SAM: The Swiss Army Menu

David Bonnet
Université Paris-Sud - INRIA - CNRS
Orsay, France
bonnet@lri.fr

Caroline Appert
Université Paris-Sud - CNRS - INRIA
Orsay, France
appert@lri.fr

RESUME

Cet article présente le Swiss Army Menu (SAM), un menu circulaire permettant de rendre accessible un très grand nombre de fonctions sur un seul petit écran tactile. Le design de SAM repose sur quatre types d'items différents, la possibilité de naviguer dans des hiérarchies d'items et un contrôle qui repose simplement sur de petits mouvements du pouce. SAM offre ainsi un ensemble important de fonctions qui aurait typiquement requis un nombre de widgets impossible à afficher en même temps sur un petit écran.

MOTS CLES

Dispositif mobile, technique de menu, widget

ABSTRACT

This article introduces the Swiss Army Menu (SAM), a radial menu that enables a very large number of functions on a single small tactile screen. The design of SAM relies on four different kinds of items, support for navigating in hierarchies of items and a control based on small thumb movements. SAM can thus offer a set of functions so large that it would typically have required a number of widgets that could not have been displayed in a single viewport at the same time.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms

Design, Human Factors

Keywords

Mobile device, menu technique, widget

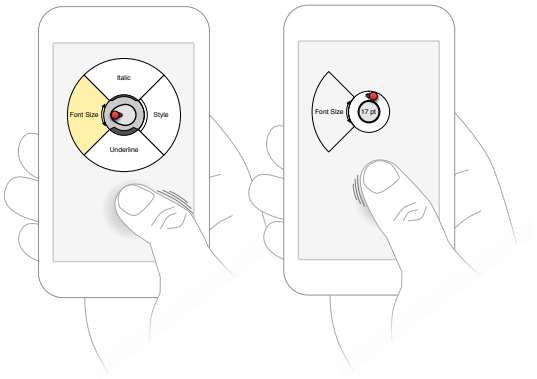


Figure 1: The Swiss Army Menu allows the user to navigate in a large and diverse command set without any occlusion problem.

1. INTRODUCTION

Interaction on mobile devices mainly relies on large buttons and multi touch gestures, involving the two hands of the user. One hand holds the device to allow the other hand to gesture or reach a button anywhere on the screen. Some interactions with the user's environment then become limited. For example, holding a paper document in one hand while using a mobile application (e.g. web browsing) with the other hand at the same time is almost impossible.

HCI researchers have designed powerful techniques that only require the user's thumb (e.g., [6] or [7]) to allow the user to use the mobile device in the casual way illustrated by Figure 1. However none of these techniques fully addresses both of the following requirements at the same time:

- P_1 Easy navigation and exploration in the whole set of commands: the thumb must be able to easily *reach* any command and never *hide* its label;
- P_2 Large *capacity*: the user must be able to invoke the whole set of commands in an application. This means a potentially large set of commands including commands that take as input a continuous parameter.

This article introduces the Swiss Army Menu (SAM, Figure 1), a radial menu that proposes different kinds of items so that it can replace a large set of commonly-used widgets (buttons, checkboxes, radio buttons, sliders, scrollbars and lists). The control of SAM makes use of a *puck* the user remotely moves (i.e., indirect pointing) in a limited area so

that any command can be easily reached while preserving the visibility of the other labels in the menu.

2. RELATED WORK

Directly interacting with fingers on a small display is prone to **occlusion** issues because of the size of a finger relative to the size of the screen. One drastic solution consists in moving all touch interactions to the back of the device [2], by making the rear surface touch-sensitive. However, when designing this technique, Baudisch and Chu were considering a two hand use: the user holds the tiny display in one hand while the index of the other hand interacts on the backside. Others have designed solutions for more standard devices solely equipped with a one sided touch-screen. For example, MagStick [9] is a telescopic stick that allows to acquire a small target without occluding it. When the user touches the screen, a pointer appears that the user remotely controls. Indirect pointing combined with symmetrical control ensures the visibility of the targeted object but also increases the user's cognitive load, at least in the learning phase.

