multi-finger input (Figure 8(b)), gloves (Figure 8(d)), or eyes-tracking [Kammerer et al. 2008]. These alternate modalities can serve to increase the number of commands, to improve performance, or to compensate the lack of input resources. While now quite common, it is worth noticing that touchscreens, lack some operating states (e.g., hover) because of the absence of mouse buttons [Buxton 1990]. Touchscreens raise several problems when interacting with menus:

- *Menu activation*. Opening context menus on touchscreens often rely on a delay, which acts as a common substitute for the right mouse button. Other solutions include multiple taps [Wu and Balakrishnan 2003], multiple-finger taps (two-finger taps on

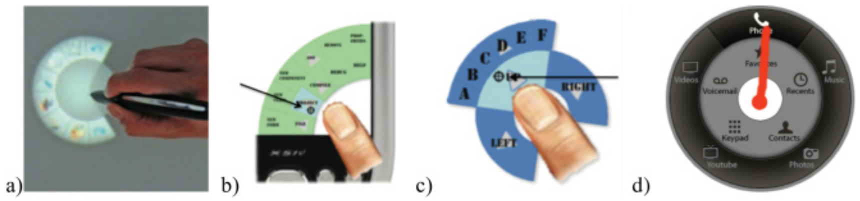

Fig. 7. (a) **Occlusion aware menus** [Brandl al. 2009]; some items are left empty to avoid occlusion. (b) **Arch Menu** and (c) ThumbMenu [Huot et al. 2007] use an offset cursor to improve finger selection accuracy with small items. (d) Wavelet menus [Francone et al. 2010] place submenus at the center of the menu to fit small screens without degrading navigation.

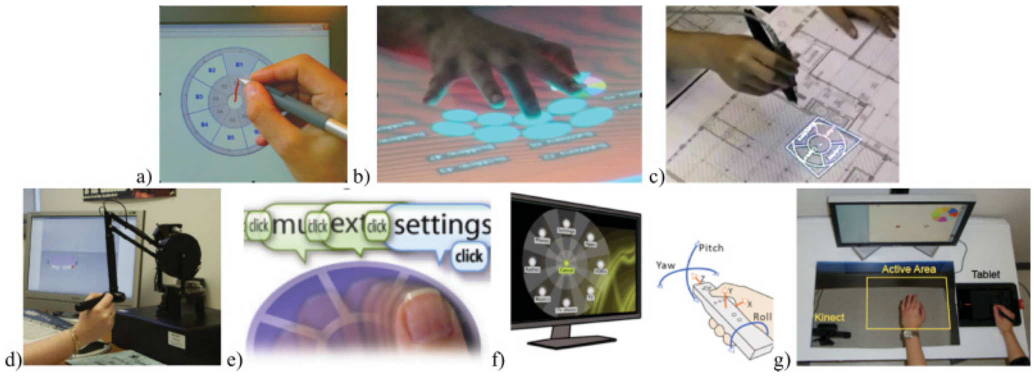

Fig. 8. Menus using non-traditional input/output capabilities. (a) **Push menus** use pressure input [Huot et al. 2008]. (b) **MTM** uses multi-finger inputs [Bailly et al. 2008]. (c) Marking menu using a pen with a pico-projector [Song et al. 2009], © ACM. (d) **HardBorder** uses haptic feedback [Essert-Villard and Capobianco 2009]. (e) **EarPod** uses audio feedback [Zhao et al. 2007]. (f) TV Menus using a gesture-aware remote control [Bailly et al. 2011]. (g) Two-handed Marking menus are a near-surface menus [Guimbretière and Nguyen 2012].

the iPhone and Macintosh touchpads, five-finger taps [Lepinski et al. 2010]), taps with the heel of the hand [Bailly et al. 2008], dedicated buttons [Hinckley et al. 2006], or Microrolls gestures [Roudaut et al. 2009].

- *No keyboard shortcuts*. Stroke shortcuts can serve as a substitute for hotkeys on devices that do not have a hardware keyboard [Bailly et al. 2010; Kurtenbach 1993; Roudaut et al. 2009].
- *Occlusion*. Linear menus generally appear in the bottom right area of the activation point, which is often masked by the user's hand (for right-handers). Occlusion introduces fatigue and inhibitory movements [Hancock and Booth 2004]. The default position of the menu can be changed statically [Roudaut et al. 2009] or dynamically [Hancock and Booth 2004] to solve this problem. Alternately, some items can be left empty [Brand et al. 2009] (Figure 7(a)).
- *Insufficient accuracy*. Interacting with fingers can produce imprecise item selection [Parhi et al. 2006]. A trivial solution increases the size of items, but this is at the cost of wasting screen real estate. A review of various solutions can be found in Roudaut et al. [2008]. Among these solutions, only the "Offset Cursor" mechanism[3] [Potter 1988; Huot et al. 2007] (Figure 7(b–c)) and gesture input [Roudaut et al. 2009] have been applied to menus.

[3]Offset Cursor [Potter 1988] displays a cursor above the point where the user touches the screen to avoid finger occlusion. It is efficient, but slow and loses the directness of interactive screens [Roudaut et al. 2008].

Output modalities. While audio feedback [Zhao 2008] (Figure 8(e)) and tactile feedback [Essert-Villard and Capobianco 2009] (Figure 8(d)) have been proposed as output modalities for menus, visual feedback largely remains dominant. But the amount of screen real estate needed for displaying menus can be a major limitation, particularly for mobile devices. This can make interaction cumbersome and even impossible in certain cases. The most common solutions use scrollbars or translate menus when they are close to a border of the screen to prevent them from being truncated. The position of each menu item can also been decided by the user by drawing a path such as in **User-drawn context menus** [Leithinger et al. 2007]. The lack of screen real estate can constrain the use of certain menu layouts. For instance, hierarchical Marking menus require too much space for mobile devices[4] [Zhao et al. 2004; Bailly et al. 2007]. Solutions such as **Wavelet menus** [Francone et al. 2010] have been proposed to solve this problem (Figure 7(d)). We will discuss the challenges related to menu layout in more depth in Section 4.

To sum up, the contributions of this section were (1) to propose a definition based on four characteristics extracted from the literature and our experience and (2) to analyze menu interaction by considering users' characteristics, menu behavior, and the context of use.

3. MENU PERFORMANCE

The main objective of menu designers is to improve the *performance* of menus given an application, a platform (i.e., devices), and/or a class of users. However, what performance means is not always precisely defined. Moreover, it may be hard to predict or measure the quality of a specific technique. Conducting user studies is common practice in HCI to get insights about the performance of an interaction technique. However, such studies generally involve a limited number of users and a small set of tasks. Thus, they may not take into account all possible factors. Moreover, there is no established list of criteria, tasks or benchmark scenarios to guide empirical evaluations and to precisely define the scope of the results.

The goal of this section is to better understand the different aspects of menu performance. This will be useful in the next section to better understand the impact of menu properties on performance. Toward this goal, we first present the framework of interface expertise proposed by Scarr et al. [2012] that we apply to the context of menus. We then describe analytical and empirical methods to evaluate the performance of menus.

3.1. Framework of Interface Expertise

The framework of interface expertise [Scarr et al. 2012] (Figure 9) attempts to characterize the development of user performance over time, that is to say, given a context of use (i.e., application, platform, class of users), how user performance improves from novice to expert mode. In this article, the novice mode relies on menus (first modality) and the expert mode on the second modality such as keyboard or gesture shortcuts.

Three stages can be observed regarding performance in novice mode: *initial performance*, *extended learnability,* and *ultimate performance*. *Initial performance* concerns neophyte users who never used the menu before (Section 2.1). Ideally, these users should be able to easily and quickly understand how the menu works. *Extended learnability* refers to how fast users reach their ultimate level of performance. Indeed,

[4]Hierarchical Marking menus require much screen real estate to display submenus and thus is a poor candidate for small screens. For instance, a three-level Compound Marking menu requires more horizontal space than 10 linear one-level menus [Bailly et al. 2007] because (1) circular menus are at least two times larger than linear menus, (2) circular menus generally appear centered around the activation point, and (3) they require space on both sides to display submenus.

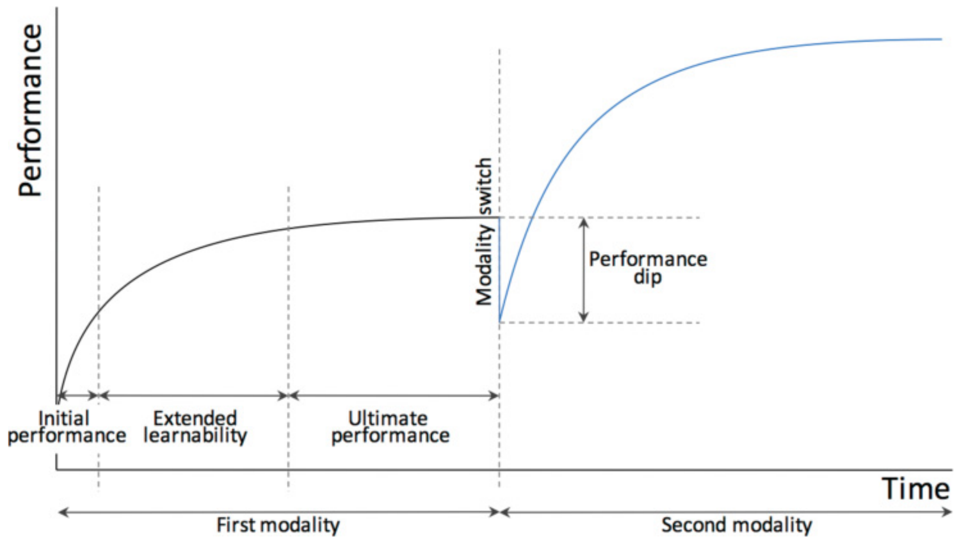

Fig. 9. Framework of interface expertise [Scarr et al. 2012], showing the transition from the menu (first modality) to the second modality (expert mode).

strategies favoring incidental or explicit learning accelerate learning. Finally, *ultimate performance* (or performance ceiling) corresponds to a level of performance, which cannot be exceeded.

This framework also characterizes intermodal expertise development. It highlights the *performance dip* when users switch to a new modality, even if it ultimately offers a higher performance ceiling. This notion is important because it explains why many users (and even experienced users) do not switch from the first modality (menu) to the second modality (shortcuts) [Lane et al. 2005]. In practice, users often favor short-term productivity rather than long-term productivity. If users perceive the expert mode as difficult to learn, they may reject the technique and thus never attain the performance ceiling it enables [Scarr et al. 2012].

Based on this framework, Malacria et al. [2013] identified several factors that can favor the transition from novice to expert mode:

- *Awareness of other modalities*. Users are often not aware an expert mode exists. For instance, the study of Grossman et al. [2007] suggests that hotkeys are ignored by some users.
- *Perception of performance*. Scarr et al. [2011] note that the users' perception of the potential gain they may obtain by using a new modality strongly influences their decision of whether or not to switch. While hotkeys offer a higher-performance ceiling than menus [Lane et al. 2005; Odell et al. 2004], Tak et al. [2013] found that some participants did not use hotkeys because they believed using toolbar buttons would be faster.
- *Lack of motivation*. Finally, there are many elements of motivation at play in a user's decision to switch to the expert mode. Users can be motivated to improve their performance (intrinsic motivation) or be influenced by social factors (extrinsic motivation).

In summary, this framework characterizes the *development* of user performance with menus by revealing different stages (initial performance, extended learnability, and ultimate performance) for both the novice and expert modes. We now aim at

Table I. The Two factors, Usability and Applicability, and Their Corresponding Criteria

Factor	Criteria
Usability. Adequacy of menu systems for the cognitive, motor, and sensory abilities of the user	**Speed and accuracy.** Menu efficiency for selecting commands
	Learning and memorization. The capacity of the menu to allow the user to use it in an optimal way quickly and in the long term
	Satisfaction. The capacity of the menu to provide a pleasant feeling that results from the fulfillment of what the user wants
Applicability. Adequacy of menu systems for the user's needs that are related to the application	**Application adequacy.** Capacity of the menu to become integrated into applications
	Device adequacy. Capacity of the menu to adapt different input and output devices
	User adequacy. Capacity of the menu to adapt to different classes of users

While usability is mainly related to menu usage, applicability has a more functional connotation.

better defining menu performance by discussing criteria that can be used for analytical evaluations.

3.2. Analytical Evaluation

Analytical evaluation methods do not require user testing, which is expensive and time-consuming. These methods include heuristics, cognitive walkthrough, literature review, model-based evaluations, and so on [Shneiderman 1992]. In the following text, we discuss performance criteria and predictive models of performance in the case of menus.

3.2.1. Factors and Criteria. Our performance criteria are organized as a hierarchical set (Table I) inspired from the multi-criteria analysis from [McCall 1977; Abowd et al. 1992] and several other studies [Card et al. 1990; Bernsen 1996; Chuah et al. 1996; Beaudouin-Lafon 2000; Dachselt 2007; Shneiderman et al. 1992; Bastien et al. 1993; Nielsen 1993]. At the first level of the hierarchy, *usability* and *applicability* are two factors referring to the practical acceptance of a menu technique by users.

Usability is related to menu usage. It refers to the adequacy of the menu technique from the perspective of the cognitive, motor and sensory abilities of the user. Several definitions have been proposed to characterize usability [Welie et al. 1999]. We adopt the definition given by Shneiderman [1992], which is based on the five following criteria: Speed, Accuracy, Learnability, Memorization (retention over time), and Satisfaction. We will consider speed and accuracy together because they are strongly related (for instance, faster execution generally produces more errors). This is also the case for learning and memorization, which both depend on usage frequency. The satisfaction criterion is particular because it is a subjective measure that is directly linked to the comfort and the acceptability of use. This criterion is also related to the previous ones: for example, a user can like a menu because it is fast.

Applicability. Applicability has a more functional connotation and can be seen as an adaptation of Nielsen's utility factor [Nielsen 1993] for menus. Unlike the previous factor, applicability focuses on the adequacy of the menu technique with respect to the application, the devices and the users. For instance, menus that only allow accessing a limited number of items are not adapted for applications having a large number of commands. Similarly, techniques using a large amount of screen real estate are not well suited for smartphones. Finally, menus requiring precise movements are not adapted for elder users [Worden et al. 1997] or users with motor impairment. These three examples illustrate the three criteria we propose for applicability: application adequacy, device adequacy, and user adequacy.

