An Accessible and Usable Soft Keyboard

Alexandros Mourouzis[1], Evangelos Boutsakis[1], Stavroula Ntoa[1], Margherita Antona[1], and Constantine Stephanidis[1,2]

[1] Foundation for Research and Technology – Hellas (FORTH), Institute of Computer Science, Heraklion, GR-70013, Greece
{mourouzi,boutsak,stant,antona,cs}@ics.forth.gr
[2] University of Crete, Department of Computer Science

Abstract. *AUK* is a 3x3 multi-tier onscreen keyboard. It supports various entry modes, including 1 to 10-key and joystick modes, allowing text entry with a remarkable range of devices. This paper presents the menu structure of *AUK*, the alternative entry modes, and several layouts for novice, moderate and expert users. The potential of *AUK*, as a text entry solution both for disabled and able-bodied users, is discussed. Overall, the work presented here is considered as a contribution to Universal Access and towards *ambient text entry*.

Keywords: Text entry, soft keyboard, Universal Access, Ambient Intelligence.

1 Introduction

Manual text entry, as the human process of communicating text to computer-based systems, is, in essence, an alternative way of *writing*. The main difference is that, in text entry, the storage medium is a computer memory rather than paper. Naturally, the medium itself poses certain limitations. Computers require users to select among a set of encoded characters[1], whereas paper, as a flat surface, implies the use of 2D symbols. However, the new medium brought along new options that changed writing forever. For instance, it enabled authors to act on transcribed texts without imposing restrictions for modifications, and to copy-and-paste text sections easily.

In these terms, text entry is merely a selection procedure [1], and can be defined as follows (adapted from [2]): *T(i, C, M)* is the human process of selecting from *a set of encoded characters C*[2] and *modifying options M* (backspace, delete, etc.), by means of a total number *i* of discrete and detectable *user actions* (e.g., tap a key).

According to [3], there have been two major waves of research on text entry, focussing on mainstream desktop computers and on mobile devices respectively. However, today a number of reasons call for practical solutions for further widening the potential user base and context of use of text entry systems. Such reasons include the need for inclusion into the information society [4] and the emergence of ambient intelligence environments [5]. This paper presents *AUK*, a multi-device soft keyboard for accessible, usable and efficient text entry across multiple contexts of use.

[1] Graphemes, symbols, etc., or a set of encoded strings of characters, i.e., words or sentences.
[2] E.g., the 95 printable ASCII characters based on English, numbered 32-126.

C. Stephanidis (Ed.): Universal Access in HCI, Part II, HCII 2007, LNCS 4555, pp. 961–970, 2007.

2 Background

Typing, originally introduced by mechanical typewriters, has been a diachronic entry method since the early emergence of computers and command line interfaces. In this method, there is a correspondence between characters and a set of keys spread across a device. Keyboards with different numbers of keys exist, but the 101-keys keyboard is predominant. In these terms, typing can be considered a *T(i, C, M)*, where *i* equals 101+, *C* includes 95+ characters[2], and *M* includes numerous options for modifying text. Such options are mainly of two types: navigation / selection (back, forward, home, select all, etc.) and modifying / processing functions (delete, backspace, copy-paste, etc.). Clearly, with a 101+ keys keyboard, *C* and *M* together exceed the number of input controls (i.e., of keys), thus *synchronous and asynchronous key combinations* (e.g., though the SHIFT and CAPS LOCK key) are employed to allow associating more than one character (or modifying option) to a single key.

With the emergence of graphical interfaces, menu-based text entry was introduced as a supplement of typing, allowing users to enter additional characters[3] not found on the keyboard through menus. In this *T(i, C, M)* case, if navigation across a characters set is made through the keys back, forward, down, up, and enter, *i* equals 5, *C* includes, for example, 256 characters[4], and *M* is an empty set. Such menus can be perceived as a primitive form of *soft* or *onscreen keyboards*. These are features of a program or operating system that generate a graphical keyboard operated through mouse, stylus, etc. Soft keyboards are particularly useful in two cases: (a) for making text entry accessible to people with manual dexterities - e.g., it can be combined with scanning techniques [6] and allow operation through few special switches; (b) for removing keypads from mobile devices and reducing their physical size – e.g., characters can be selected through tapping on a touch display. In both cases, soft keyboards are usually a representation of a full-size QWERTY keyboard.