In addition to minimizing occlusion, MagStick allows the thumb to access objects that **could not be reached** without it. Regarding this issue, ThumbSpace [7] brings a different solution by providing a soft touchpad superimposed on the display when the user presses a given physical button. Once calibrated in order to accommodate the user's characteristics (e.g., thumb length), the entire screen space is mapped into a smaller region which can be easily accessed by the thumb. MagStick and ThumbSpace are powerful pointing techniques but do not explicitly address how a small display can accommodate a large set of commands.

ArchMenu and ThumbMenu [6] are both semi-circular menus designed to make **several commands** accessible with small thumb moves. The layout of the items ensures that they all fall within thumb's reach. Both menus provide an offset cursor to avoid the thumb occluding items of interest. They also allow the user to explore the menu hierarchy by displaying the pointed submenu in an extra layer above the parent item. However, the semi-circular shape limits the number of items these menus can contain. Conversely, the Wavelet Menu [4], is a full circular menu that features an inverted concentric layout, where submenus are shown at the center while the parent menus move outward, but suffers from usability problems due to occlusion.

In summary, each of these techniques solves a specific problem at the expense of others. The goal of SAM's design is to meet all the requirements: limit occlusion and accommodate a large set of commands easy to reach.

3. SWISS ARMY MENU DESIGN

SAM is presented as a radial menu [8]. When invoked, it pops up at a given location in the upper part of the screen. A finger then remotely controls a pointer (a *puck*) in an indirect way (Figure 2) to interact with the menu items. In most cases, the mobile will be held in the standard way depicted in Figure 1 so that the thumb controls from the lower part of the screen. While SAM can still be controlled by any finger from any location, it always appears at the same location so as to avoid common problems radial menus face with screen edges.

The best way to invoke SAM is up to the designer according to the application he considers. He can for example choose to use a double tap event, a given gesture, a shake event, etc.

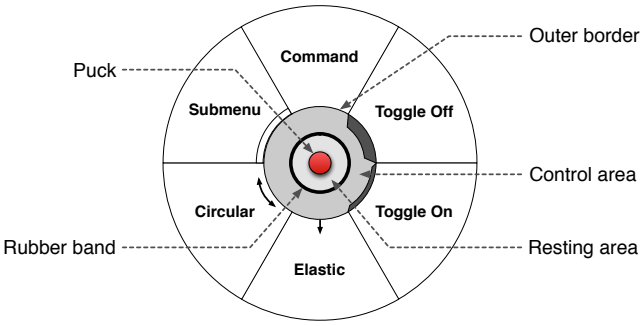
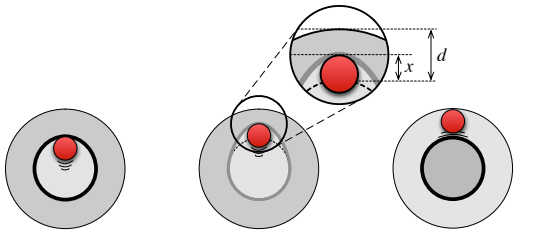


Figure 2: The design of SAM



(a) *Puck touch* (b) *Puck stretch*$_x$ (c) *Puck cross*

Figure 3: Event types

The combination of indirect control and a small central area addresses two issues mentioned in the introduction: the thumb can *reach* any command in the menu without *occluding* it.

3.1 Indirect control

The central area is composed of two concentric circles: the *rest area* and the *control area* (Figure 2). Both are separated by a rubber band. The menu responds to events triggered by interactions between the puck and the rubber band. There are three types of events:

- *Puck touch*: The puck touches the rubber band (Figure 3-(a));
- *Puck stretch*$_x$: The puck stretches the rubber band by a distance x (Figure 3-(b)). When $x \geq d$, where d is the distance between the rubber band and the outer border, puck movements are constrained by the rigid outer border;
- *Puck cross*: This may occur when the puck stretches the rubber band towards a *circular* item or a *submenu* item (see below). In these cases, when the puck stretches the rubber band by a distance $\geq d$, the puck crosses the rubber band (Figure 3-(c)) and enters the control area. It will cross back to the rest area when it stretches the rubber band by a distance $\geq d$ in the opposite direction (i.e. towards the menu center).