While the usability and applicability criteria are related, they do not correspond to the same point of view. For instance, linear menus are not well suited for a smart device running application because the user would have hard times pointing to the right item while running. However, this does not mean that the usability of linear menus is mediocre in general but just that they are not well suited for this kind of application on this kind of device. Applicability thus serves to take into account this dependency on the context.

Example. Let us assume that designers want to integrate a Marking menu in their applications. They are aware that usability can affect software popularity, but they do not have the time and money to conduct empirical user studies. Going through the aforementioned proposed set of criteria will provide them with a better understanding of the possible advantages and drawbacks of Marking Menus:

- *Speed and accuracy*. Marking menus are fast because they reduce the average distance to select items. Additionally, marks can be performed at any size reducing the need of precision.
- *Learning and memorization*. Marking menus exploit spatial memory and can highlight semantic relationships between commands. For instance, some opposite commands ("Open" and "Close") can be assigned to opposite gestures. Moreover, Marking menus provide a fluid transition from novice to expert usage because users perform the same gestures in novice and expert mode.
- *Satisfaction*. The circular design of Marking menus is esthetic, and gestural interaction is known to be pleasant.
- *Application adequacy*. Marking menus can hardly contain more than 8 or 12 commands at each menu level for expert users while several categories of the application contain more than 12 commands.
- *Device adequacy*. The application has been developed for the desktop but will be probably deployed on mobile devices. Unfortunately, Marking menus are too large (more than two times larger than Linear menus), which is not compatible with mobile device small screens.
- *User adequacy*. Marking menus are adapted to users with motor impairment, as they do not require precise pointing abilities [Harada et al. 2007].

This example shows that trade-offs can be put into evidence by considering criteria in a systematic way. While Marking menus are efficient for the usability factor, they are not well suited for small screens or when numerous commands are needed. For application designers, these criteria can be used to choose between existing menus. For researchers, it can open novel research directions to overcome existing limitations. For menu designers, it can be used as guidelines for applying a systematic multi-criteria analysis of the menu during the design process.

3.2.2. Models of Performance. Models of menu performance synthetize phenomena by quantifying the impact of menu properties on users' behavior. They provide an efficient way of encapsulating scientific knowledge, and they avoid carrying out extensive user trials [Cockburn et al. 2007; Bailly et al. 2014]. While several models have been proposed for interaction techniques (e.g., GOMS [John et al. 1996], CIS [Appert et al. 2004]), few of them focus on menus. We first present existing models of menu performance. We then discuss limitations and possible directions.

Mathematical models. Total Search Time (TST) is probably the first model of menu performance [Lee and MacGregor 1985]. It predicts search/selection time for balanced hierarchical linear menus (also called symmetric menus), that is, menus with constant menu breadth, for a given menu breadth (b), processing time per option (t), user response time (k), computer response time (c), and menu system depth (d). According to this model, time is summarized as:

$$\mathrm{TST} = (\mathrm{b} * \mathrm{t} + \mathrm{k} + \mathrm{c})\ \mathrm{d}$$

This model has several limitations. It assumes that all menu items are examined in a linear fashion, have the same popularity and the same pointing time. Some variants of this model have been proposed to consider more complex localization strategies (self-terminating, i.e., non-exhaustive search) [Lee and MacGregor 1985; Pap and Roske-Hofstrand 1986] to take into account the level of experience of the user or the semantic organization of the menu.

Liu et al. [2002] presented the GS menu model, which is based on the Guided Search (GS) model [Cave and Wolfe 1990]. The GS model is a model of visual search from of the perceptual literature that quantitatively describes the role of parallel and serial processing in visual search. The GS menu model predicts item selection time as a function of (1) item frequency, (2) item length, and (3) item color. However, in common menus, the color of items do not vary. Moreover, it does not consider important aspects such as the level of practice or the location of the items in the menu. .

Cockburn et al. [2007] proposed the Search-Decision-Pointing model (SDP model) [Cockburn et al. 2007]. The SDP model is a regression model using four variables and seven parameters to predict total selection time. The four predictors are number of items, item frequency, spatial, and item position. The model relies on five components:

1) The **S**earch component predicts that the time to localize a command increases linearly with the number of commands in the menu, but is independent of the item location.
2) The **D**ecision component predicts that the time to decide from among commands depends on the "entropy" of their relative frequencies in previous selections. It is given by the Hick-Hyman law [Hick 1952; Hyman 1953].
3) The **P**ointing component is based on Fitts' law [Fitts 1954] and predicts that commands closer to the top are faster to select. It assumes that the mouse does not move during the localization of the target. The mouse is thus on the top of the menu when starting the pointing task.
4) An expertise scalar modulates the components by the number of repetitions with an item. With practice, performance shifts from being dominated by search (linear) to decision (logarithmic).
5) Finally, the model includes a scalar "predictability" variable (1: unchanging, 0: constantly changing order) that predicts a detrimental effect of spatial consistency.

The SDP model has also been generalized to long lists (i.e., when a large part of the list is not visible on the screen) [Cockburn and Gutwin 2009]. The authors showed that selection time depends on the ability of users to anticipate the location of the target. When users can anticipate the location of the target, selection time is best modeled by functions that are logarithmic with respect to the length of the list. When users cannot anticipate the location of the target, selection time is best modeled by functions that are linear.

Finally, the SDP model has also been generalized to grid layouts (Square menu) and circular layouts (Pie menu) [Ahlstrom et al. 2010]. The authors showed that the pointing component depends on the layout of the menu. Indeed, pointing time linearly increases with the number of items of circular menus. However, they used Pie menus with an uncommonly large number of items (more than 36, while Pie or Marking menus rarely contain more than 8–12 items), and the diameter of the Pie menus was not constant but increased with the number of items.

Simulation models. Simulation models are the second class of models. They mainly focus on the cognitive processes of visual search to explain the total time. We present three models: EPIC, ACT-R/PM and a computational rational model of menu search.

Executive Process Interactive Control (EPIC) [Kieras and Meyer 2007] consists of a production-rule cognitive processor and perceptual motor peripherals. Applications to menu performance encode search strategies as production rules. Their execution is limited by temporal and capacity limitations posed by the cognitive infrastructure [Halverson 2008; Hornof and Kieras 1997, 1999]. Four strategies are distinguished: serial search (one menu item processed at a time), parallel search (many items processed at the same time), random search, and systematic search. The last two are combination of the others. Data suggested that parallel search with both random and systematic search well matched observed data. For mouse control, EPIC predicts that there should be a single aimed mouse movement from the initial position to the target item once that target item has been located.

ACT-R/PM [Byrne 2001] extends ACT-R [Anderson and Lebiere 1998], which is also a production rule architecture. ACT-R/PM posits a systematic, top-to-bottom search with eye fixations on menu items that share features with the target item. Moreover, ACT-R predicts that the mouse should "trail" the eyes such that once the target item is located, there is an approximately constant and short distance to the target. This predicts multiple mouse movements that are correlated with the number of eye movements.

Chen et al. [2015] have recently proposed a model of visual search in linear menus. It predicts search time and eye movements from assumptions about the user's task environment and cognitive limitations. Assumptions are about saccade duration, fixation duration, and peripheral vision. In contrast with EPIC and ACT-R/PM, this does not require the modeler to hand-code production rules. The user strategies derive from a principle of rationality to (1) the structure of interaction, (2) cognitive and perceptual limits, and (3) the objective to maximize the trade-off between speed and accuracy (e.g., utility). The model relies on Q-learning, a model of learning reinforcement.

Toward a unified model of menu performance. Bailly et al. [2014] proposed a model of menu performance called VSST that unifies the ACT-R and EPIC models. VSST assumes a serial search component such as ACT-R and a directed search component, which refines the random search component from EPIC or SDP. Directed search consists of a direct attempt at moving the eyes on top of the target. At first, such attempts are random, as users try to guess the location, but with more exposure they become more accurate. Moreover, this model also proposes a pointing component based on Fitts' law. It also assumes two starting locations of the cursor: The first at the top of the menu (such as SDP or EPIC), the other from a constant distance in the vicinity of the target (such as ACT-R). VSST also captures the change of performance with menu organization (alphabetic, semantic and random). Finally, this model not only predicts menu and target selection time but also the gaze distribution (i.e., where users are looking at).

3.2.3. Discussion. Simulation models (e.g., ACT-R/PM) explain the progression of search by reference to underlying cognitive processes, such as perception, attention and memory. Mathematical models (e.g., SDP) expose fewer details about the process but may be more straightforward to apply than simulation models. These models have never been empirically compared, and it is not clear which one is the best approach. However, VSST made an effort to cover phenomena that were not covered by the previous models. Maybe more important according to the focus of this article, these models rely on a small number of *menu properties*: they do not consider the impact of the semantics or of visual cues such as separators, icons, and the like. Moreover, they are limited to the novice mode. Considering the expert mode (keyboard and gesture shortcuts) as well as the transition between these modes (see Section 3.1) is equally needed. Finally, these models assume that performance means “Time” and do not consider other performance metrics such as user satisfaction. We argue that the main reason is that menu properties are not well identified and organized. In other words, the design space of menu properties is not sufficiently characterized, which is the first step to elaborate advanced models of menu performance [Bailly et al. 2014]. This is why we present a taxonomy of menu properties in Section 4.

3.3. Empirical Evaluation

Despite their utility, existing predictive models of menu performance are not mature enough to avoid the need of empirical evaluations to fully validate a menu technique. In the following text, we focus on the comparative evaluation of such techniques. However, it is worth noting that some studies focus on the impact of certain factors on performance such as menu length or menu structure (see Cockburn and Gutwin [2009] for more details).

3.3.1. Class of Experiments. Very few field studies have been conducted to compare menus. One exception is a field study that compared Marking menus and Linear menus during 29 days using a real application [Kurtenbach et Buxton 1994], but the number of commands (4) and users (2) were limited. In contrast, many menu techniques have been compared through controlled environments. Most experiments required around 12 participants for about 1 hour. In the following text, we detail different types of experiments.

Performance in novice mode. Most linear menus have been evaluated in novice mode [Ahlström et al. 2005, 2006; Cockburn and Gin 2006; Tsandilas and Schraefel 2007; Tanvir et al. 2008]: Users already know the location of the target item (which is generally highlighted), and they must select it as fast and accurately as possible. Users thus follow a straight path (Section 2.3.3) in order to investigate the performance ceiling with the novice mode (Section 3.1). In the case of hierarchical menus, parent items are also highlighted to simulate experienced users searching a command. A trial is terminated as soon as a participant activates an item (either wrong or correct). The content of the menu is arbitrary (familiar words). Only a subset of the available items are tested due to the large total number of items and the conventional 1-hour limitation. This kind of experiment has two major practical advantages. Trials are short and do not require cognitive efforts as visual cues guide participants to select the target. The experiment also avoids potential problems related to the familiarity of users with the menu.

Navigation in novice mode. Few experiments focused on navigation. They simulate inexperienced users searching a functionality (rather than a command) and thus visiting several submenus before finding the target (roundabout path; Section 2.3.3). This kind of experiment involves two problems. First, performance highly depends on user experience, thus introducing a high level of variability between participants.[5] Second, the task is tiring because participants must constantly evaluate the relevance of the current item according to the target. Two strategies have been proposed to control the amount of exploration in the hierarchy of items. The first one dynamically displays the target in the *n* visited submenu [Appert et al. 2006]: participants then visit n-1 menu distractors before having the opportunity to find the target. The second places N occurrences of the same item in the menu hierarchy and asks participants to select all

[5]The quantity of exploration in the menu hierarchy depends on the quality of the organization, the category titles, as well as user experience and their strategies.

of them [Bailly et al. 2007]. Some menus were disabled to reduce the total amount of visited submenus.

Immediate usability. Experimenters generally explain how the technique works before starting the experiment. In contrast, some experiments investigate initial performance. The task then consists of selecting few items without previous explanations [Francone et al. 2010; Walter et al. 2014]. These experiments generally provide qualitative rather than quantitative results.

Performance in expert mode. Several experiments evaluated the performance ceiling (Section 3.1) of Marking menus in expert mode [Kurtenbach et al. 1993; Balakrisnan et al. 1998; Zhao and Balakrisnan 2004; Zhao et al. 2006; Kin et al. 2011]. They simulate expert users' behavior by only investigating motor control performance (direct path; Section 2.3.3). Users reproduce a mark that is shown on the screen. This avoids a long learning phase that would be not compatible with a laboratory study. This class of experiment is easy to conduct and does not require cognitive efforts for participants. However, results can be difficult to generalize as many users never switch to the expert mode or only use it for a very small number of commands [Lane et al. 2005].

Memorization performance. In comparison, a relatively small number of studies focused on learning and memorization. We distinguish experiments focusing on *intentional* and *implicit* learning. Bailly et al. [2008] compared the *intentional* learning of the expert mode of several menu techniques. The experiment was divided in two phases. In the first phase (training phase), the participants could select as many commands they wanted during 5 minutes, either in novice or expert mode. In the second phase (testing phase), they were asked to select commands only in expert mode. The dependent variable, recall, was the percentage of items correctly selected in expert mode. An improvement consists in alternating the training and testing phases several times [Bau and Mackay 2008; Bailly et al. 2012]. This scheme provides information about the evolution of learning over time. One possible difficulty of this class of experiment is to distinguish between motor control error (false execution), recognition error (limitation of the recognizer/technology) and recall error. To precisely estimating recall error, it is thus necessary to formerly estimate both motor control and recognition errors.

Experiments focusing on *implicit* learning let participants choose whether they want to select commands in novice or expert mode. Users generally start in novice mode. Then, they continue in expert mode when they are confident enough to save time. If the experiment is long enough, participants reach their ultimate performance in expert mode. As participants are not instructed to use the expert mode, it happens that up to 50% of the participants did not make the effort to learn it [Grossman et al. 2007]. A variation of this design *explicitly* instructs users about the availability of the expert mode [Roudaut et al. 2009], but they are free to choose the modality.

3.3.2. Discussion and Guidelines. The previous paragraphs showed that different experimental designs have been proposed to evaluate different aspects of menu performance. Our goal is not to suggest a universal protocol for menu evaluation (which is probably impossible due to the inherent complexity of menus and the constraints related to laboratory studies) but to make trade-offs explicit in order to thoroughly understand the remit of the results. We now summarize the main questions that can help to design a menu experiment:

1) *Who are the target users?* We have seen in Section 2 that there are several ways of using menus and that they are related to user knowledge, user goals, and the development of user expertise [Cockburn et al. 2014]. Does the experiment focus

on immediate usability, novice mode, expert mode or the transition from novice to expert mode?