With the emergence of mobile devices, typing and menu-based methods were combined to allow text entry through few-keys [3]. The success of SMS messaging on mobile phones raised the attention on 12-key keypads, with letters A-Z traditionally encoded on eight keys[5]. A number of 12-key entry methods exist so that more than one letters can be assigned to each key [7], but the most common are *multi-tap* and *dictionary-based disambiguation*[6] (for descriptions see [8, 9]).

Finally, other widely explored entry techniques include *speech*, *handwriting* or *sign language recognition* [1, 3, 9] and *virtual keyboards*[7] [10]. These techniques, up today, have failed to enter mainstream technologies, mainly because they do not reach acceptable levels of *speed* and *accuracy*, two primary *performance metrics* for text entry [3]. For details on text entry issues and methods, see [1, 3, 8, 9, and 11].

---

[3] Over the years, the volume of encoded characters increased dramatically and, ultimately, included characters from numerous writing systems around the world.

[4] From ISO 8859-1 (the most popular encoding in the Western world with 256 characters).

[5] See ITU E.161 international recommendation for telephone keypads at http://www.itu.int.

[6] The most common application is *T9* by Tegic Communications (http://www.t9.com).

[7] Although the term virtual keyboards (VKs) is often used interchangeably with soft keyboards, VKs are defined as touch-typing keyboards that do not have physical manifestation of the sensing areas [10], and thus should be perceived as different from onscreen keyboards.

3 The *AUK* Concept

The work presented here was originally motivated from previous experiences with scanning techniques for users with limited motor functionality of upper limbs. The evaluation of an accessible Web browser [12], which included a QWERTY-like soft keyboard operated via three special switches, indicated frustrating rates of *words per minute* (wpm)[8]. To overcome this barrier, the idea was put forward to use the layouts and current practices in mobile phones and implement a 12-key soft keyboard for offering movement minimisation and high learnability thanks to familiarity. Such a keyboard was anticipated to allow entry with various devices for typing (i.e., keypads), 2D indication and selection (e.g., a mouse), or simple selection (e.g., a single switch) if a scanning technique is built in. Then, having in mind that the letters A-Z on a 12-key keypad are typically encoded on eight keys, an 8-direction method[9] was explored for text entry also by means of a joystick: In that case, if one takes into account the joystick's default state (centre), there are 9 possible positions that can be mapped to a 3x3 grid, which leads us to the basic concept of *AUK*. That is, a multi-tier 3x3 menu system, consisting of a primary menu, where eight cells are used for entering letters and one cell is reserved for entering into additional, again 3x3, menus that accommodate extra characters and options.

Initially, letters were placed alphabetically, similarly to phone keypads (see Fig.1), and the *multi-tap* technique was considered for character disambiguation, employing *system timeout* to segment consecutive letters on the same key. The main benefit of such an approach is clear: familiarity. However, there are also some disadvantages recorded: (a) timeouts slow down users; (b) with an alphabetic layout, some frequent letters require more keystrokes than some less frequently needed letters [1]. Thus, as for people with reduced manual dexterity movement minimisation is a primary requirement, less *keystrokes per character* (KSPC [8]) were further pursued.

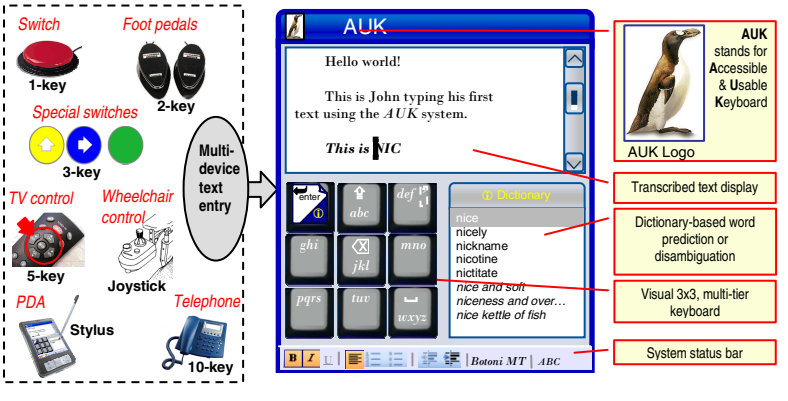

Fig. 1. Overview of *AUK*[10] soft keyboard for multi-device text entry (*stand-alone version*)

[8] The main reason is that the QWERTY layout has been designed for ten-fingers typing and not for linear or tree-based navigation as in scanning techniques.