When the rubber band is crossable, its transparency depends on how much it is stretched so as to tell the user he can pass through it in one or the other direction. Furthermore, the respective colors of the rest area and the control area depend on the puck location: the area that contains the puck is always light grayed while the other is dark grayed.

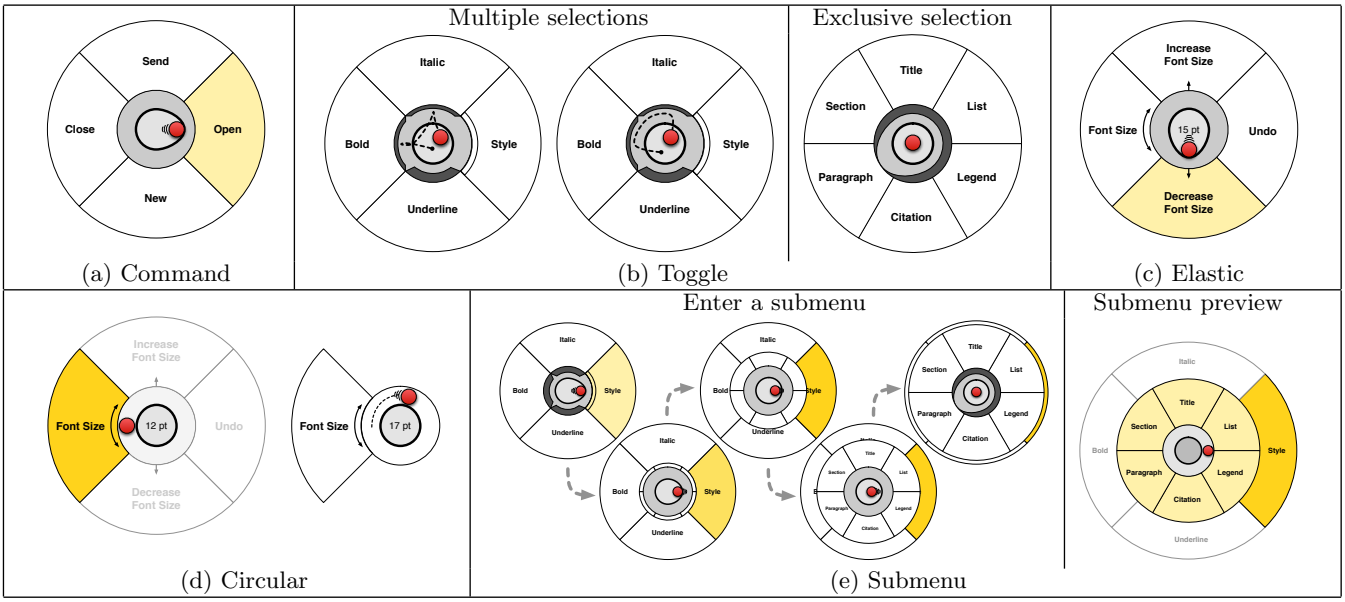


Figure 4: Interacting with the different kinds of items (Command, Toggle, Elastic, Circular) and hierarchy (Submenu)

3.2 Item types

In conjunction with the different types of items SAM can contain, the vocabulary of events described above allows to replace all types of existing widgets. Figure 2 illustrates the different types of items: **Command** items that stand for buttons, **Toggle** items act as checkboxes or radio buttons, **Elastic** items and **Circular** items cover what sliders and scrollbars can do. The type of an item is revealed by a discrete but explicit graphical cue. For example, a one-way arrow is displayed on an **Elastic** item or a curved two-way arrow on a **Circular** item.