2) *What is the context of use?* Most experiments do not measure the “true” cost of command selection [Dillon et al. 1990] because they do not measure time to access the menu or to come back to the objects of interest or the cost of errors. In contrast, some experiments consider the task in which the menu is involved [Appert et al. 2004; McGuffin et al. 2002; Guimbretiere et al. 2005; Cance et al. 2006].
3) *Ecological vs. external validity*? Field studies provide more ecological validity but less control. A related aspect is the choice of the frequency of commands. Zipfian distribution[1] of item frequency has been shown to be realistic [Findlater and McGrenere 2004] but can introduce an interaction effect with the item position of gesture assignation.

In this section, we discussed different aspects of menu performance. The objective was twofold.: First, it motivated the need to characterize the design space of menus in order to elaborate advanced predictive models of menu performance. Second, it serves for the better understanding of the role of the menu properties presented in the following section.

4. MENU PROPERTIES

Menu techniques are complex interaction techniques that can be decomposed in basic primitives that we call *menu properties*. A property improves menu techniques according to one or more criteria (presented in Section 3.2). In the following Sections, we present a taxonomy of menu properties. This taxonomy is intended to better understand similarities and differences between menus. Because of the complexity of menu techniques, this taxonomy is not meant to be a complete review of all existing properties but aims at providing a comprehensive synthesis of research studies on visual menu techniques.

4.1. Structure of the Taxonomy

The taxonomy (Table II) organizes properties according to the following three dimensions: Item (Table IV), Menu (Table V), and Menu System (Table VI) for both the novice and the expert mode. The dimensions follow a hierarchical organization: *items* are part of *menus*, which are part of a *menu system*. This hierarchical structure makes it possible to study visual menus at different levels of granularity. For instance, a designer may wish to improve the efficiency of an existing menu without changing the global organization of the *Menu system* or the layout of the *Menu* because users are already familiar with it. In contrast, designers creating a new application from the beginning may want to optimize the entire hierarchy. The *Item* and the *Menu system* sections propose solutions for these two scenarios. In *Expert mode,* we only retain items because this mode provides a direct access to items (Section 2.3). We thus present properties favoring the transition from novice (visual menus) to expert usage (hotkeys or gestures). The analysis of these properties highlights the available alternatives for designers to improve the performance of menus as well as emphasize less explored dimensions such as semantics.

4.2. Item

Items are the smallest components leading to the execution of a command. They have geometrical, graphical and semantic attributes that we now describe.

Table II. The Taxonomy Organizes Menu Properties According to Three Dimensions: Item, Menu, and Menu System

It distinguishes Novice and Expert mode. As the Expert mode provides a direct access to commands, it only involves the item dimension. Each dimension is divided into subdimensions.

	Dimension	**Subdimension**
Novice Mode (first modality)	Item	Geometry
		Visual representation
		Semantics
	Menu	Geometry
		Temporality
		Semantics
	Menu System	Semantics
		Menu depth
		Menu breadth
Expert mode (second modality)	Item	Modality
		Semantics
		Geometry
		Visual representation
		Temporality

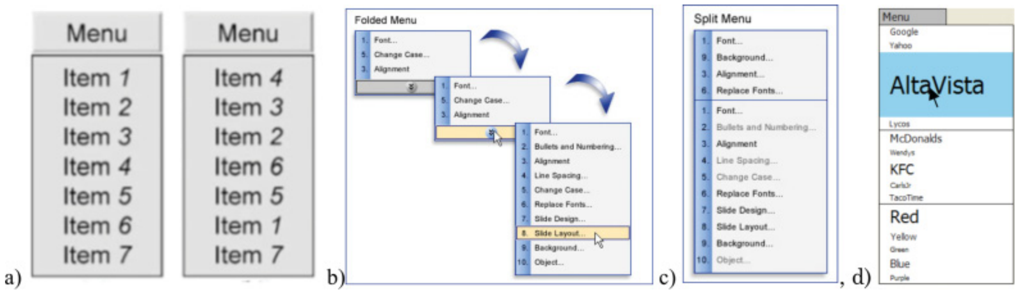

Fig. 10. (a) **Frequency-ordered menus**: the more frequently an item is used, the higher its position in the menu. (b) **Folded menus** initially present frequent commands, then present the entire list of commands after a time delay or a click on the folded button. (c) **Split menus** [Sears and Shneiderman 1994] contain two parts: a top area containing the most frequent items and a bottom area containing all menu items. (d) **Morphing menus** [Cockburn et al. 2007] dynamically increase the size of frequent items.

4.2.1. Item Geometry.

Reducing the distance. According to Fitts' law [Fitts 1954], reducing the distance of an item from the top of the menu improves motor control performance. Moreover this also reduces the time needed for localizing the item, especially for novice users who tend to perform a "serial inspection of items"[6] [Norman 91] from the top of the menu. **Frequency-ordered menus** [Lee and Yoon 2004] and **Folded menus** (Figure 10(a,b)) move the position of the most frequent items automatically to the top of the menu to reduce this distance. Unfortunately, this does not maintain *spatial consistency*: the locations of the items can vary over time, which makes it difficult for users to predict their location. **Split menus** [Sears and Shneiderman 1994] avoid this problem by duplicating frequent items (Figure 10(c)): their top area only contains the most frequent

[6]Novice users often perform a *serial inspection* of items [Norman 1991]. They read items one by one from the top of the list until they reach the desired item. They can also perform *Random inspection and either keep* keeping track of items already inspected (*without replacement*) or not (*without replacement*).

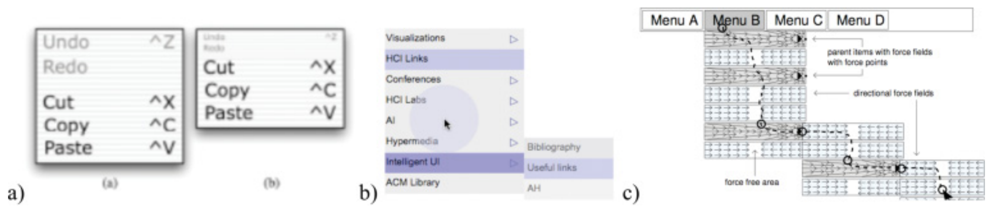

Fig. 11. (a) **Semantic menus** [Blanch et al. 2004] display all items with the same size in the visual space (screen). In the motor space related to mouse movements, disabled items are smaller to make their selection more difficult. (b) **Bubbling menus** [Tsandillas and Schraefel 2007] dynamically adapt the cursor size to overlap the closest favorite item (with a blue background) when users perform a drag action. Users can thus select this favorite item even if the default cursor is not over the item. (c) **Force Field menus** [Ahlström 2005] attract the cursor in the center of nonhierarchical items to improve accuracy and toward the submenu when the cursor is over a parent item to make the steering task easier to perform.

items while their bottom area contains all items. This design comes from observations of menu usage showing that the two or three most frequent items are selected 70% to 90% of the time. It is used in commercial software, such as in the "Font" menu of Microsoft Office. Lee and Yoon [2004] compared Folder menus as well as Split menus with traditional linear menus. They showed that Folded menus and Split menus are more efficient than Linear menus *only* if frequent items are selected more than, respectively, 90% and 30% of the time. Below 30%, there is no benefit gained by adopting a different technique than traditional linear menus.

Increasing the size. As stated by Fitts' law, increasing the size improves motor control performance. A larger size also improves item localization because this increases saliency as larger items are likely to attract gaze. **Morphing menus** [Cockburn et al. 2007] build on this idea by changing the size of the items dynamically depending on their frequency of use (Figure 10(d)). While changing the geometry of items, this solution preserves the relative ordering of the items.

Visual vs. motor space geometry. Semantic pointing[7] and related techniques [Blanch et al. 2004; Ahlström 2005; Tsandilas and Schrafel 2007] improve performance by dealing differently in the motor space (mouse movements) and the visual space (graphical representation on the screen). **Semantic menus** [Blanch et al. 2004] and **Bubbling menus** [Tsandillas et al. 2007] (Figure 11(a,b)) make items easier or more difficult to select without changing their graphical representation as geometry only changes in the motor space. In the same spirit, **Force Fields menus** [Ahlström 2005] improve accuracy by using force fields that attract the cursor in the center of nonhierarchical items (i.e., items that do not open submenus (Figure 11(c)).

Another solution consists of making pointing gestures *scale-independent*: the selection does not depend on the length of the gesture, which means that the target width can be seen as infinite. This strategy has been successfully used in **Marking menus** [Kurtenbach et al. 1991]. "**Impenetrable borders**" [Walker and Smelcer 1990] and the **MacIntosh menu bar** follow a similar idea for linear menus. They either prevent the cursor from overshooting the last menu item (first case) or the menu bar (second case), which lies at the top border of the screen.

4.2.2. Visual Representation.

Conveying information. A major goal of a menu item is to convey information about the corresponding command, using a textual and/or a visual representation. Menu

[7]Semantic pointing [Blanch et al. 2004] dynamically changes the control/display ratio of the mouse depending of the location of the cursor on the screen: The same movement in the motor space (mouse movement) will move the cursor in a different way depending on the objects under the cursor.

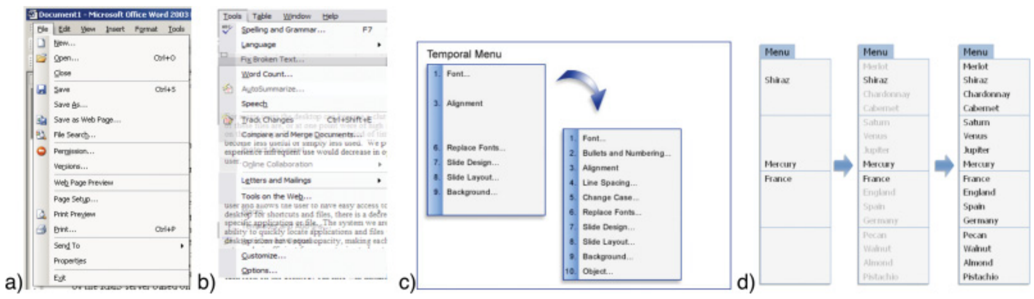

Fig. 12. (a) Linear menus with colored icons. (b) **Transparent menus** [Bowes et al. 2003] reduce the visibility of infrequent commands by increasing transparency. (c) **Temporal menus** [Lee and Yoon 2004] display items in two stages: first, only frequent items are displayed, then all items are displayed after a delay of 170ms. (d) **Ephemeral menus** [Findlater et al. 2009] work in a similar way but infrequent items gradually fade in.

items generally include a textual representation on desktop workstations. The choice of the wording is related to semantics and discussed in Section 4.2.3. In this section, we focus on the visual representation of the items. For instance, menu items often contain icons on Microsoft Windows or mobile operating systems. An advantage of using icons in menu items is that the same icon can also appear in the toolbar, which highlights the link between these interactors. While icons require less real estate than text, they may, however, be more difficult to interpret.

Increasing saliency. We already presented **Morphing menus** [Cockburn et al. 2007] (Figure 10(d)) that improve directed search by increasing the size of the items. Some other techniques increase saliency without changing item geometry. **Bubbling menus** [Tsandillas and Schraefel 2007] (Figure 11(b)) and **Transparent menus** [Bowes et al. 2003] (Figure 12(b)), respectively rely on transparency and background colors to impact item saliency. Temporal priority [Lee and Yoon 2004] is another option: **Temporal menus** [Lee and Yoon 2004] and **Ephemeral menus** [Findlater et al. 2009] (Figure 12(c,d)) first display frequent items, then make the other items visible after a delay. **Icons** also increase saliency (Figure 12(a)), with the advantage of providing additional information [Helander et al. 1997]. Graphical elements such as "**separators**" can also be used to attract attention by grouping items visually.

Increasing visual context. Because users need to remain concentrated on their main task, menus must not obliterate their visual focus. This is why menus are "transient" [Jackobsen and Hornarek 2007] and displayed "on demand" [Hinckley and Sinclair 1999], contrary to tool palettes and similar interactors. As mentioned earlier, transparency can serve to decrease the saliency of certain items but also to help sharing attention between the menus and the main task of the user [Bowes et al. 2003; Tapia and Kurtenbach 1995; Harrison and Vicente 1996; Hinckley and Sinclair 1999; Rubio and Janecek 2002]. However, transparency can make it difficult to read items. The "**Anti-Interference**" font [Harrison and Vicente 1996] allows increasing transparency without decreasing text legibility. Alternatively, certain items can be masked when navigating in the hierarchy of commands [Tapia and Kurtenbach 1995], for instance by displaying only the current menu instead of displaying all parent menus [Zhao and Balakrishnan 2004].

4.2.3. Semantics. Surprisingly, most studies on menu techniques focus on interaction but rarely on semantics. Because the names of the categories and the commands they contains are strongly related, the way they are chosen can strongly impact performance

Table III. Summary of the Main Properties for the Dimension Item

Dimension	Subdimension	Properties	Menus
Item	Geometry	Reducing distance	Frequency-ordered menus; Folded menus; Split menus
		Increasing size	Morphing menus; Marking menus; Impenetrable borders; Semantic menus; Bubbling menus;
	Visual representation	Conveying information	No specific menu
		Increasing saliency	Morphing menus; Bubbling menus; Transparent menus; Temporal menus; Ephemeral menus;
		Increasing visual context	Transparent menus; Anti-Interference font.
	Semantics	Choosing relevant name	No specific menu
		Reducing the length	No specific menu

[Helander et al. 1997]. The names (and the length of the names) of the items should thus be chosen with care by designers.

Choosing relevant names. Choosing command names is challenging because users often search for functionalities but do not know the exact command names (Section 2). Command names should thus be comprehensible and reflect the purpose of the corresponding functionalities to facilitate the match between the targeted functionality and the item.

Reducing items length. Using long names conveys more information but long names are slower to read and require more screen real estate.

4.3. Menus

Menus organize items in a coherent way. The design of menus raises three main questions: how to organize items, where to display menus and when to display them? The two first questions are related to the *geometry* of the menu while the third one is related to its *temporality* (Table III).