[9] With one direction N, E, S, W, NE, ES, SW, WN used for each of the eight keys.

[10] About the *AUK* logo: According to wikipedia.com, auks are birds that look like, but aren't penguins; they are considered as a product of "*moderate convergent evolution*".

In summary, *AUK* supports (a) several different entry modes (1 to10-key, and joystick), rendering it appropriate for the widest possible range of input devices; (b) various layout configurations for different performance levels; and (c) the potential to integrate additional techniques, such as *text prediction* and *dictionary-based* or *prefix-based disambiguation*, as these ensure even lower KSPC values [3, 8, 10].

4 A Multi-tier 3x3 Menu Structure

AUK is similar to a multi-tier 3x3 menu system. Each 3x3 grid has eight virtual *character keys* and a *menu key* for entering or exiting alternative menus. Five *basic menus* can be interchanged sequentially[11] (Fig. 2). In the *primary menu* (1), the letters A-Z, the apostrophe, SPACE (SP), BACK (BK), and SHIFT (SH)[12] are placed on the character keys, and the menu key is used for *timeout kill*, swapping to menu (2), or entering a *linear menu* (D) for navigations, e.g., within a *word prediction list*[13]. Additionally, there are five *secondary menus*: *numerals* (6), *special characters* (7), *brackets* (8), *formatting options* (9), and *numeric operators* (10). Finally, menu (5) has a cell for selecting extra characters through menu (D), and an empty cell to allow space for adding new menus as necessary, thus rendering the structure extensible.

5 Entry Modes and Menu Layouts

The presented menu structure can be employed with various input methods. In particular, given *T(i, C, M)*, *AUK* supports text entry for:

- ***i*=9**. This is the basic mode of *AUK*, in which there is a correspondence between the nine *AUK* menu options (see Fig. 2) and nine keys (e.g., phone keypads). This mode has two main variations: (a) the *0-key mode* for *command without click* [15] e.g., by means of eye-tracking; and (b) the *joystick mode*.
- **i<9**. There are also several modes for products and assistive devices that have less than nine 'keys': *1-key*[14] (single switch or key, etc.); *2-key* (e.g., foot pedals); *3-key*[15] (three switches, rotating wheel, etc.); *4-key*; and *5-key* (e.g., pager keypad). Basically, in this category, a *menu cursor* travels, automatically or manually, across each 3x3 menu, always starting from a *home position*. Once the desired input element is reached, a *select button* is used for "tapping" the onscreen key.
- **i>9**. In this category focus is mainly on 10-key modes for number pads or 10-thumbs typing (see section 8). In the first case, the tenth key (*zero key*) is used for

[11] E.g., in the *9-key mode,* starting from the menu (1), the key combination 1-1-1-1-1 will result returning to menu (1), after visiting the *text navigation menu* (2), the *punctuation menu* (3), the *mode menu* (4), and the *extend menu* (5). If the user selects any *character key* in one of these menus, then the menu key changes to the *exit key* for returning directly to (1).

[12] SP, BK, and SH are very common [14]; in *AUK*, SP, being the most common key, is always the most easy to reach; the apostrophe, indicates missing letters, and thus is included here.

[13] If *dictionary-based* disambiguation is activated, then a list of alternative words is displayed.

[14] An automatic scanning technique is employed in this case.

[15] Two methods are used for the *3-key mode*: the *date stamp method* [see 985], and one proposed for *AUK*, through the buttons DOWN, RIGHT and SELECT, as this provides less KSPC.

timeout kill (single tap) and for the SH modifier (hold down). A second approach, mainly for 10-thumbs-typing, is to place the SP key alone, and use the tenth key for timeout kill (first tap) and BK (additional taps). Furthermore, in this second case, the space key may also be used for the CTRL modifier (hold down) and the tenth key for SH, whereas SH+SP is for entering menu (2) - see Fig. 4.