Once the menu is invoked, the puck follows thumb movements (as a pointer follows mouse movements) and as soon as it touches the rubber band, the closest item is highlighted whatever its type. The interaction then depends on the item's type as explained below. For all types of items, the interaction with SAM ends when the user releases his finger and the puck is in contact with the rubber band[1]. Otherwise, when the puck is in the rest area, the user is allowed to reposition his thumb if he needs to be in a more comfortable hand posture (in this case, the menu remains displayed).

Command: The command item is the most simple one. While highlighted, the command is triggered when the user releases his thumb.

Toggle: The toggle item can stand for either a checkbox (multiple selections) or a radio button (exclusive selection) as illustrated on Figure 4-(b). The toggle gets selected (resp. deselected) when the puck stretches the rubber band with a distance $\geq d$ (i.e. the puck touches the outer border). The layout of the menu can allow the user to hit several toggles in a continuous move by sliding along the outer border if setting several properties in a row makes sense for the application (for example, bold and italic can be side by side as on Figure 4-(b)). When several properties are grouped, as in the case of radio buttons, the corresponding items are preceded by a string that links them together. Moving the string in the direction of one item will make the string leave other items in the group so as to ensure exclusive selection.

Elastic: The elastic item allows the user to change a value with elastic rate control. The more the rubber band is stretched, the faster the value is increased (or decreased). Note that this first order control of a value is not usually supported by sliders or scrollbars. To avoid accidental setting, the rubber band is a bit more rigid in the direction of an elastic item.

Circular: The circular item allows the user to set a value with a zero order control as a common slider does. As mentioned above, the puck can cross the rubber band when it moves towards this kind of item. Once the rubber band has been crossed, the user can move the puck clockwise (resp. counterclockwise) to increase (resp. decrease) the value. Other menu items disappear to reduce screen clutter. For example, the user sets the font size on Figure 4-(d). The full range of values can be mapped on one circle to allow the user to get a clear idea of the location of the different values. For example, if the range of values is [0,100], the user reaches 25 by sliding a quarter circle away or 50 by going to the diametrically opposed location. Note that, as several recent techniques like [10], a circular control also allows the user to handle a potentially very large continuous range with a high precision by mapping the full range of values on an arbitrary number of circles. The designer will choose the most appropriate mapping (one or several circles) according to the application considered. At any moment, the user can get back to the rest area to cancel the current value setting. Otherwise, the value gets set when the user releases his finger while the puck is in the control area.

Altogether, these different types of items allow the user to select a command, enable/disable properties and set a con-

[1]It is up to the designer to automatically hide the menu once an interaction with a given item has ended or to keep it displayed so as to favor the sequences of several interactions and hide it only in response to an additional explicit invocation event.

tinuous value. With the support for hierarchy, described in the next section, the number of functions SAM can propose is very large (*capacity*).

3.3 Hierarchy

To navigate in hierarchies of items, SAM's design is largely inspired by the Wavelet Menu [4]. As shown in Figure 4-(e), the submenu progressively appears outwards from the center of the menu.

Releasing the thumb while a submenu item is highlighted actually selects the submenu as in the case of a command item. Thus, an item which is not in the first level of the menu is reachable *via* a multi stroke. In order to get back to the parent level, the user simply taps once. A double-tap completely closes SAM.

Also, unlike command items, the user can cross the rubber band when moving towards a submenu to enter a preview mode where he can see the items located in the submenu without fully replacing the current menu. He can get back to the rest area to leave this preview mode at any time. When in preview mode, items that are not submenus are grayed out to allow the user to quickly move from one submenu to the other by sliding along the outer border. The ability to explore the set of available items is especially useful to novices who discover the interface.

4. CONCLUSION AND FUTURE WORK

To summarize, the Swiss Army Menu (SAM) is a new menu that allows users of mobile devices to activate many commands without facing the common issues due to direct input on small displays. A prototype can be found at `http://insitu.lri.fr/sam/`. The evaluation of SAM is left for future work.