4.3.1. Geometry I: Menu Layout.

Linear Layout. Most menus are laid out linearly. Items are organized vertically, except in menu bars where they are horizontally laid out. The linear layout is space-efficient because the geometry of items fits well the geometry of text labels (especially for Western languages) and the geometry of rectangular screens. Moreover, it avoids unnecessary eye movements during serial inspection as each item is close to the next one [Ahström et al. 2010; Samp and Decker 2010].

Grid Layout. A *grid layout* can be used to reduce the mean distance between items hence pointing time as in **Square menus** [Ahlstrom et al. 2010] (Figure 13(a)) or **FastTap** [Gutwin et al. 2014]. Using a grid layout does not dramatically alter the rectilinear layout of Linear menus [Ahlstrom et al. 2010] but provides more flexibility for highlighting semantic relationships between items (for instance, related items "Save" and "Save As" can be located on the same row).

Radial Layout. A *radial layout* (**Pie menus**[8] [Callahan et al. 1988]; Figure 13(b)) places items in a circular design at an equal radial distance from the center. This layout ensures constant access time and improves global performance: Callahan et al. [1988] showed that radial menus were 15% faster than linear menus for eight items. A

[8]Preliminary versions of the Pie menus appeared in 1969 [Wiseman et al. 69; Forsey 84].

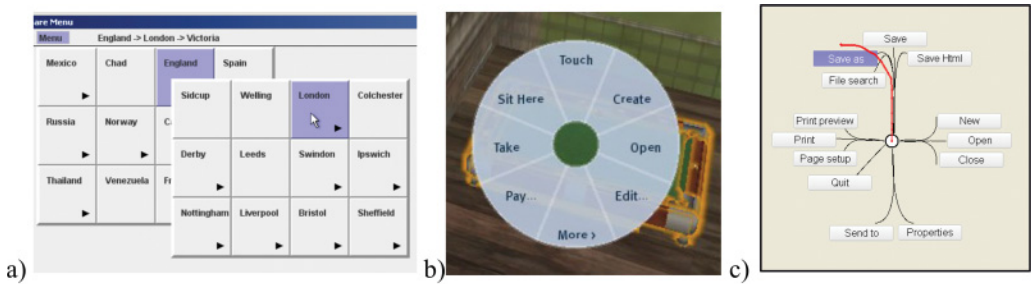

Fig. 13. (a) **Square menus** [Ahlstrom et al. 2010] organize items in a matrix. (b) **Pie menus** [Callahan et al. 1988] place items in a circular design at an equal radial distance from the center (radial layout). (c) **Flower menus** [Bailly et al. 2008] are a variant of Marking menus, which rely on curved gestures to increase the number of commands.

radial layout exploits spatial memory by associating commands with cardinal orientations. Orientations can serve to strengthen semantic relations [Soliz and Paley 2003]. For instance, opposite commands, “Open” and “Close” can be placed in symmetrical locations. Thanks to procedural memory, such relationships help the learning and the memorization of commands.

Other layouts. Variants of radial layouts have been proposed such as the *polygon layout* [Zhao et al. 2006], *half pie layout* [Hesselman et al. 2009], *concentric layout* [Samp and Decker 2010], *interverted concentric layout* [Bailly et al. 2007; Francone et al. 2010], and the *flower layout* [Bailly et al. 2008]. Instead of only relying on orientation, they also use additional dimensions such as relative position, depth, curvature, and so on (see Section 4.4.3). For instance, in Flower menus (Figure 13(c)), the orientation is mapped to a semantic group of commands, which are differentiated through curvature. More generally, combining several dimensions favor visual perception by “making relatively crude judgments of several things simultaneously” [Miller 1956]. This also increases design possibilities for organizing commands in a meaningful way by visually highlighting associations between related elements.

4.3.2. Geometry II: Menu Positioning. The relative position of the menu from the activation point (i.e., the cursor position before opening the menu) impacts performance. Context menus are displayed on the southeast side of the activation point, which makes them quick to access. However, in hierarchical menus, users must perform a horizontal movement to reach a cascaded menu. These menus appear on the right side of their parent item, hence introducing a steering cost [Accot and Zhai 1997]. Two opposite strategies have been proposed to reduce this cost. They either consist in moving the menu close to the cursor position (**Motion menus** [Kobayashi and Igarashi 2003]; Figure 14(a)) or moving the cursor close to the menu (**Jumping menus** [Ahlström et al. 2006]; Figure 14(c)). Both solutions involve potential drawbacks. In the first case, the submenu overlaps the parent menu, which may alter navigation. In the second case, users may be disoriented by loosing the complete control of the cursor, which “jumps” from the parent item to the submenu [Tanvir et al. 2008].

A radial layout ensures constant access time for all items because they are centered around the cursor. This contrasts with linear menus because items on the top of the menu are usually faster to select. An alternative places the menu at mid height relatively to the activation point to decrease the average distance for reaching items (**Walker menus** [Walker and Smelcer 1990] and **AAMUs** [Tanvir et al. 2008]; Figure 14(b)).

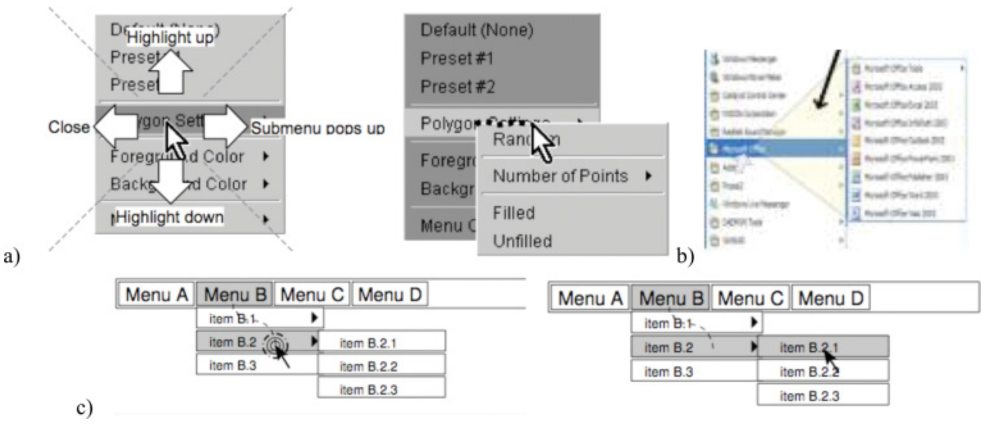

Fig. 14. (a) **Motion menus** [Kobayashi and Igarashi 2003]: an horizontal cursor movement opens/closes submenus at the position where the horizontal movement occurs. Vertical movements are only used to highlight the item under the cursor. (b) **AAMUs** [Tanvir et al. 2008] display submenus at mid-height and provides an enlarged activation area to make it easier to reach submenus. This area, which is triangular-shaped, partially overlaps the parent menu. While the cursor is over this area, the submenu remains open. (c) **Jumping menus** [Ahlström et al. 2006] warp the cursor to the submenu when the user clicks on its parent item so that the cursor "jumps" from the item to its submenu.

Opening a menu close to a screen border can be problematic because part of the menu may lie outside of the screen. Linear menus solve this problem by shifting the menu on the left side of the activation point when there is not enough space remaining on its right side (similarly, the menu may appear on the top instead of the bottom of the activation point). Circular menus can also be shifted, but this can break the correspondence between the orientations of the items and the gestures the user must produce for reaching them. This problem is especially critical for **hierarchical Marking menus** [Kurtenbach et al. 1993] because their submenus are displayed in the direction of their parent item. Hence, a two-level Marking menu requires three times the amount of space it occupies in the vertical and horizontal directions for displaying its submenus. This makes it difficult to use them on small screens without altering the interaction. **Multi-Stroke menus** [Zhao et al. 2004] offer a solution to this problem. They superimpose submenus instead of placing them on the sides of the parent menu.

4.3.3. Temporality of Menus. This section addresses the question of when menus should be opened and closed. Menus are usually opened on demand and closed as soon as a command is activated to not obliterate the screen. However, as explained in the following text, some subtle differences may impact performance.

Browsing Menus. Menu preview [Rekimoto et al. 2003; Bailly et al. 2007] consists of automatically opening a submenu when the cursor lies over its parent item for a short amount of time. This mechanism facilitates visual search in linear menus because users can quickly explore a set of submenus, without clicking, simply by moving the mouse over the menu items. Surprisingly, this mechanism has not received much attention in the menu literature and it is rarely present in circular menus. Only **Wave menus** [Bailly et al. 2007] and **Wavelet menus** [Francone et al. 2010] (Figure 15(a)) offer the capability to preview circular submenus.

Selecting items in submenus. As mentioned earlier, accessing submenus (hence submenu items) involve a steering cost in linear menu systems. Current menu implementations attempt to solve this problem through a temporal delay that prevents a menu to be closed when the user quickly moves the cursor to an opened submenu. This allows

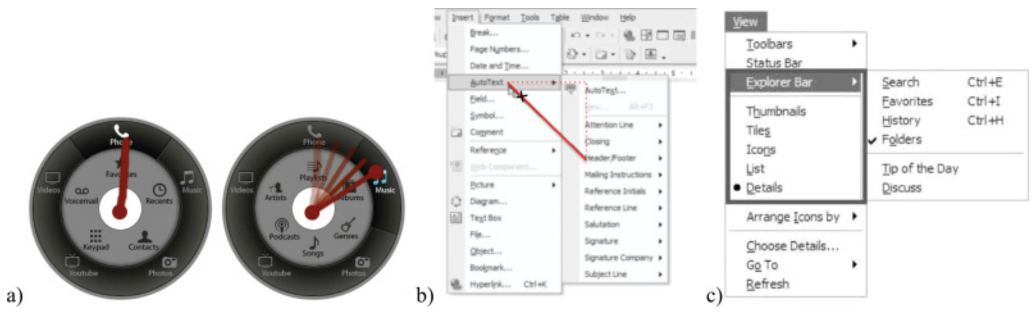

Fig. 15. (a) **Wavelet menus** [Francone et al. 2009] provide an inverted concentric layout: Submenus are in the center, while parent menus are on the outmost rings. Users perform a circular gesture illustrated in red (dark grey in black and white) with the thumb to preview the different submenus. (b) Traditional menus using a delay to allow users to perform a diagonal path to access submenu. (c) **EMUs** [Cockburn and Gin 2006] keep the submenu opened while the cursor does not enter a new hierarchical item.

accessing submenu items by performing an optimal diagonal movement (Figure 15(b)). This delay, which is platform and application dependent, may be too short for novice users and too long for expert users according to [Cockburn and Gin 2006]. In order to avoid using delays, in **EMUs** (Figure 15(c)), the last opened submenu remains opened when the cursor moves to another item, provided that this new item does not open another submenu [Cockburn and Gin 2006]. The "submenu activation area" hence depends on the density of hierarchical items in the current menu. Another alternate approach (**AAMUs** [Tanvir et al. 2008], Figure 14(b)) consists in providing an enlarged activation area making it easier to reach submenus. This area is triangular-shaped and partially overlaps the parent menu. The submenu is closed when the cursor leaves this area.

4.3.4. Semantics. When considering the semantics of a menu (in addition to the semantics of individual items; Section 4.2.3), designers should favor the consistency of item logic and select a relevant menu title.

Making wording consistent. Designers should ensure that the wording is consistent within a menu to guide users. For instance, all items should either be verbs or nouns.

Choosing a relevant menu title. Categories should guide users and encourage learning [Norman 1991; Bastien and Scapin 1993; Lee and Raymond 1993]: The title should reflect the commands in the corresponding submenu while not overlapping with other submenus.

4.4. Menu System

In this section, we now consider the impact of the breadth and depth of a menu system from a practical point of view as some techniques may be inherently more or less suited for a given application or context (*applicability* factor of Table I).

4.4.1. Semantics. The semantics of a menu system strongly impacts exploration efficiency [Helander et al. 1997]. However, it is difficult to organize commands in a manner that reflects the users' perception [Tullis 1985]. To define efficient menu structure, menu designers should identify logical relationships between commands and organize them into a meaningful hierarchy.

Identifying logical relationships. One user-elicitation approach consists of asking some users to make semantic similarity comparisons among *all* the desired commands. However, this is time consuming as the number of comparisons exponentially increases

Table IV. Summary of the Main Properties for the Dimension Menu

Dimension	Subdimension	Properties	Menus
Menu	Geometry	Layout	Linear menus; Square menus; Pie menus; Marking menus; Polygon menus; Flower menus Half-Pie menus; Wave menus; Wavelet menus
		Positioning	Motion menus; Jumping menus; Pie menus; Walker menus; AAMUs
	Temporality	Browsing menus	Wave menus; Wavelet menus
		Selecting items in submenus	EMUs, AAMUs
	Semantics	Making wording consistent	No specific menu
		Choosing relevant menu title	No specific menu

with the number of commands. Another approach consists of asking some users to sort the commands into mutually exclusive groups based on their similarity [Tullis 1985]. Hierarchical clustering methods can then provide the logical relationships between commands that users perceive. However, these relationships are still not sufficient to design the menu hierarchy (e.g., how many menu levels should be used).

Building the hierarchy of commands. A menu system is characterized by its *breadth* (the number of commands in the largest menu in the menu system) and its *depth* (the minimum number of menu levels to traverse ensuring that all items can be accessed). As these two characteristics impact learning and selection performance, many studies considered the advantages of broad and deep structures and how they should be balanced to obtain an efficient trade-off [Snowberry et al. 1983; Kiger 1984; Landauer and Nachbar 1985; Snyder et al. 1985; Norman 1991; Jacko and Slavendy 1996; Larson and Czerwinski 1998; Zaphiris 2002; Zhao et al. 2006]. While there is no consensus on whether menu structure should be deep or broad, several studies conclude that minimizing the depth increases performance (see Zhao et al. [2006] or Cockburn and Gutwin [2009] for a detailed analysis of this trade-off). The main reason is that users may have difficulties to guess what low-level commands fall under each of the subcategories of a high-level menu.

We focused so far on cognitive considerations, that is, how the menu structure should reflect user's perception. However, some menu techniques are not well suited for a given menu structure because they impose specific constraints on the depth or breadth of the menu hierarchy.

