Typing Quickly and Relaxed with the Eyes

A case study comparing switch based and Gaze Controlled Input Methods

Dipl.-Inform. Joern U. Garbe

jgarbe@acm.org

Abstract

This paper presents the communication aid software ERIC (Efficient Reduced Input Communication) which is in use for more than 5 years by a woman suffering from cerebral palsy. The system has been used with up to four switches. Very recently the software has been extended to provide gaze controlled access. The paper will present the ERIC software and the special requirements that had to be fulfilled to make it more gaze friendly. During a number of trials the two input methods have been compared with respect to two criteria: speed and convenience. The results of the trails will be presented.

Introduction

Physically handicapped people with severe motor impairments cannot use conventional computer input devices. However, computers provide unparalleled new ways of augmentative and alternative communication (AAC). Therefore, customized input programs have to fill the gap and provide users with motor impairments an easy means to enter text or even control arbitrary computer applications (including environment control applications).

At the age of fifteen Kathrin Lemler from Koblenz, Germany asked the Computer Science department at the University of Koblenz if the scientists would be interested in improving her communication device. Kathrin was suffering from cerebral palsy and was forced to use a wheelchair. For communication she was using a variety of communications means. She used gestures and basic sounds, spelled words with eye-script and a (virtual) letter grid and produced texts with a symbol based approach (using Minspeak). All approaches had specific advantages and disadvantages. The fasted way to communicate was using the letter grid. However, this was only possible with people who were trained in decoding her eye movements back to the words she encoded. The number of distinct sounds she could produce and hence the vocabulary of the sounds she could produce was very limited. Using her AAC device she could talk to strangers. However, also this means of communication suffered from several drawbacks: Of course she always had to have the device with her, the vocabulary was still limited to about 3000 word forms, she had to memorize rather cumbersome symbol combinations for words, and most importantly using a single switch writing sentences was prohibitively slow. Especially the expressive limitations were a major concern when Kathrin was about to change to a regular high school.

When Kathrin approached the university the first idea was to extend the symbol based approach to allow for an easier integration of new words. Soon it became clear that since Kathrin was able to spell words another approach which uses conventional word encoding with letters would be preferable.

With the help of a computer linguist a prototypical system has been developed [Meyreis 00]. The system used 8 virtual keys. The 30 letters of the German alphabet where distributed on 6 keys, two other keys were used to delete a letter or accept the word. The user could select the keys with a scanning approach. Hence, each key is focused cyclically and the user can select the key while it is focused using a single switch.

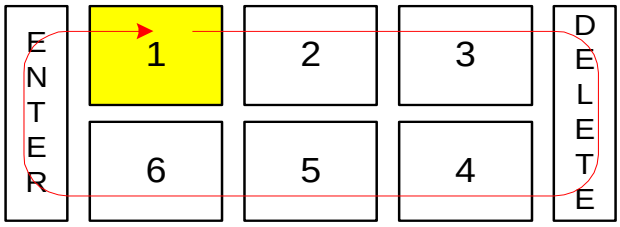

Figure 1: UKO program developed at the University of Koblenz

As the number of available keys is smaller than the number of characters that should be available several characters have to be placed on each key. There are different strategies to resolve the ambiguity involved in this approach. The purest forms are character-level disambiguation and word-level disambiguation. However, there is also the possibility to combine both methods.

The strategy that is used in the ERIC system is based on word-level disambiguation, i.e. the user writes a word letter per letter ignoring the ambiguity of the keys. Once a word is finished all words that match the input sequence are presented ordered by probability.

The time required to enter a word is the sum of the time needed to select all letters and press ENTER and the time needed to select the word from the list of all matching words.