Indicatively, the arrangement of keys for the 3-, 9-, 10-key and joystick modes are presented in Fig. 3, along with the indicative entry scenarios presented in Table 1.

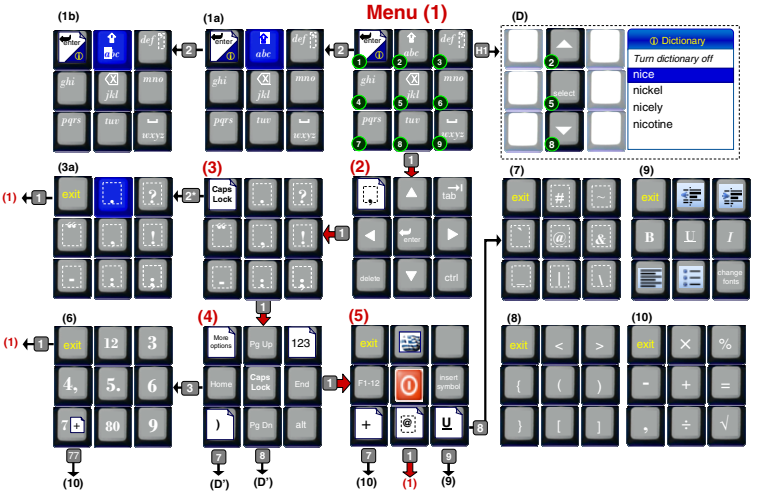

Fig. 2. The basic menu structure of *AUK* (for *i*=9)

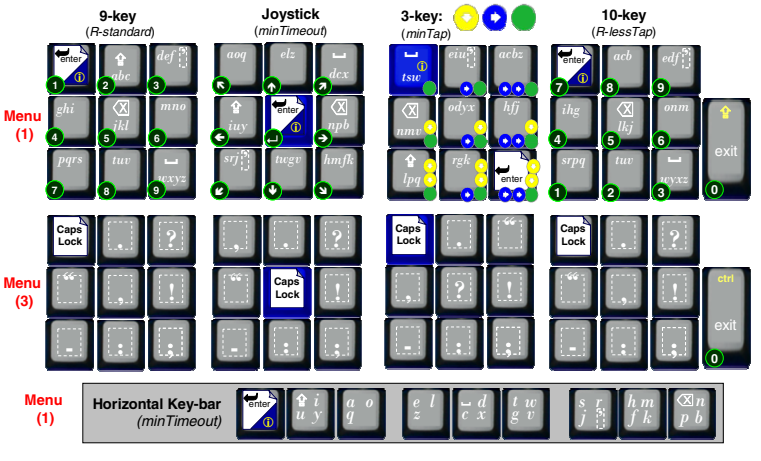

Fig. 3. Keys arrangement for modes 3-, 9-, 10-key and joystick. The bottom part shows the AUK menu as a horizontal bar for PDAs (see section 6).

6 Layouts for Novice, Moderate and Expert Performance

A significant amount of research has been conducted on 12-key text entry methods for mobile phones. Most efforts aim at achieving the upper limits of typing speed with a particular design, often at the expenses of an acceptable experience for novice users [3]. Yet, the ease of use for first-time users is critical for the success of any method. For this purpose, in addition to the entry mode described above, different A-Z letters layouts have been designed for novice, moderate and expert users. The following are layouts for multi-tap and involve the use of system timeout.

Table 1. Examples of user input, resulting transcribed text and menu transitions

User Input (with layouts in Fig. 3)				**Menus onscreen**‡	**Text** (transcribed)
9-key (*R-stan*)	Joystick (*minTout*)	3-key (*minTap*)	10-key (R-Lesstap)		
1	↗	↵	3	(1)	␣_
2,3	↤←,↑	↓,↓,↵,→,↵	0+9	(1)	E_
3,3,4,4,4	↑,←	→,↵,→,↵,↵	9,4	(1)	ei_
2,2,1,2,2,2	↖,↗,↗,↗	→,→,↵,→,→,↵,↵	8,0,8,8	(1)	ac_
3,3,8	↑,↓	→,↵T, ↵,↵	9,2	(1)	et
3,3,9	↑,↗	→,↵T, ↵	9,3	(1)	e␣_
3,3,1,1,1,2	↑,↵,↵,↵	→,↵↓,↓,→,→,↵,↵,→,↵	9,7,7,2	(1>2>3>1)	e._
1+5	H↵,↵	H↵,→	1+5	(1>D>1)	*1st word*␣_
1,1,9,7,7,7,7	↵,↵,↘,↙	↓,↓,→,→,↵,→,↵,↵,↵,↵	7,0+1	(1>2>1)	(ctrl+s)