Future work also includes the refinement of the expert mode. Indeed, unlike Flow Menus [5], SAM supports a real expert mode in the same spirit as marking menus [8]. For now, the prototype implements only a "naive" version that simply consists in hiding the menu graphical appearance and disabling the possibility for the user to reposition his thumb when the puck is in the rest area. Releasing the finger always stops the interaction with SAM.

However this naive version is at the risk of the user's bad perception of the puck location. What are thought of as circular movements can actually be narrow ellipses. The puck can also unintentionally switch between rest and control areas. Strokes might be shorter so that they actually do not touch the rubber band or hit the outer border as the user would expect. The envisioned "clever" version considers a stroke-based input that gets rid of precision in shaping and stroke length by making use of:

- the stroke *orientation* to trigger commands, toggles and submenus (as in the case of marking menus). Therefore the stroke length does not carry information, except for elastic items;
- the *dynamics of the movement*, similar to [1], to detect when the puck enters the control area for circular items;
- the *changes in rubber rigidity*, i.e. the rubber band becomes more rigid when the user is controlling a value with a circular item so as to accept circles that are potentially very degenerated[2]. The rubber rigidity can be simulated by distorting the motor space [3] since the puck is controlled in an indirect way. In case the user wants to cancel the current value he is setting, he just pauses to enter novice mode. The menu thus appears so that he can get back to the rest area as described above.

Acknowledgments: We would like to thank Clément Pillias, Julien Altieri, Pierre Rossinès and Rodrigo Andrade de Almeida for discussions about the menu design, and Michel Beaudouin-Lafon, Olivier Chapuis and Emmanuel Pietriga for their feedback on earlier versions of this paper.

5. REFERENCES

[1] Agar, P., and Novins, K. Polygon recognition in sketch-based interfaces with immediate and continuous feedback. In *Proc. Conf. on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, GRAPHITE '03, ACM (2003), 147–150.

[2] Baudisch, P., and Chu, G. Back-of-device interaction allows creating very small touch devices. In *Proc. Conf. on Human Factors in Computing Systems*, CHI '09, ACM (2009), 1923–1932.

[3] Blanch, R., Guiard, Y., and Beaudouin-Lafon, M. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proc. Conf. on Human Factors in Computing Systems*, CHI '04, ACM (2004), 519–526.

[4] Francone, J., Bailly, G., Lecolinet, E., Mandran, N., and Nigay, L. Wavelet menus on handheld devices: stacking metaphor for novice mode and eyes-free selection for expert mode. In *Proc. Conf. on Advanced Visual Interfaces*, AVI '10, ACM (2010), 173–180.

[5] Guimbretière, F., and Winograd, T. FlowMenu: combining command, text, and data entry. In *Proc. Symp. on User Interface Software and Technology*, UIST '00, ACM (2000), 213–216.

[6] Huot, S., and Lecolinet, E. ArchMenu et ThumbMenu: contrôler son dispositif mobile "sur le pouce". In *Proc. Conf. of the Association Francophone d'Interaction Homme-Machine*, IHM '07 (2007), 107–110.

[7] Karlson, A. K., and Bederson, B. B. One-handed touchscreen input for legacy applications. In *Proc. Conf. on Human Factors in Computing Systems*, CHI '08, ACM (2008), 1399–1408.

[8] Kurtenbach, G., and Buxton, W. Issues in combining marking and direct manipulation techniques. In *Proc. Symp. on User Interface Software and Technology*, UIST '91, ACM (1991), 137–144.

[9] Roudaut, A., Huot, S., and Lecolinet, E. TapTap and MagStick: improving one-handed target acquisition on small touch-screens. In *Proc. Conf. on Advanced Visual Interfaces*, AVI '08, ACM (2008), 146–153.

[10] Smith, G. M., and m.c. schraefel. The radial scroll tool: scrolling support for stylus- or touch-based document navigation. In *Proc. Symposium on User Interface Software and Technology*, UIST '04, ACM (2004), 53–56.

[2]Even in novice mode the default elasticity makes the technique tolerant enough to not constrain the user to draw perfect circles (the user has to be far from at least a distance d from the outer border to cross back to the rest area).