The typing speed that can be accomplished with this approach crucially depends on the distribution of the characters on the keys. In a diploma thesis Meyreis developed a distribution using a set of linguistic hypothesis. Many criteria and rules can guide the way towards an efficient character distribution. However, many of those rules are contradictory and hard to apply. Therefore, solving the problem by hand is a thankless job. For instance “letters that are likely to appear in sequence should appear in the same sequence – or at best on the same key (if multi-clicking is permitted) – on the keyboard” seems to be a perfectly reasonable rule (to increase typing speed). However, consequent application of this rule will ultimately yield a solution where each letter is placed on the same key. Then, of course, all words of same length will match the same input sequence and selection of the right word will take an enormous amount of time. Therefore, another rule that minimizes ambiguity of input sequences – such as “letters that are likely to appear in similar contexts should not be placed on the same key” – has to be introduced. But which rule is more important in which context is not at all clear.

A further drawback of using linguistic rules is that the rules are language specific. Therefore, the manual optimization has to be done for all languages that should be available.

Fortunately, the ultimate performance measure – which is how fast a typical text in a given language could be typed – can be expressed by a simple function. The only requirement for this is a dictionary containing the most relevant words with their respective frequency of use. For the performance measure the time required to type each word is calculated and multiplied by the frequency of use. The sum of these values over all words is reciprocal to the distribution's efficiency.

Since there are almost 221 sextillion (10^{21}) reasonable distributions of 30 letters on 6 keys the problem cannot be attacked by a brute force approach. A complete search would take more than 7 trillion years even if a solution could be evaluated in a single millisecond (which is not possible for large dictionaries).

In order to validate or improve the results from the linguistic rule based approach the problem was applied to a so-called Genetic Algorithm [Garbe 00].

Optimization Problem

Given an ordered alphabet = $a_0,\ldots,a_{n-1}$, a language L *, a subset K L of known words, and a frequency function *Frq*:K IN assigning the frequency of use to each word in K, which partition P on allows for the fastest way to enter each word K exactly Frq( ) times?

The time needed to enter a given word is provided by some objective function Time(P): * IN.

$$\text{Time}\ l_1..l_m : \text{Distance}\ 0, \text{Key}\ l_1 \\ \sum_{i=2}^{m} \text{Distance}\ \text{Key}\ l_{i-1}, \text{Key}\ l_i \\ \text{Distance}\ \text{Key}\ l_m, 0 \quad \text{Pos}\ w$$

Key(l) returns the index of the key on which letter l is placed – of course, this value is determined by the partition P. Distance(i, j) calculates the cycle distance between the keys with index i and j. If the input sequence for a given word is ambiguous Pos returns its position in the frequency-sorted list of matching words, and 0 otherwise.

Optimization Results

The results of the computer linguist are used as a performance measure. With the help of the optimization method it has been possible to increase the typing speed by 17%. For the comparison the same corpus was used, which the expert used as the basis for word prediction. As it was not at all clear if 6 keys would really be optimal the number was reduced down to five, four and three. Interestingly when using a scanning approach three turned out to be the optimum. The best solution found increased the typing speed by 32%. At first sight the linguist's solution seems to be devastatingly bad. The optimization algorithm comes up with a better solution within a few seconds. Therefore, it is tempting to discredit the expert's work or even question his competency. However, studying the different results carefully reveals the shortcomings of the method applied by the expert. His study was based on letter and bi-gram probability and tried to optimize both the number of matches and typing speed. Humans are tempted to neglect typing speed in favor of the optimization of matches. The rather small number of at most 13 matches (for the most ambiguous input

sequence) supports this theory. Selection costs, however, are tiny in comparison to typing costs and account for only 3% to 8% of the total costs of reasonable solutions. We also have to keep in mind that the fitness function (which required an explicit programming effort) was not accessible to the domain expert. Therefore, the expert could not possibly evaluate a large number of different partitions. An inherent advantage of the automatic optimization approach is that the problem can be reformulated (use another dictionary or language, change the number of keys) easily. The expert was not even aware (neither were we before we tried it) that 3 keys might be sufficient.