T: system timeout _: text cursor's position ␣: SPACE H↵ Hold down key until system timeout

R-standard. This layout is the *standard phone layout*[5] adapted to include SP, BK, SH and apostrophe to the character keys with the lowest overall frequency (see the example layout for *9-key* in Fig. 3). This is a familiar and easy to remember layout.

R-lessTap. This layout is an adaptation of the *LessTap* [13], in which letters within each key of the standard layout are rearranged according to their frequency in the English Language[16] and thus achieving lower KSPC. *R-LessTap* simply includes the four extra options mentioned above (see the layout for *10-key* in Fig. 3).

Taking into account that several studies [e.g., 16] have indicated that an alphabetic order is not critical for user experience, more effective layouts can be created by disregarding the alphabetical order all together, as described below.

minTap. In this layout, the more common an input is, the easier it is to enter it. This is achieved through the following algorithm: (a) *inputs* are sorted by *frequency* order[17]; (b) all menu *positions* are coded, e.g., from *1-001* (representing the 1st position, on key 1, in menu *1*) up to *n-900* (representing the third position, on key 9, in menu *n*), and sorted by their *reachability* order (e.g., the *number of keystrokes* (KS) required to select each position); (c) the most frequent input (i.e., SP) is assigned to the position with the best reachability, and so on; (d) if an option can be assigned to

[16] E.g., see http://en.wikipedia.org/wiki/Letter_frequencies (used here for KSPC calculations).
[17] Frequency order used here: SP BS eta SH oinshrdlcumwfgypbv apostrophe kjxqz.

more than one positions that share equal reachability values, then it is placed on the key that currently has the lowest overall frequency[18]. An example appears in Fig. 3.

minTimout. In this layout, (a) letters on each key have the lowest possible *digraph frequencies* [11] in an *English text corpus* in order to have consecutive letters on the same key (and thus timeouts) less frequently[19], and (b) the letters on each key are arranged in decreasing *letter frequency* order – thus ensuring less 'taps' for frequent letters (see the layout used for the *joystick mode* in Fig. 3).

R-qwerty. In this layout, characters are grouped and arranged in a way that is related to the QWERTY layout. Such relation can be vision-based (e.g., having the letters "qwer" grouped together on the same key) or motor-based (i.e., with keys that are assigned to a specific finger[20] being grouped together (see Fig. 4).

In modes with i<9, there are three potential versions to be explored: (a) *persistent*, in which the virtual cursor snaps to home position after each character entry; (b) *static*, in which the cursor does not change position after each character entry, and (c) *dynamic*, in which the goal is to minimise the cursor-key distance to the next character (e.g., by moving the cursor on the most probable key, or by automatically rearranging entirely the characters layout). The latter, can also be applied in modes i>=9 for automatically rearranging the letters within each key, so that the first letter is always the most probable. In addition, *AUK* may have the following general modes: (a) multi-tap; (b) dictionary based disambiguation; (c) prefix based disambiguation [18]; (d) with or without *text prediction* [19] displayed though the *menu* (D); (e) synthetic speech output enabled for blind users. Finally, the following display options have been considered: (a) 3x3 keypad; (b) vertical or horizontal bar (e.g., for PDAs – see Fig. 3); and (c) cursor view (e.g., see Fig. 4).