4.4.2. Menu Depth.

Increasing the menu depth. While, as explained earlier, the menu depth should be kept small whenever possible, some menu systems require three or more menu levels. This can be problematic for hierarchical Marking menus because their accuracy may be insufficient for more than two levels [Kurtenbach et al. 1993]. Moreover, certain items of these menus must remain empty to guarantee scale independence,[9] with the consequence that more menu levels must be used [Zhao et al. 2004]. **Multi-Stroke menus** [Zhao et al. 2004] solve this problem by relying on temporal rather than spatial composition of marks, meaning that the user performs a series of simple marks instead of drawing one complex mark. This strategy not only increases precision, but it avoids ambiguous zigzag marks.[9]

[9]In hierarchical Marking menus, some zigzag marks are ambiguous because of scale independence. For instance, in a three-level Marking menu, the system cannot distinguish between [South; South; East] and [South; East; East]. Indeed, these two marks will be decomposed in the same way into two components (delimited by a unique inflexion), independently of their lengths.

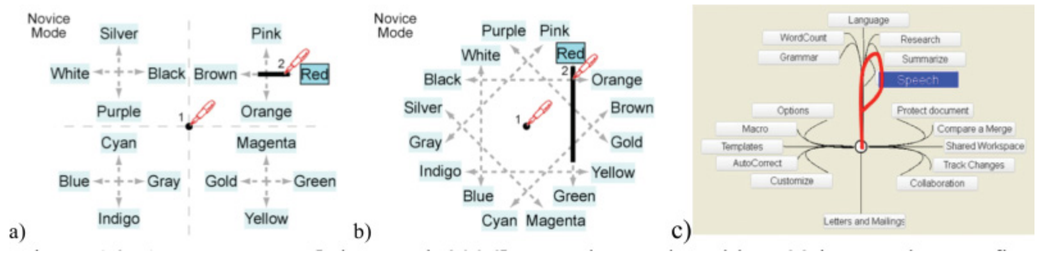

Fig. 16. (**a) Zone menus** [Zhao et al. 2006], extend menu breadth to 32 items. The user first taps to specify the menu origin. This action virtually splits the screen into four spatial areas corresponding to a different breadth-4 marking menu that the user activates in the usual way. (b) **Polygon menus** work in a similar way except that the items are the vertices of a N-sided polygon to contain up to 16 items. (c) **Flower menus** [Bailly et al. 2008] extend menu breadth to 56 items by combining orientation and four levels of curvature.

Logical groups (also called Within groups) provides another way to structure menu items. Contrary to hierarchical menus, they operate at a given menu level and do not increase the depth of the menu system [Lee and Raymond 1993; Bailly et al. 2008]. Logical groups are common in linear menus where they are delimited by a *separator* that appears as a thin horizontal line. Marking menus do not provide this feature—maybe because of their limited number of items—except in **Flower menus** [Bailly et al. 2008] where related items are grouped into branches (Figure 16(c)). Logical groups make visual search more efficient because the users can know whether they are searching in the right group simply by scanning few items of it.

4.4.3. Menu Breadth.

Increasing the breadth of circular menus. Marking menus can hardly contain more than eight items in expert mode (12 at the price of degraded performance) [Kurtenbach 1993]. However, applications often have one or several menus containing 12 or more items [Zhao et al. 2006; Bailly et al. 2008]. This may impose awkward semantic groupings of the menu items, hence leading to a hierarchy that does not reflect the user's perception [Zhao et al. 2006]. One solution is to combine **Marking menus with linear parts** [Kurtenbach 1997] (Figure 19(a)), but not all items can then be selected in expert mode. Another strategy is to combine orientation with other input dimensions [Nancel and Beaudouin-Lafon 2008]. The additional dimensions considered so far are:

- *Relative position*. **Zone menus** and **Polygon menus** use the relative position of strokes from the activation point of the menu to increase the number of commands. Zone menus can be seen as a combination of four Marking menus, as explained in Figure 16(a). Polygon menus involve a more complex structure where marks are not radial (Figure 16(b)) which may negatively impact the learning performance [Bailly et al. 2008].
- *Curvature*. Curvature has been used in addition to orientation in **Hybrid menus** [Isokoski et al. 2002] and **Flower menus** [Bailly et al. 2008] (Figure 16(c)). By introducing four levels of curvature, Flower menus can provide a theoretical maximum of 56 items.
- *Distance*. **DartBoard menus** [Kurtenbach et al. 1993; Nancel and Beaudouin-Lafon 2008] combine orientation and distance (Figure 17(a)). By considering four different distances, they can contain up to 32 items.
- *Pressure*. Pressure has been used in **Donut menus** [Lai et al. 2005; Ren et al. 2008] and **Push menus** [Huot et al. 2008] (Figure 17(b)). By exploiting three levels of pressure, users can activate up to 24 items.

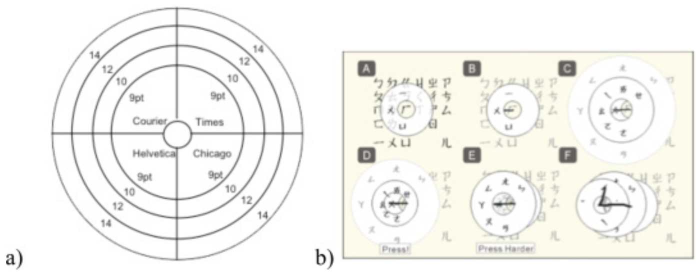

Fig. 17. **DartBoard menus** in (a) [Kurtenbach 1993], increase menu breadth by combining orientation and distance. (b) **Donut menus** [Lai et al. 2005; Ren et al. 2008] combine orientation and two or three levels of pressure.

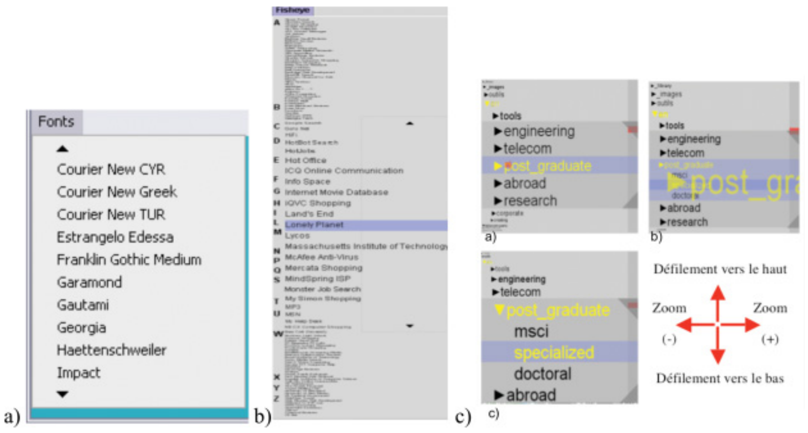

Fig. 18. (a) **Scrollable menus**. Two scroll buttons enable the access to previous or next items. (b) **Fisheye menus** [Bederson 2000]. The size of items dynamically changes relatively to their degree of interest (DOI): items located in the focus area (with a high DOI) are displayed at a readable size, while the size of the items lying in the context area is inversely proportional to their distance from the focus zone. (c) **FishTree** [Lecolinet and Nguyen 2006] extends the Fisheye menus for hierarchical menus.

Increasing the breadth of Linear menus. While linear menus do not suffer from the same limitations than circular menus, they may not fit on the screen if they contain too many items (e.g., more than 30 items on a 1,024×768 screen). A scrollbar can then be used (Figure 18(a)), but this tends to make the interaction slower and cumbersome [Bederson 2000]. Fisheye views [Furnas 1986] may then be an effective alternative such as in the **Fisheye menus** [Bederson 2000] and **Fish Tree** [Lecolinet and Nguyen 2006] (Figure 18(b,c)) techniques. Additionally, several menu bars can be used as in the Hotbox [Kurtenbach et al. 1999] (Figure 19(b)).

4.5. Expert Mode

The expert mode lets users activate commands without displaying the menu. The visual focus of users is not obliterated so that the users can remain concentrated on their main task. However, the expert mode normally requires more cognitive efforts at the beginning as it is based on recall rather than recognition (Section 2.2.1). For the expert mode, designers should consider:

- Which modality to use (hotkeys/gestures).
- How to assign commands to hotkeys or gestures (mapping/semantic).

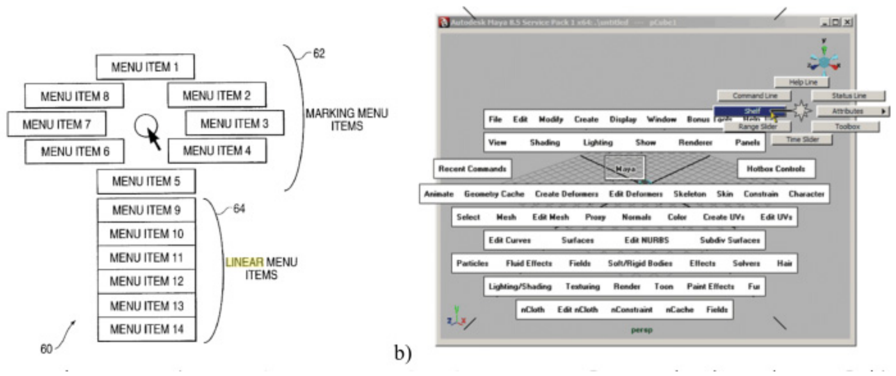
MENU ITEM 1
MENU ITEM 2
MENU ITEM 3
MENU ITEM 4
MENU ITEM 5
MENU ITEM 6
MENU ITEM 7
MENU ITEM 8
62
MARKING MENU ITEMS
MENU ITEM 9
MENU ITEM 10
MENU ITEM 11
MENU ITEM 12
MENU ITEM 13
MENU ITEM 14
64
LINEAR MENU ITEMS
60
a)
File Edit Modify Create Display Window Bonus
View Shading Lighting Show Renderer Panels
Recent Commands
Maya
Hotbox Controls
Animate Geometry Cache Create Deformers Edit Deformers Skeleton Skin Constrain Character
Select Mesh Edit Mesh Proxy Normals Color Create UVs Edit UVs
Edit Curves Surfaces Edit NURBS Subdiv Surfaces
Particles Fluid Effects Fields Soft/Rigid Bodies Effects Solvers Hair
Lighting/Shading Texturing Render Toon Paint Effects Fur
nCloth Edit nCloth nConstraint nCache Fields
Help Line
Command Line
Status Line
Shelf
Attributes
Range Slider
Toolbox
Time Slider
persp
b)

Fig. 19. (a) **Marking menus with linear parts** [Kurtenbach et al. 1997], (b) **Hotbox** [Kurtenbach et al. 1999]. Several menu bars are displayed in the central zone. Each of the four zones around invokes a different marking menu, which can be accessed by pressing the mouse. The Hotbox, which can contain up to 1,200 commands, is used in Maya products.

Table V. Summary of the Properties of the Menu System Dimension

Dimension	Subdimension	Properties	Menus
Menu System	Semantics	Identifying semantic relationships between commands	No specific menus
		Building a hierarchy	No specific menus
	Menu Depth	Increasing depth	Multi-Stroke menus; Polygon menus
		Logical grouping	Flower menus
	Menu Breadth	Increasing breadth	Zone menus; Polygon menus; Flower menus; Hybrid menus; Donut menus; Push menus; DartBboard menus; Donut menus; Push menus; Scrollable menus; Fisheye menus; FishTree menus

- How to make the mapping observable (geometry, visual cues, and temporality of the cues).

4.5.1. Modality. We distinguish keyboard shortcuts (hotkeys) and stroke shortcuts (gestures). Stroke shortcuts have several advantages in comparison with hotkeys. First, they often offer good memorization performance because gestures are spatial and iconic [Appert and Zhai 2009]. Then, they rely on the same input device (e.g., the mouse) than the novice mode. For instance **Marking menus** [Kurtenbach and Buxton 1991] are not affected as much as hotkeys when users switch from the novice to the expert mode (Figure 9) because users do not change of input device. This allows users to physically rehearse the way an expert would issue a command (rehearsal principle [Kurtenbach 1993]). Indeed, users perform similar gestures in novice and expert mode, only the visual appearance of the menu changes. So, they learn the expert mode implicitly, simply by using the menu repeatedly in novice mode. When the gesture is performed fast enough, the technique is automatically in expert mode. This is possible because gestures are encoded into muscle memory (also called motor learning), which is a form of procedural memory that involves consolidating a specific motor task into memory through repetition.

4.5.2. Semantics: Mapping between Commands and Shortcuts. Designing an efficient mapping between commands and their shortcuts can strongly impact performance. Surprisingly, few studies focused on this aspect. In the case of keyboard shortcuts, a

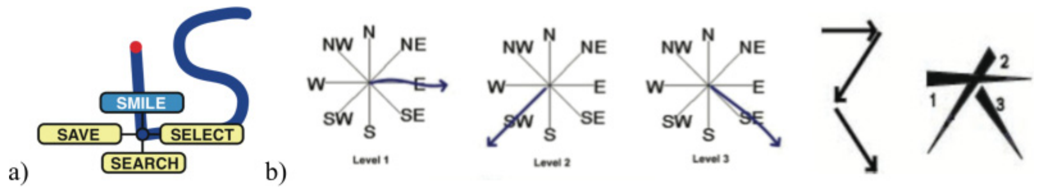

Fig. 20. (a) **Augmented Letters** [Roy et al. 2013]: Users draw a unistroke “S” related to the first letter of the command “Smile”). The unistroke is augmented with a tail to avoid collision. (b) The series of strokes are composed to form a Chinese glyph [Zhao and Chignell 2007].

straightforward strategy uses the **first letter of the command** as a hotkey, such as Ctrl+C for “Copy” or Ctrl+S for “Save.” However, this is only suitable for a small number of commands because of the name collision problem (or, more precisely, the collision of their first letters). Different modifiers (e.g., the Alt or Shift keys) or combination of modifiers can then be used, but this can lead to complex shortcuts that are cumbersome to perform and uneasy to memorize (e.g., Alt+Shift+Cmd+V for “Paste and apply style” on Safari on the Mac). Spatial proximity is sometimes used to solve this problem. An example is the “Copy,” “Cut,” “Paste” trio of related commands. The first letter is used as a hotkey for “Copy,” but not for “Cut” and “Paste” that rely on keys (X and V) that are adjacent to the C letter on a Qwerty keyboard. Metaphors are also used as an additional mean to memorize the mapping in this case because X can be seen as an iconic view of a pair of scissors. Using alphabetical proximity is another possibility, as for instance for the “Undo”/“Redo” (Ctrl-Z/Ctrl-Y) on Windows software. However, none of these solutions makes it possible to solve the name collision problem simply and efficiently for a large number of commands. Physically augmenting the keyboard can alleviate this limitation. For instance, the **Métamorphe keyboard** [Bailly et al. 2013] allows assigning several commands to the same key.