Eric

Kathrin started to work with a prototypical system developed at the University of Koblenz in 2000. Since the results of the diploma thesis suggested significant speed improvements the ERIC system has been developed. ERIC was built from the ground up as a new system allowing for a flexible number of keys and different modes for one-switch or multi-switch operation. The confirmation (Enter) and correction (Delete) functions were integrated into a single menu on the first button. With the requirement of only four input keys Kathrin managed to access the keys directly with switches mounted to her wheelchair. Being able to select the keys directly brought further speed improvements and the distribution has been recalculated for highest performance with direct access. ERIC has been designed to allow for a maximum of performance when entering text. It has not been designed as a virtual keyboard providing access to arbitrary applications.

Figure 1 shows the main window of ERIC. It consists of 6 elements: the completion list on the left side, a prediction bar on the top, a menu control and 3 letter controls. To write a word the user simply types the word letter per letter as on a normal keyboard. A letter is typed by selecting the control which is marked with the letter. The only difference to a normal keyboard is that each control is marked with several letters. However, this fact should be ignored at first. If the words “Hello world” should be typed we select the third letter control, because it is captioned with H. The window now looks like in Figure 2. For the E we select the third letter control again. The same goes for the L as it is also on the third letter control. Finally for the O we have to select the first letter control. Now the word is completed and we would have to type a space. To do this we have to select the menu control. Now the letter controls provide access to menu items. The function we want to use now is “Select”. As the name implies it is used to select the word we want to write. Because the input sequence 3-3-3-3-1 for hello is ambiguous (it can be used for “hello” just as well as for “hills”) we now have to select the word we actually had in mind. This can be done either in a sequential mode or with a smart narrowing function. There is also a shortcut to words that appear in the completion list (we could have selected “hello” just after typing the h). Now that “hello” is completed we can go on with “world”. We have to type the input sequence 1-1-2-3-2 (W and O are on the first control, R is on the second, L is on the third and D is on the second), go to select and select “world”.

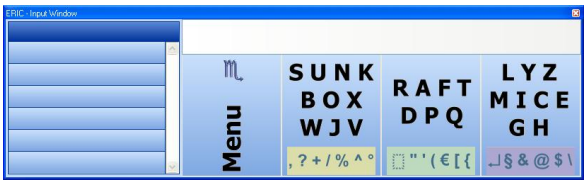

Figure 2: ERIC’s main window

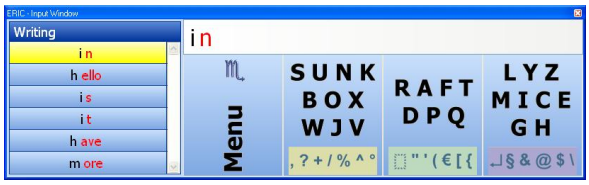

Figure 3: Main window after selecting the third letter control

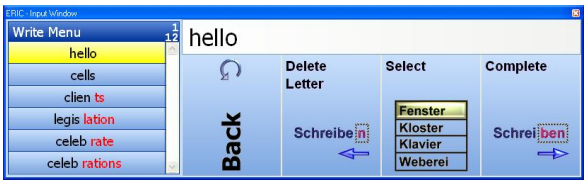

Figure 4: Main window after typing the sequence 3-3-3-3-1 and selecting the menu control

ERIC offers a lot of more advanced functions, including functions to enter new words, use text templates, speak texts with a text-to-speech system and revise texts.

Direct Access Using a Gaze Tracker

As Kathrin has communicated with her mother and later on with her personal assistant using a letter encoding scheme that relies on eye movements, the idea of using a gaze tracking system to control her communication aid was the obvious next step. In contrast to conventional virtual keyboards ERIC requires only very few control elements. This is not only very helpful to get familiar with a gaze tracker, but can also make gaze controlled applications accessible to people whose gaze accuracy would otherwise not be high enough.

In 2005 Kathrin tried different gaze tracking systems for the first time. As Kathrin's performance with one of the systems looked very promising ERIC was changed for the special requirements of gaze controlled applications.