7 Discussion: *Universal Access* to Text Entry

The proposed 3x3 structure shows a number of benefits. First of all, it complies with the fact that visual search involves "*combination of gestalting the area surrounding the current center of interest –3x3 matrix of letters– and using linear search … as a back up strategy*" [17]. Yet, this structure offers access to a large and extensible input set, and thus *AUK* can operate as a stand-alone word processor or even as a plug-in for other applications. Furthermore, it is potentially easy to learn and use for users with different expertise, since: (a) functions can be placed as close to the user expectations, e.g., for improving novice experience [9], or according to their (individual or related) probabilities for expert performance; (b) onscreen visual[21] and aural feedback is provided for character entry and menu transitions; and (c) the backspace key is always the second most reachable option, thus allowing fast *error recovery* [20]. In addition, since $i <= 10$, the input device operation can be easily blind, thus rendering *AUK* a *two-FOA* or even a *single-FOA* method[22] [3].

---

[18] So that frequent options are grouped with the less need ones without dispensing reachability.
[19] According to [1], the most frequent digrams account for over 80% of text in a language.
[20] In expert QWERTY typing each finger is assigned to specific keys.
[21] E.g., selected keys are highlighted (e.g., see (1a) and (1b)).
[22] The *focus of attention* (FOA) can single if AUK is visualised as cursor (see Fig. 4).

Table 2. Indicative performance rates of *AUK*, in comparison to other text entry methods, using a specific sentence that includes very common words in English Language[23]

Transcribed sententece: "As you have all said to her, she was not the one for him."									
(57 Chars; 14 words (from the 50 most frequent words in English); average word length 3; correlation with English: 0.8921)									
		T(i, C, M)		User actions		System actions		ARPI	
System (Char. layout)	**Method**	i	#C	Total	Mean	Total	Mean	Total	Mean
Desktop (QWERTY)	1- tap	101+	95+	57	1,000	0	0,000	57	1,000
Mobile keypad (standard)	T9	10	72+	60	1,053	0	0,000	60	1,053
AUK, joystick (minTap)	**directional**	**10**	**95+**	**79**	**1,386**	**2**	**0,035**	**81**	**1,421**
AUK, 10-key (R-lessTap)	**multi-tap**	**10**	**95+**	**79**	**1,386**	**4**	**0,070**	**83**	**1,456**
Mobile keypad (standard)	multi-tap	10	72+	116	2,035	4	0,070	120	2,105
AUK, 9-key (R-standard)	**multi-tap**	**9**	**95+**	**125**	**2,193**	**4**	**0,070**	**129**	**2,263**
FORTH Editor, 3-key [12]	manual scanning	3	95+	792	13,895	0	0,000	792	13,895
AUK, 3-key (minTap)	**manual scanning**	**3**	**95+**	**168**	**2,947**	**2**	**0,035**	**170**	**2,982**
Click-N-type	auto-scanning	1	95+	174	3,053	478	8,386	652	11,439
Microsoft On-screen keyboard	auto-scanning	1	95+	232	4,070	564	9,895	796	13,965
FORTH Editor, 1-key [12]	auto-scanning	1	95+	135	2,368	760	13,333	895	15,702
AUK, 1-key (minTap)	**auto-scanning**	**1**	**95+**	**174**	**3,053**	**118**	**2,070**	**292**	**5,123**
#C: length of Character set									

Regarding the potential user performance with various entry modes of the *AUK* system, the total number of *actions required per input* (ARPI[24] - i.e., *user actions*, such as taps, keystrokes or joystick movements, and *system actions*, such as timeouts or scanning intervals) was calculated in typing several sentences and compared to that of other techniques. These early results were very encouraging. In order to allow quick comparisons of *AUK* with other techniques, the average ARPI for typing a simple sentence that contains very frequent English words are presented in Table 2.

On the other hand, the fact that *AUK* supports numerous entry methods for *i* that ranges from 1 to 10+ renders the proposed text entry technique accessible from almost any device, including mainstream mobile devices and assistive technologies for people with disability (e.g., special switches and screen readers for blind users).

Under the light of the above, *AUK* is considered as a method that can potentially provide **Universal Access [4] to text entry** for anyone, regardless of the device used, the expertise of the user, or any other contextual parameter.