Similar to the traditional hotkey assignation, a gesture can be related to the first letter of the command [Li et al. 2012; Roy et al. 2013]. The unistroke letter can be augmented with a tail to avoid collision (similar to the use of additional modifiers for hotkeys) [Roy et al. 2013], as shown in Figure 20(a). In Zhao and Chignell [2007], strokes are composed to graphically represent the corresponding Chinese glyph of the command (Figure 20(b)). Gestures offer more flexibility than hotkeys because they provide more design possibilities for assigning commands. For instance, a gesture can be related to the meaning of the command and mimic the physical or conventional effect of the gesture in the real word (e.g., “Next” can be assigned to a left-to-right stroke).

We believe that optimizing command-hotkey and command-gesture mappings is a promising direction for future work. Few studies investigated this problem for menus although this aspect can strongly impact performance. User-defined gestures [Wobbrock et al. 2009] define an interesting approach that involves presenting “the effects of actions to participants and elicits the causes (*gestures*) meant to invoke them.” This approach is useful for defining an “intuitive” mapping between gestures and commands. However, this approach tends to be limited to small command sets, can hardly be used for abstract commands, and may make it difficult to maintain a logical organization. Finding a good balance between analogue and abstract mappings[10] is an open research question that not only concerns menus but also more generally gestural interaction.

[10]When the gesture is related to the meaning of the command (mimic the physical or conventional effects in the real word), the mapping is called analogue [Zhai et al. 2012]. In contrast, abstract mappings use arbitrary gestures that do not resemble physical effects. Analogue/abstract mapping is more a spectrum than a dichotomy [Zhai et al. 2012]. Analogue mappings are generally easier to learn but slower to execute.

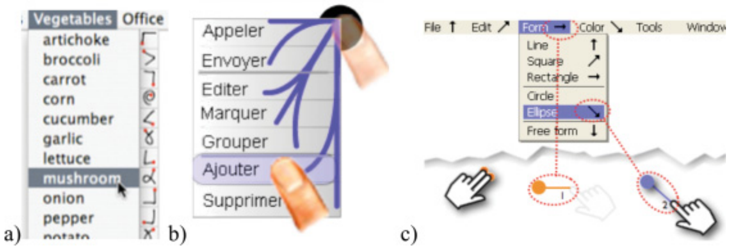

Fig. 21. (a) Command-gesture mapping displayed [Appert et al. 2009]. (b) **Leaf menus** [Roudaut et al. 2009]. (c) **Radial-Stroke shortcuts** [Bailly et al 2010].

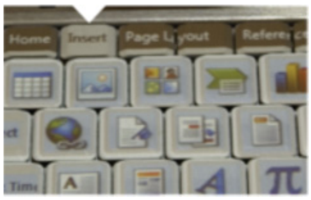

Fig. 22. Command-hotkey mapping displayed on the keyboard. The **TDK Keyboard** [Block et al. 2010]. Icons are dynamically displayed on the corresponding hotkey.

We presented different strategies for improving the mapping. However, defining efficient mappings is not sufficient. If users are not aware of the expert mode and not motivated for using it, they will not switch from the novice to the expert mode. In practice, Lane et al. [2005] observed that most users, and even highly experimented users, rarely employ the expert mode. This is related to the “paradox of the active user” [Carrol and Rosson 1987], which results from a trade-off between efficiency on long-term and productivity on short-term. Users tend to avoid disrupting their current task by learning a new shortcut. More precisely, this behavior is an example of a wider human phenomenon called *satisficing* [Simon 1987], where people are satisfied to use suboptimal strategies due to limited cognitive resources: Users have difficulties to estimate whether learning the expert mode will be a worthwhile investment. In this context, a first step is to make the expert mode observable by the users to encourage them to switch to the expert mode.

4.5.3. Geometry.

In menus. Command-hotkey mappings are generally located in menus, on the right side of item labels. This location has also been used for displaying stroke shortcuts [Appert and Zhai 2009; Kurtenbach 1993; Bailly et al. 2010; Roudaut et al. 2009], as shown on Figure 21. However, the results of Grossman et al. [2007] suggest that some users commonly ignore them. This may be due to the fact that the shortcut is not displayed until the user has already done most of the work for selecting the command with the mouse. And at this stage, the users have no incentive to learn the hotkeys [Malacria et al. 2013].

On the input device. Commands-hotkey mappings can also be displayed on the keyboard such as with the **LogicKeyboard** [2015]; **Optimus Keyboard** [Optimus 2016]; Microsoft Adaptive Keyboard [MAK 2010] or **TDK** [Block et al. 2010]) shown on Figure 22. However, displaying all shortcuts can be difficult due to the limited number of keys in comparison with the total number of commands.

Finally, some applications display a subset of the available shortcuts (hotkeys or gestures) as tooltips of toolbar buttons, or show all of them in a cheat sheet.

However, tooltips only appear after a delay and cheat sheets are generally not well integrated in the user task and may be somewhat difficult to access [Malacria et al. 2013].

While several studies have investigated the display location of the shortcuts, we are not aware of approaches investigating visual representations to highlight the shortcuts or command-shortcut mappings. We could imagine highlighting the shortcuts of frequent commands. For instance, the size of the keyboard shortcuts inside item could depend on how often they are used.

4.5.4. Temporality.

Feedforward. This paragraph addresses the question of when command-shortcut mappings must be displayed. Usually, this mapping appears once the menu is opened, for instance just before users select a command using the mouse. In contrast, **Expose-Hotkey** [Malacria et al. 2013] open all menus when users press a modifier key (e.g., Ctrl) to display frequent hotkey mappings. This solution provides feedforward information about hotkeys without requiring pointing in menus. A variant of this technique [Tak et al. 2013] displays these mappings in a window at the center of the screen when users press a modifier key, but it does not exploit users' previous knowledge on command locations. While these two techniques favor hotkey usage [Malacria et al. 2013; Tak et al. 2013], they assume that users are inclined to press a modifier to get informed about the expert mode.

Another strategy consists of modifying the behavior of the menu. For instance, **Marking menus** [Kurtenbach and Buxton 1991] use a delay before displaying the novice mode. This delay deteriorates the novice mode but increases the user's perception of future performance gain by using the expert mode. Increasing the delay can thus favor the transition from novice to expert mode. Grossman et al. [2007] exploited this idea for linear menus by using an infinite delay so that the novice mode is disabled. Commands are visible in the menu, but users cannot click on them. Instead, they must enter the corresponding keyboard shortcut. While this cost-based approach has been proved efficient, the risk is that users may feel frustrated and stop using the technique.

Feedback. An alternative is to use feedback (instead of feedforward): the mapping is presenting after (instead of before) the selection of a command. The feedback can be visual, as in Grossman et al. [2007], where the hotkey or the mapping remains displayed after the selection of the command. Visual effects and animations can serve to better attract the attention of users [Grossman et al. 2007]. In **Marking menus** an ideal mark is displayed right after the activation of a command to show the users which mark should be performed in expert mode [Tapia et al. 1995]. Audio feedback can also be used. For instance, a vocal synthesizer can enunciate the mapping when the user activates an item. While more efficient than visual feedback, audio feedback can be annoying or undesirable in certain situations (e.g., in public areas). Haptic feedback has also been investigated in the **Métamorphe** keyboard [Bailly et al. 2013]. Once a command is activated, the corresponding (hot)key raises under the finger of the user to attract her attention.

The aforementioned strategies increase the awareness of the technique and help users to learn the mapping. In contrast, some techniques aim at informing about potential performance gain (Section 3.1) with the expert mode [Malacria et al. 2013]. Indeed, the perception of performance is a critical factor, which influences the decision of users of whether to switch from novice to expert mode [Scaar et al. 2011]. For instance, in **Skillometers** [Malacria et al. 2013], a comparison of the needed time in novice and expert modes is displayed after each command selection.

Table VI. Summary of the Properties of the Dimension Item of the Expert Mode

Dimension	Subdimension	Properties	Menus/Studies
Item	Modality	Hotkey	Linear menus
		Gesture	Marking menus
	Semantics	Efficient command-shortcut mappings	First letter; Métamorphe keyboard; Augmented Letters; [Zhai and Chignell 2007].
	Geometry	Display in menus	[Kurtenbach 93]; [Appert et al. 09]; Leaf menu; Radial-Stroke shortcuts
		Display on the keyboard	logicKeyboard; Optimus Keyboard; TDK
	Visual representation	Highlihghting the command-shortcut mapping	No specific menu or study
	Temporality	Feedforward	ExposeHotkey; [Tak et al. 2013]
		Delaying the menu itself	Marking menus; [Grosman et al. 2007]
		Feedback	[Grosman et al. 2007]; Marking menus; [Tapia et al. 1995]; [Bailly et al. 2013].

5. INTERACTIVE WEB SITE

We built an interactive Web site (www.gillesbailly.fr/menua/) illustrating more than 60 menus. This Web site provides abstracts, figures illustrating the menu techniques and information about their authors and the related publications. It also includes tools for searching and filtering amongst the set of described menu techniques. A timeline view highlights the growing number of menu techniques proposed during the last few years. This tool can help designers to quickly get an overview of existing menus.

6. CONCLUSION AND CHALLENGES

While extensive research has been conducted on menus from more than 40 years, we are far to fully understand the design of menus. At first glance, a menu can be seen as simple graphical widget that solves a simple problem: the selection of an element among N offered by a computer. In fact, a menu is a complex interaction technique that relies on various different properties. In this article, we formulated menu design as an optimization problem and we addressed a first challenge, by attempting to characterize the design space of menus. To achieve this, we first presented a definition of menus and discussed menu usage to better understand what a menu is and the related implications. We then focused on menu performance through a list of quality criteria and by reviewing existing analytical and empirical methods for quality evaluation. We then proposed a taxonomy of menu properties to structure existing work and highlight their impact on performance. This analysis should help designing novel menu techniques and informing application designers about possible design choices.

While this work contributes to the advance of menu design, several challenges remain unanswered:

Identifying. While our taxonomy aims at identifying and describing menu properties, it does not cover all of them. Identifying all possible menu properties is a difficult if not impossible task as new interaction techniques, new devices and new modalities will undoubtedly be introduced in the future. Covering this large and continuously evolving design space is by itself a challenge.

Understanding. Another challenge is to precisely evaluate the impact of all these properties on usability and applicability. Menu techniques are generally evaluated in one specific context, for a specific task, with specific users. More user studies are needed to understand the role of each property on interaction quality. For instance, there is relatively little knowledge about the role of semantics on menu interaction. Still, it is not clear how menu item wording, or the way hotkeys are chosen, impact on performance, navigation, learnability, and the like.

Predicting. Predictive models of menu performance are useful to encapsulate scientific knowledge. While many menu techniques have been designed, few predictive models of menu performance have been proposed. Moreover, they only cover a subset of the existing menu properties. From our point of view, establishing efficient models of performance is the next important challenge to address. However, building a predictive model is a complex work, which first necessitates a deep understanding of menu properties and their respective impact [Bailly and Oulasvirta 2014]. The present study is one step in this direction.

Optimizing. The size of the design space of menu techniques is so large that precisely testing and comparing all of them is not realistic. The development of optimization methods able to explore some subsets of the design space in a systematic way is a promising direction to provide optimal designs. However, predictive models are a requirement for optimization methods. This probably explains why only a couple of optimization methods have been proposed for menus [Bailly et al. 2014].

Designing. Technology advances continuously provide new opportunities to create new menu techniques or to improve existing ones. For instance, multi-touch technologies enlarged the design space by letting users select items using multiple fingers and two hands. As mentioned earlier, this will lead to the identification of new properties in the design space.

Implementing. Implementing menus is a challenging task. As a result, most techniques proposed in the literature have not been completely implemented and/or have not been made publicly available. This also partly explains why few research techniques have been released in commercial products. One promising direction is the elaboration of a toolkit making it possible to create menus more easily, and thus to test various configurations in order to compare techniques.

Evaluating. Finally, a benchmark of tasks, criteria and techniques is needed to allow comparing menu techniques precisely. Evaluation setups tend to vary a lot from one study to another. This requires a better understanding of menu usage in order to derive more realistic tasks. As a step in this direction, we developed a tool [Bailly and Malacria 2013] that inspects the content of all menu systems on MAC OS X to inform about the consistency of menus between applications. Furthermore, MozillaLabs provides logging data that can help learning how users select commands in Firefox (http://blog.mozilla.org/labs/2010/02/menu-item-study/). The elaboration of benchmarks is a necessary step to increase the quality of menus and the reliability of user studies.

ACKNOWLEDGMENTS

A special thanks to Michel Beaudouin-Lafon, Sylvain Malacria, Antti Oulasvirta and Catherine Plaisant.

REFERENCES

G. Abowd, J. Coutaz, and L. Nigay. 1992. Structuring the space of interactive system properties. In *EHCI, Elservier Science Pub*. 113–128.

J. Accot and S. Zhai. 1997. Beyond fitts law: Models for trajectory-based HCI tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'97)*. ACM, New York, NY, 295–302.

D. Ahlström. 2005. Modeling and improving selection in cascading pull-down menus using Fitts law, the steering law and force fields. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'05)*. ACM, New York, NY, 61–70.

D. Ahlström, R. Alexandrowicz, and M. Hitz. 2006. Improving menu interaction: A comparison of standard, force enhanced and jumping menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'06)*. ACM, New York, NY, 1067–1076.

D. Ahlström, A. Cockburn, C. Gutwin, and P. Irani. 2010. Why it's quick to be square: Modelling new and existing hierarchical menu designs. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems (CHI'10)*. ACM, New York, NY, 1371–1380.

J. Anderson and C. Lebiere. 1998. *The Atomic Components of Thought*. LEA, Mahwah, NJ.

G. Apitz and F. Guimbretière. 2004. Crossy: A crossing-based drawing application. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST'04)*. ACM, New York, NY, 3–12.

C. Appert, M. Beaudouin-lafon, and W. Mackay. 2004. Context matters: Evaluating interaction techniques with the CIS model. In *Proceedings People and Computers XVIII—Design for Life (HCI'04).* Springer Verlag, 279–295.

C. Appert and J. D. Fekete. 2006. Orthozoom scroller: 1D multi-scale navigation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'06).* ACM, New York, NY, 21–30.