Currently, an interactive prototype of *AUK* has been implemented simulating the joystick and the 1-, 3-, 5-, 9- and 10-key modes through a desktop keyboard, and early tests are currently in progress for fine-trimming the *interface* of *AUK* and all layouts. Regarding near future plans: (a) the ARPI for each *AUK* entry mode and proposed character layout will be calculated using public corpora, and detailed comparison results with other techniques will be presented; (b) user performance will be estimated and predicted using theory-based methods such as Hick and Human's method and Fitt's Law; (c) user trials will be conducted with the interactive prototype of *AUK*, e.g., using the public phrase sets of MacKenzie and Sourkoreff[25]; (d) a version for Greek language will be implemented; (e) a mechanism for recording and employing personalised user corpora[26] will be developed and integrated; (f) plug-in versions of *AUK* will be developed for various platforms and applications; and (g) research will

[23] The correlation to English language was alculated with *AnalysePhrases* tool available at: http://www.yorku.ca/mack/PhraseSets.zip (Author: I. S. MacKenzie, 2001).

[24] This term is used in analogy to the term *keystrokes per character* (KSPC) introduced in [8].

[25] Available at http://www.yorku.ca/mack/PhraseSets.zip.

[26] Performance rates with *minTap* and *minTimeout* may be further improved by using individual user corpora for automatically rearranging the menu layouts accordingly.

be conducted to identify the frequency of all input options and the *dioption* frequencies (term used in analogy to digraph frequency) in desktop text entry in order to optimise the tree structure of the *AUK* menus, especially for 9-, and 10-key modes.

8 Towards Ambient Text Entry

Since the *AUK* approach is consistent and applicable across various (small) devices, it is anticipated to find application, and provide Universal Access, in the context of Ambient Intelligence environments [5]. In such an environment, the need will arise to provide ubiquitous and efficient text entry to displays of various sizes and characteristics, and in many cases without a traditional keyboard.

Additionally, *AUK* offers the possibility to achieve *ambient text entry*. If, for example, optical tracking through a portable camera is employed to track the movement of finger of a user, then *AUK* can support the development of a ubiquitous *virtual keyboard,* not necessarily visible, for 10-finger text entry. Such a technique could be based on the mapping of each finger to one of the virtual keys of AUK in the 10-key mode. For instance, assuming the mapping presented in Fig. 4 (right), if the user makes a discrete move of the small finger of the left hand, then the character 'a' will be entered. Alternatively, for visual feedback purposes, the 3x3 *AUK* menus can be presented in a cursor-like form within the text (see Fig. 4, left).

Overall, this approach has the potential to introduce significant advantages**. Text entry becomes *target-less* and *eyes-free* for all**, i.e., the user does not have to search for, and achieve physical contact with, a specific area (e.g., a key), thus **eliminating performance delays** due to time used (a) to visual scan in target identification (Hick and Human's method), and (b) to target selection (Fitt's Law). Furthermore, **finger and hand movement is minimised**, since there is no need to travel across a keyboard to reach different keys. On the other hand, using the cursor-like version of *AUK*, extra time is saved as the user does not have to focus from the text to the keyboard and back. Taking this a step further, a virtual mouse can be integrated. For instance, discrete finger moves can be reserved for typing and continuous ones for moving the mouse cursor, and inferring minimal hand travelling from the "keyboard" to the "mouse". Finally and overall, **this approach is flexible and potentially adaptable to any user ability**, since it also works consistently and effectively for cases where i<10.

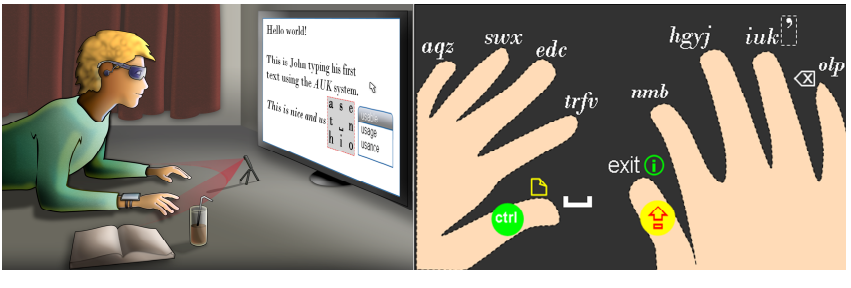

Fig. 4. Towards *ambient text entry for all* (sketch courtesy of Anthony Katzourakis)

Acknowledgments. Part of the work reported here has been carried out in the framework of the integrated project *ASK-IT* (Contract no 511298) funded by the European Commission.