C. Appert and S. Zhai. 2009. Using strokes as command shortcuts: Cognitive benefits and toolkit support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*. ACM, New York, NY, 2289–2298.

G. Bailly, E. Lecolinet, and L. Nigay. 2007. Wave menus: Improving the novice mode of marking menus. In *Proceedings of the International Conference on Human-Computer Interaction (INTERACT'07).* Springer, 475–488.

G. Bailly, E. Lecolinet, and L. Nigay. 2008. Flower menus: A new type of marking menu with large menu breadth, within groups and efficient expert mode memorization. In *Proceedings of the Working Conference on Advanced Visual interfaces (AVI'08).* ACM, New York, NY, 15–22.

G. Bailly, A. Demeure, E. Lecolinet, and L. Nigay. 2008. Multi-touch menu (MTM). *Conférence IHM 2008, 20*ème *Conférence Francophone sur l'Interaction Homme-Machine*. ACM Press.

G. Bailly, E. Lecolinet, and Y. Guiard. 2010. Finger-count and radial-stroke shortcuts: 2 Techniques for augmenting linear menus on multi-touch surfaces. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems (CHI'10)*. ACM, New York, NY, 591–594.

G. Bailly, D. Vo, E. Lecolinet, and Y. Guiard. 2011. Gesture-aware remote controls: Guidelines and interaction technique. In *Proceedings of the 13th International Conference on Multimodal Interfaces (ICMI'11)*. ACM, New York, NY, 263–270.

G. Bailly, J. Müller, and E. Lecolinet. 2012. Design and evaluation of finger-count interaction: Combining multitouch gestures and menus. *Int. J. Hum.-Comput. Stud.* 70, 10 (October 2012), 673–689.

G. Bailly, T. Pietrzak, J. Deber, and D. J. Wigdor. 2013. Métamorphe: Augmenting hotkey usage with actuated keys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, New York, NY, 563–572.

G. Bailly, A. Oulasvirta, T. Kötzing, and S. Hoppe. 2013. MenuOptimizer: Interactive optimization of menu systems. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST'13)*.

G. Bailly and A. Oulasvirta. 2014. Toward optimal menu design. *Interactions* 21, 4, July 2014, 40–45.

G. Bailly and S. Malacria. 2013. MenuInspector: Outil pour l'analyse des menus et cas d'étude. In *Proceedings of the 25th Conference on l'Interaction Homme-Machine (IHM'13)*. ACM, New York, NY.

R. Balakrishnan and P. Pranay Patel. 1998. The padmouse: Facilitating selection and spatial positioning for the non-dominant hand. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'98).* ACM Press/Addison-Wesley, New York, NY, 9–16.

L. Barfield. 1993. *The User Interface Concepts and Design*. Addison Wesley.

J. M. C. Bastien and D. Scapin. 1993. Ergonomic criteria for the évaluation of human-computer interfaces. *Institut National de Recherche en Informatique et en Automatique.*

O. Bau and W. E. Mackay. 2008. OctoPocus: A dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology (UIST'08)*. ACM, New York, NY, 37–46.

M. Beaudouin-Lafon. 2000. Instrumental interaction: An interaction model for designing post-WIMP user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'00).* ACM Press, New -York, 446–453.

B. B. Bederson. 2000. Fisheye menus. In *Proceedings of the 13th Annual ACM Symposium on User interface Software and Technology (UIST'00)*. ACM, New York, NY, 217–225.

N. O. Bersen. 1996. A taxonomy of input modalities. *Amodeus Project Deliverable TM/WP22*.

W. L. Bewley, T. L. Roberts, D. Schroit, and W. L. Verplank. 1983. Human factors testing in the design of Xerox's 8010 Star office workstation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'83)*. ACM, New York, NY, 72–77.

R. Blanch, Y. Guiard, and M. Beaudouin-Lafon. 2004. Semantic pointing: Improving target acquisition with control-display ratio adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'04)*. ACM, New York, NY, 519–526.

F. Block, H. Gellersen, and N. Villar. 2010. Touch-display keyboards: Transforming keyboards into interactive surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'10)*. ACM, New York, NY, 1145–1154.

D. A. Bowman and C. A. Wingrave. 2001. Design and evaluation of menu systems for immersive virtual environments. In *Proceedings of the Virtual Reality 2001 Conference (VR'01)*. IEEE Computer Society, Washington, DC, 149.

J. Bowes, D. Dearman, and R. Perkins. 2003. Transparency for item highlighting. *Poster Presentation at Graphics Interface 2003*, Halifax, Nova Scoita.

P. Brandl, J. Leitner, T. Seifried, M. Haller, B. Doray, and P. To. 2009. Occlusion-aware menu design for digital tabletops. In *Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA'09)*. ACM, New York, NY, 3223–3228.

W. Buxton. 1990. A three-state model of graphical input. In *Proceedings of the IFIP TC13 3rd International Conference on Human-Computer Interaction (INTERACT'90)*. North-Holland Publishing Co, Amsterdam, The Netherlands, 449–456.

W. A. Buxton. 1995. Chunking and phrasing and the design of human-computer dialogues. In *Human-Computer Interaction: Toward the Year 2000,* R. M. Baecker, J. Grudin, W. A. Buxton, and S. Greenberg (Eds.). Morgan Kaufmann Publishers, San Francisco, CA, 494–499.

M. Byrne. 2001. ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *IJHCS*, 55. 41–84.

J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. 1988. An empirical comparison of pie vs. linear menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'88)*. ACM, New York, NY, 95–100.

J. Cance, M. Collomb, and M. Hascoët. 2006. Accelerating Object-command transitions with pie menus. *Proceeding of 3rd International Conference on Enactive Interfaces (Enactive'06).* 109–110.

S. Card, T. Moran, and A. Newell. 1983. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates.

S. K. Card, J. D. Mackinlay, and G. G. Robertson. 1990. The design space of input devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Empowering People (CHI'90)*. ACM, New York, 117–124.

J. M. Carroll and M. B. Rosson. 1987. Paradox of the active user. In *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction,* J. M. Carroll (Ed.). MIT Press, Cambridge, MA, 80–111.

L. D. Catledge and J. E. Pitkow. 1995. Characterizing browsing strategies in the World-Wide Web. *Comput. Netw.* ISDN Syst. 27, 6 (Apr. 1995), 1065–1073.

M. C. Chuah and S. F. Roth. 1996. On the semantic of interactive visualizations. In *Proceedings of the IEEE Symposium on information Visualization*. INFOVIS. IEEE Press, Washington, DC, 29.

X. Chen, G. Bailly, D. Brumby, A. Oulasvirta, and A. Howes. 2015. The emergence of interactive behavior: A model of rational menu search. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15).* ACM, New York, NY, 4217–4226.

A. Cockburn and A. Gin. 2006. Faster cascading menu selections with enlarged activation areas. In *Proceedings of Graphics Interface 2006 (GI'06)*. Canadian Information Processing Society, Toronto, Ontario, Canada, 65–71.

A. Cockburn, C. Gutwin, and S. Greenberg. 2007. A predictive model of menu performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (*CHI'07). ACM, New York, NY, 627–636.

A. Cockburn and C. Gutwin. 2009. A predictive model of human performance with scrolling and hierarchical lists. *Hum.-Comput. Interact.* 24, 3, 273–314.

A. Cockburn, C. Gutwin, J. Scarr, and S. Malacria. 2014. Supporting novice to expert transitions in user interfaces. *ACM Comput. Surv.* 47, 2, Article 31 (November 2014), 36

R. Dachselt and A. Hübner. 2007. Virtual environments: Three-dimensional menus: A survey and taxonomy. *IEEE Comput. Graphics* 31, 1, 53–65.

R. F. Dillon, J. D. Edey, and J. W. Tombaugh. 1990. Measuring the true cost of command selection: techniques and results. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Empowering People (CHI'90)*. ACM, New York, NY, 19–26.

C. Essert-Villard and A. Capobianco. 2009. Hardborders: A new haptic approach for selection tasks in 3D menus. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology (VRST'09),* Steven N. Spencer (Ed.). ACM, New York, NY, 243–244.

L. Findlater and J. McGrenere. 2004. A comparison of static, adaptive, and adaptable menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'04)*. ACM, New York, NY, 89–96.

L. Findlater, K. Moffatt, J. McGrenere, and J. Dawson. 2009. Ephemeral adaptation: The use of gradual onset to improve menu selection performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09).* ACM, New York, NY, 1655–1664.

P. M. Fitts. 1954. The information capacity of the human mote; system in controlling the amplitude of movement. *J. Exp. Psychol.* 1954, 47, 381–391.

J. D. Foley, A. Van Dam, S. K. Feiner, and J. F. Hughes. 1990. *Computer Graphics: Principles and Practice*, 2nd ed. Addison Wesley Longman.

J. D. Foley, V. L. Wallace, and P. Chan. 1984. The human factors of computer graphics interaction techniques. *IEEE Comput. Graphics Appl*. 4, 11 (1984), 13–48.

J. Francone, G. Bailly, E. Lecolinet, N. Mandran, and L. Nigay. 2010. Wavelet menus on handheld devices: Stacking metaphor for novice mode and eyes-free selection for expert mode. In *Proceedings of the International Conference on Advanced Visual Interfaces (AVI'10)*. ACM, New York, NY, 173–180.

T. Grossman, P. Dragicevic, and R. Balakrishnan. 2007. Strategies for accelerating on-line learning of hotkeys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'07)*. ACM, New York, NY, 1591–1600.

F. Guimbretiére and T. Winograd. 2000. FlowMenu: Combining command, text, and data entry. In *Proceedings of the 13th Annual ACM Symposium on User interface Software and Technology (UIST'00)*. ACM, New York, NY, 213–216.

F. Guimbretière, A. Martin, and T. Winograd. 2005. Benefits of merging command selection and direct manipulation. *ACM Trans. Comput.-Hum. Interact.* 12, 3 (Sep. 2005), 460–476.

F. Guimbretiere and Nguyen. 2012. Bimanual marking menu for near surface interactions. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems (CHI'12)*. ACM, New York, NY, 825–828.

C. Gutwin, A. Cockburn, J. Scarr, S. Malacria, and S. C. Olson. 2014. Faster command selection on tablets with FastTap. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'14).* ACM, New York, NY, 2617–2626.

S. Harada, J. Wobbrock, and J. Landay. 2007. Voicedraw: A hands-free voice-driven drawing application for people with motor impairments. In *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility (Assets'07).* ACM, New York, NY, 7–34.

T. E. Halverson. 2008. An active vision computational model of visual search for human-computer interaction. *Ph.D. Dissertation.* University of Oregon, Eugene, OR, Advisor(s) Anthony J. Hornof. AAI3346648.

M. S. Hancock and K. S. Booth. 2004. Improving menu placement strategies for pen input. In *Proceedings of Graphics Interface 2004 (GI'04)*. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 221–230.

M. G. Helander, T. K. Landauer, and P. V. Prabhu (Ed.). 1997. *Handbook of Human-Computer Interaction*. 2nd ed. Elsevier Science Inc.

W. Hick. 1952. On the rate of gain of information. *J. Experimental Psychol.* 4. 11–36.

T. Hesselmann, S. Flöring, and M. Schmitt. 2009. Stacked half-pie menus: Navigating nested menus on interactive tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS'09)*. ACM, New York, NY, 173–180.

K. Hinckley and M. Sinclair. 1999. Touch-sensing input devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: the CHI Is the Limit (CHI'99)*. ACM, New York, NY, 223–230.

K. Hinckley, F. Guimbretiere, P. Baudisch, R. Sarin, M. Agrawala, and E. Cutrell. 2006. The springboard: multiple modes in one spring-loaded control. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'06)*. ACM, New York, NY, 181–190.

A. J. Hornof and D. E. Kieras. 1997. Cognitive modeling reveals menu search in both random and systematic. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'97).* ACM, New York, NY, 107–114.

A. J. Hornof and D. E. Kieras. 1999. Cognitive modeling demonstrates how people use anticipated location knowledge of menu items. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'99)*. ACM, New York, NY, 410–417.

A. Howes. 1994. A model of the acquisition of menu knowledge by exploration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Celebrating Interdependence (CHI'94)*. ACM, New York, NY, 445–451.

S. Huot and E. Lecolinet. 2007. Archmenu et thumbmenu: Contrôler son dispositif mobile sur le pouce. *19ème Conférence Francophone Sur l'Interaction Homme-Machine (IHM'07)*,

S. Huot, M. Nancel, and M. Beaudouin-Lafon. 2008. Push menu: Extending marking menus for pressure-enabled input services. *Rapport de Recherche* n 1502.

R. Hyman. 1953. Stimulus information as a determinant of reaction time. *J. Exp. Psych.* 45, 188–196.

ISO. 1998. ISO/DIS 9241-14. Ergonomics requierements for office work with visual display terminals (VDTs) – part 14: Menu dialogues. International Organization for Standardization.

P. Isokoski and M. Käki. 2002. Comparison of two touchpad-based methods for numeric entry. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'02)*. ACM, New York, NY, 25–32.

J. A. Jacko and G. Slavendy. 1996. Hierarchical menu design: breadth, depth and task complexity. *Perceptual and Motor Skills* 82, 1187–1201.

M. R. Jackobsen and K. Hornaek. 2007. Transient visualizations. In *Proceedings of the 2007 Conference of the Computer-Human Interaction, Special Interest Group (CHISIG) of Australia on Computer-Human Interaction: Design: Activities, Artifacts and Environments*. ACM, 251, 69–76.

R. H. Jackoby and S. R. Ellis. 1992. Using virtual menus in a virtual environment. *Visual Data Interpretation.* V1668 i1, 39–48.

B. E. John and D. E. Kieras. 1996. The GOMS family of user interface analysis techniques: Comparison and contrast.. *ACM Trans. Comput. Human Interact.* 3, 4, 320–351.

Y. Kammerer, K. Scheiter, and W. Beinhauer. 2008. Looking my way through the menu: The impact of menu design and multimodal input on gaze-based menu selection. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications (ETRA'08)*. ACM, New York, NY, 213–220.

D. E. Kieras and D. E. Meyer. 1997. An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Hum.-Comput. Interact.* 12, 4, 391–438.

J. I. Kiger. 1984. The depth/breadth trade-off in the design of menu-driven user interfaces. *Int. J. Man-Mach. Stud.* 20, 2, 201–213.

M. Kobayashi and T. Igarashi 2003. Considering the direction of cursor movement for efficient traversal of cascading menus. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology (UIST'03)*. ACM, New York, NY, 91–94.

G. Kurtenbach and W. Buxton. 1991. Issues in combining marking and direct manipulation techniques. In *Proceedings of the 4th Annual ACM Symposium on User Interface Software and Technology (UIST'91)*. ACM, New York, NY, 137–144.

G. Kurtenbach and W. Buxton. 1993. The limits of expert performance using hierarchic marking menus. In *Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems (CHI'93)*. 482–487.

G. Kurtenbach. 1993. The design and evaluation of marking menus., PhD., Thesis, Dept. of Computer Science, University of Toronto.

G. Kurtenbach and W. Buxton. 1994. User learning and performance with marking menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'94).* ACM, New York, NY, 258–264.

Kurtenbach 1997. Display and control of menus with radial and linear portions. Filed April 1, 1997, *patent* #5,926,178.

G. Kurtenbach, G. W. Fitzmaurice, R. N. Owen, and T. Baudel. 1999. The hotbox: Efficient access to a large number of menu-items. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: the CHI Is the Limit (CHI'99)*. ACM, New York, NY, 231–237.

J. T. Lai, R. Anderson, and Y. Li. 2005. A chinese input technique using pressure-sensitive marking menus. In *Proceedings of UIST'05*.

T. K. Landauer and D. W. Nachbar. 1985. Selection from alphabetic and numeric menu trees using a touch screen: breadth, depth, and width. *SIGCHI Bull.* 16, 4, 73–78.

D. M. Lane, H. A. Napier, S. C. Pers, and A. Sandor. 2005. Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts. *Int. J. Human-Comput. Interact.* 16, 2, 133–144.

K. Larson and M. Czerwinski. 1998. Web page design: Implications of memory, structure and scent for information retrieval. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM/Addison-Wesley, New York, NY, 25–32.

E. Lecolinet and D. Nguyen. 2006. Représentation focus+contexte de listes hiérarchiques zoomables. In *Proceedings of the 18th International Conferenceof the Association Francophone D'interaction Homme-Machine (IHM'06)*. ACM, New York, NY, 195–198.

R. Lee and D. R. Raymond. 1993. Menu-driven systems. *Encyclopedia of Microcomputers* 11, 101–127.

E. Lee and J. McGregor. 1985. Minimizing user search time in menu retrieval systems. *Human Factors* 27, 2, 1985, 157–162.

D. S. Lee, and W. C. Yoon. 2004. Quantitative results assessing design issues of selection support menus. *Int. J. Ind, Ergon.* 33, 41–52.

D. Leithinger and M. Haller. 2007. Improving menu interaction for cluttered tabletop setups with user-drawn path menus. In *Proceedings of the 2nd Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'07)*. 121–128

G. J. Lepinski, T. Grossman, and G. Fitzmaurice. 2010. The design and evaluation of multitouch marking menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'10)*. ACM, New York, NY, 2233–2242.

Y. Li. 2012. Gesture search: Random access to smartphone content. *IEEE Pervasive Comput.* 11, 1 (January 2012), 10–13.

B. Liu, G. Francis, and G. Salvendy. 2002. Applying models of visual search to menu design. *International J. Hum. Comput. Stud.* 2002, 56.

Logickeyboard. 2015. Homepage. Retrieved from www.logickeyboard.com.

W. E. Mackay. 2002. Which interaction technique works when? floating palettes, marking menus and toolglasses support different task strategies. In *Proceedings of the ACM Conference on Advanced Visual Interfaces (AVI'02)*. 203–208.

H. E. McLoone, K. Hinckley, and E. Cutrell. 2003. Bimanual interaction on the Microsoft Office keyboard. In *Proceedings of INTERACT* 2003.

MAK. 2010. Microsoft Adaptive Keyboard. Retrieved from http://www.microsoft.com/appliedsciences/content/projects/uist.aspx.

A. Marcus, N. Silonich, and L. Thompson. 1994. *The Cross-GUI Handbook: For Multiplatform User Interface Design*. Addison-Wesley Professional.

S. Malacria, G. Bailly, J. Harrison, A. Cockburn, and C. Gutwin. 2013. Promoting hotkey use through rehearsal with ExposeHK. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, New York, NY, 573–582.

S. Malacria, J. Scarr, A. Cockburn, C. Gutwin, and T. Grossman. 2013. Skillometers: Reflective widgets that motivate and help users to improve performance. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST'13)*. ACM, New York, NY, 321–330.

J. McCall. 1977. Factors in software quality. General Electric Eds.

M. J. McGuffin, N. Burtnyk, and G. Kurtenbach. 2002. FaST sliders: Integrating marking menus and the adjustment of continuous values. In *Proceedings of the Symposium on Graphics Interface*. 35–41.

M. McGuffin, N. Burtnyk, and G. Kurtenbach. 2002. FaST sliders: Integrating marking menus and the adjustment of continuous values. In *Graphics Interface*.

G. A. Miller. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychol. Rev.* 63, 81–97

B. A. Myers. 1998. A brief history of human-computer interaction technology. *Interactions* 5, 2, 44–54.

M. Nancel and M. Beaudouin-Lafon 2008. Extending Marking menus with integral dimensions: Application to the dartboard menu. Technical Report of LRI, n 1503.

K. L. Norman. 1991. *The Psychology of Menu Selection: Designing Cognitive Control at the Human / Computer Interface.* Greenwood Publishing Group Inc.

E. L. Nilsen. 1991. Perceptual-motor control in human}computer interaction (Technical Report Number 37). Ann Arbor. MI: The Cognitive Science and Machine Intelligence Laboratory, the University of Michigan.

J. Nielsen. 1993. *Usability Engineering*. Academic Press Professional, 362

D. A. Norman. 1999. Affordance, conventions, and design. *Interactions* 6, 3, 38–43.

I. Oakley and J. Park. 2007. Designing eyes-free interaction. In *Proceedings of HAID 2007*. LNCS, 4813.

D. L. Odell, R. C. Davis, A. Smith, and P. K. Wright. 2004. Toolglasses, marking menus, and hotkeys: a comparison of one and two-handed command selection techniques. In *Proceedings of the 2004 Graphics*

Interface Conference (GI'04). Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 17–24.

Optimus. 2016. Homepage. Retrieved from http://www.artlebedev.com/everything/optimus/.

K. R. Paap and R. J. Roske-Hofstrand. 1986. The optimal number of menu items per panel, Human Factors. 28, 377–386.

P. Parhi, A. K. Karlson, and B. B. Bederson. 2006. Target size study for one-handed thumb use on small touchscreen devices. In *Proceedings of the 8th Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI'06)*. ACM, New York, NY, 203–210.

R. L. Potter, L. J. Weldon, and B. Shneiderman. 1988. Improving the accuracy of touch screens: An experimental evaluation of three strategies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'88)*. ACM, New York, NY, 27–32.

S. Pook, E. Lecolinet, G. Vaysseix, and E. Barillot. 2000. Control menus: Excecution and control in a single interactor. In *CHI'00 Extended Abstracts on Human Factors in Computing Systems (CHI'00)*. ACM, New York, NY, 263–264.

J. Raskin. 2000. *The Human Interface: New Directions for Designing Interactive Systems*. ACM/Addison Wesley.

J. Rasmussen. 1983. Skills, rules and knowledge: Signals, signs and symbols and other distinctions in human performance models. *IEEE Trans. Syst. Man Cybern.* 13, 257–266.

J. Rekimoto, T. Ishizawa, C. Schwesig, and H. Oba. 2003. PreSense: Interaction techniques for finger sensing input devices. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology (UIST'03)*. ACM, New York, NY, 203–212.

X. Ren, J. Yin, T. Oya, and Y. Liu. 2008. Enhancing pie-menu selection with pen pressure. In *Proceedings of the 2008 3rd International Conference on Innovative Computing Information and Control*. IEEE Computer Society, Washington, DC, 364.

A. Roudaut, E. Lecolinet, and Y. Guiard. 2009. MicroRolls: Expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. In *Proceedings of CHI'09*. ACM.

A. Roudaut, G. Bailly, E. Lecolinet, and L. Nigay. 2009. Leaf Menus: Linear Menus with Stroke Shortcuts for Small Handheld Devices. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I,* T. Gross, J. Gulliksen, P. Kotzé, L. Oestreicher, P. Palanque, R. O. Prates, and M. Winckler (Eds.). Lecture Notes In Computer Science, vol. 5726. Springer-Verlag, Berlin, 616–619.

Q. Roy, S. Malacria, Y. Guiard, E. Lecolinet, and J. Eagan. 2013. Augmented letters: Mnemonic gesture-based shortcuts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*. ACM, New York, NY, 2325–2328.

J. M. Rubio and P. Janecek. 2002. Floating pie menus: Enhancing the functionality of Contextual Tools. *Poster UIST'02*.

K. Samp and S. Decker. 2010. Supporting menu design with radial layouts. In *Proceedings of the International Conference on Advanced Visual Interfaces (AVI'10)*, Giuseppe Santucci (Ed.). ACM, New York, NY, 155–162.

A. Sears, and B. Shneiderman. 1994. Split menus: Effectively using selection frequency to organize menus. *ACM Trans. Comput.-Hum. Interact.* 1, 1, 27–51.

J. Scarr, A. Cockburn, C. Gutwin, and P. Quinn. 2011. Dips and ceilings: Understanding and supporting transitions to expertise in user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'11).* ACM, New York, NY, 2741–2750.

H. Simon 1987. Satisficing. The new palgrave dictionary of economics. 243–245. London: Macmillan Press.

B. Shneiderman. 1992. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 2nd ed. Addison Wesley.

K. Snowberry, S. R. Parkinson, and N. Sisson. 1983. Computer display menus. *Ergonomics* 26, 699–712.

K. M. Snyder, A. J. Happ, L. Malcus, K. R. Paap, and J. R. Lewis. 1985. Using cognitive models to create menus. In *Proceedings of the Human Factors Society 29th Annual Meeting*. Human Factors Society, 655–658.

E. Soliz and W. B. Paley. 2003. A Re-Interpretation of marking menus: The usage of gestalt theory as cognitive tools. *Poster of the 16th Annual ACM Symposium on User Interface and Software Technology*. ACM.

H. Song, T. Grossman, G. Fitzmaurice, F. Guimbretiere, A. Khan, R. Attar, and G. Kurtenbach. 2009. PenLight: Combining a mobile projector and a digital pen for dynamic visual overlay. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*. ACM, New York, NY, 143–152.

A. N. Sulaiman and P. Olivier. 2008. Attribute gates. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology (UIST'08)*. ACM, New York, NY, 57–66.

S. Tak, P. Westendorp, and I. van Rooij. 2013. Satisficing and the use of keyboard shortcuts: Being good enough is enough? *Interacting with Computers* 25, 5 (2013), 404–416.

E. Tanvir, J. Cullen, P. Irani, and A. Cockburn. 2008. AAMU: Adaptive activation area menus for improving selection in cascading pull-down menus. In *Proceeding of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems (CHI'08)*. ACM, New York, NY, 1381–1384.

F. Tian, L. Xu, H. Wang, X. Zhang, Y. Liu, V. Setlur, and G. Dai. 2008. Tilt menu: Using the 3D orientation information of pen devices to extend the selection capability of pen-based user interfaces. In *Proceeding of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems (CHI'08)*. ACM, New York, NY, 1371–1380.

T. Tsandilas and M. C. Schraefel. 2007. Bubbling menus: a selective mechanism for accessing hierarchical drop-down menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'07)*. ACM, New York, NY, 1195–1204.

T. S. Tullis. 1985. Designing a menu-based interface to an operating system. *SIGCHI Bull.* 16, 4, 79–84.

N. Walker and J. B. Smelcer. 1990. A comparison of selection time from walking and pull-down menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems Empowering People (CHI'90)*. ACM, New York, NY, 221–226.

R. Walter, G. Bailly, N. Valkanova, and J. Müller. 2014. Cuenesics: Using mid-air gestures to select items on interactive public displays. In *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services (MobileHCI'14)*. ACM, New York, NY, 299–308.

J. Waterworth and M. Chignell. 1991. Model of information exploration. *Hypermedia* 3, 1, 1991, 35–38.

M. V. Welie, G. G. Van der Veer, and A. eliens. 1999. Breaking down usability. In *Proceeding of Interact'99*, 613–620.

J. O. Wobbrock, M. R. Morris, and A. D. Wilson. 2009. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*. ACM, New York, NY, 1083–1092.

A. Worden, N. Walker, K. Bharat, and S. Hudson. 1997. Making computers easier for older adults to use: Area cursors and sticky icons. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'97)*. ACM, New York, NY, 266–271.

M. Wu and R. Balakrishnan. 2003. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology (UIST'03)*. ACM, New York, NY, 193–202.

P. Zaphiris. 2002. Age differences and the depth-breadth tradeoff in hierarchical online information systems. Doctoral Thesis. UMI Order Number: AAI3047597. Wayne State University.

S. Zhai, P. O. Kristensson, C. Appert, T. H. Anderson, and X. Cao. 2012. Foundational issues in touch-surface stroke gesture design—An integrative review. *Hum. Comput. Interact.* 5, 2, 116

S. Zhao and R. Balakrishnan. 2004. Simple vs. compound mark hierarchical marking menus. In *Proceedings of the 17th Annual ACM Symposium on User interface Software and Technology (UIST'04)*. ACM, 33–42.

S. Zhao, M. Agrawala, and K. Hinckley. 2006. Zone and polygon menus: Using relative position to increase the breadth of multi-stroke marking menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'06)*. ACM, New York, NY, 1077–1086.

S. Zhao, P. Dragicevic, M. Chignell, R. Balakrishnan, and P. Baudisch. 2007. Earpod: Eyes-free menu selection using touch input and reactive audio feedback. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'07)*. ACM, New York, NY, 1395–1404.

S. Zhao. 2008. Earpod: Efficient hierarchical eyes-free menu selection. Ph.D., Thesis, Dept. of Computer Science, University of Toronto.

S. Zhao and M. Chignell. 2007. Using glyphs to facilitate transition from hierarchical selection to gesturing. In *Proceedings of the 10th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems (IFAC-HMS'07)*.

G. Zipf. 1949. *Human Behaviour and the Principle of Least-Effort*. Addison-Wesley, Cambridge MA.

Received May 2016; revised August 2016; accepted September 2016