References

1. Isokoski, P.: Manual Text Input: Experiments, Models, and Systems (PhD Thesis). University of Tampere, Tampere Finland (2004)
2. Sandnes, F.E., Thorkildssen, H.W., Arvei, A., Buverud, J.O.: Techniques for fast and easy mobile text-entry with three-keys. In: Proc. of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04), pp. 1–10 (January 05-08, 2004)
3. MacKenzie, I.S., Soukoreff, R.W.: Text Entry for Mobile Computing: Models and Methods, Theory and Practice. Human-Computer interaction 17, 147–198 (2002)
4. Stephanidis, C., et al.: Toward an Information Society for All:HCI challenges and R&D recommendations. International Journal of Human-Computer Interaction 11(1), 1–28 (1998)
5. Emiliani, P.L., Stephanidis, C.: Universal access to ambient intelligence environments: Opportunities and challenges for people with disabilities. IBM Systems Journal 44(3), 605–619 (2005)
6. Edwards, A.D.N.: Extra-Ordinary Human-Computer Interaction: Interfaces for Users with Disabilities. Cambridge University Press, Cambridge (1995) ISBN 0-521-43413-0
7. Oniszczak, A., MacKenzie, I.S.: A comparison of Two Input Methods for Keypads on Mobile Devices. In: Proc. of NordiCHI'04, Tampere Finland, pp. 101–104 (October 23-27, 2004)
8. MacKenzie, I.S.: KSPC (Keystrokes per Character) as a Characteristic of Text Entry Techniques. In: Proceedings of the Fourth International Symposium on Human Computer Interaction with Mobile Devices, pp. 195–210. Springer, Heidelberg (2002)
9. Johansen, A.S., Hansen, J.P.: Augmentative and alternative communication: the future of text on the move. International Journal of UAIS 5, 125–149 (2006)
10. Kölsch, M., Turk, M.: Keyboards without Keyboards: A Survey of Virtual Keyboards. In: Sensing and Input for Media-centric Systems, SIMS 02 (2002)
11. Zhai, S., Hunter, M., Smith, B.A.: Performance Optimization of Virtual Keyboards. Human-Computer Interaction 17, 89–129 (2002)
12. Mourouzis, A., Ntoa, S., Boutsakis, E., Kartakis, G., Stephanidis, C.: Expert-based assessment of the ARGO Web browser for people with disability. In: Proc. of 8th European Conference for the Advancement of Assistive Technology in Europe (AAATE 2005), September 6-9, pp. 767–771. Lille, France (2005)
13. Pavlovych, A., Stuerzlinger, W.: Less-Tap: A Fast and Easy-to-learn Text Input Technique for Phones. In: Proc. of Graphics Interfaces, pp. 319–326 (2003)
14. Soukoreff, R.W., MacKenzie, I.S.: Input-based language modelling in the design of high performance text input techniques. In: Proc.of Graphifcs Interface, pp. 89–96 (2003)
15. Hansen, J.P., Johansen, A.S., Hansen, D.W., Itoh, K., Mashino, S.: Command Without a Click: Dwell Time Typing by Mouse and Gaze Selections. In: Proc. of INTERACT'03, pp. 121–128 (2003)
16. Norman, D.A., Fisher, D.: Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. Human Factors 24(5), 509–519 (1982)
17. Bellman, T., MacKenzie, I.S.: A probabilistic character layout strategy for mobile text entry. In: Proc. Graphics Interface '98, pp. 168–176 (1998)
18. MacKenzie, I.S., Kober, H., Smith, D., Jones, T., Skepner, E.: LetterWise: Prefix-based disambiguation for mobile text input. In: Proc.of the ACM Symposium on User Interface Software and Technology (UIST 2001), pp. 111–120 (2001)
19. Garay-Vitoria, N., Abascal, J.: Text Prediction Systems: A survey. International Journal on Universal Access in the Information Society (UAIS) 4(3), 188–203 (2006)
20. Soukoreff, R.W., MacKenzie, I.S.: Metrics for text entry research: An evaluation of MSD and KSPC, and a new unified error metric. In: Proc.of CHI 2003, pp. 113–120 (2003)