Requirements for Gaze Friendliness

Gaze control can be used to augment other forms of human machine interaction or it can be used as the one and only means to interact with a computer. When used by people with severe motor impairment no or only very few other means will be available. The user might possibly control a limited number of switches, but will typically not have access to conventional input devices.

The manipulation options within applications can be brought down to three major operations: pointing, clicking and text input. Gaze tracking will natively only provide the pointing operation. Clicking has to be provided by an external switch, blinking or dwell selection. Since intentional and unintentional blinking is hard to distinguish dwell selection or external triggers are often preferred (cf. [Lankford 00]). If no other input means is available text also has to be entered using point and click operations.

In order to be efficiently controllable with the eyes, eye friendly applications have to pay special attention to the user interface. For obvious reasons selection accuracy is improved using a reduced number of large controls. The controls should be clearly marked without distracting the user. In addition the design of the controls should facilitate focusing the center. Humans tend to pay special attention to moving or changing items. From this simple observation we can conclude that 1) the pointer should mark the presumably selected control's center but not the assumed position on the control and 2) dynamic items should only be used when user attraction is intended. The first conclusion prevents a problem commonly seen when the assumed gaze position is shown: if there is a small offset between the assumed and the actual gaze position the user is attracted to look on the pointer. This, however, will move the pointer further in the undesired direction. This may result in pointer chasing. Of course, there might be situation in which the actual gaze position on the control is of interest (e.g. sliders or lists). In such situations it should be analyzed if the controls can be substituted by other controls. If the application requires exact positions on controls, zooming techniques (cf. [Bates 02]) will improve usability. As the users of the application will have individual preferences both, the position and the size of the controls should be customizable.

Specific Changes to ERIC

As ERIC has specifically been designed to help motor-impaired people to enter text efficiently it always had to cope with a quite limited number of input controls. With very few controls the required changes to make it gaze friendly were limited to changes of the user interface. The rather small and fixed-size control elements have been made sizable. The prediction list was separated from the main window in order to arrange the entries according to the user's preferences. When used with the Tobii gaze tracking system the software sends information about the position and size of the available controls to Tobii. The gaze tracker will only allow the user to select these controls and will guide the user towards the center using a visual cue.

Switch Based vs. Gaze Controlled Input

Kathrin has been working with ERIC using switch based input for more than 5 years. Changing to a different input paradigm can only be encouraged if there are substantial benefits. The most important performance criteria from the user's point of view are 1) the speed being achievable with the system and 2) the effort being required to type text. The second criterion also includes the fatigue involved when using the system for a longer time. The relative importance of the criteria depends very much on the user. However, since AAC systems are sometimes the only unconstrained means of communication for people with severe motor impairments the achievable speed is always an issue.

In normal oral conversation a rate of well over 150 words per minute can be reached without any difficulty. Also typists can reach similar speeds on longer texts. Even radio operators skilled in Morse code can often consistently produce more than 20 words per minute. In contrast the speed achievable with AAC devices using head pointers, gaze trackers or up to four switches is only a few words per minute.

Speed is an important factor for several reasons. In competitive situations such as exams in school, the pure quantity of information transmittable can become an issue. Also the time required to answer to questions is of utmost importance. Below a certain rate of words per minute keeping up with the conversation becomes

an ordeal. This is especially true for people who are not regularly confronted to slow talkers. However, even people used to the rather slow flow of information tend to start parallel conversations if other people are around. This often leads to broken conversation threads and frustrations on the side of AAC users.

The effort required to type or speak with an AAC device will always be higher than the effort required for touch typing. Keeping the user comfortable with the system and reducing fatigue, of course, also has an important impact on the speed which is achievable in the long run.

Experiment Setup

In order to find out if Kathrin can profit from gaze controlled input an experiment was conducted. In the experiment she had to type several short texts. The texts were selected randomly from yearly awarded speeches, Wikipedia articles and poems. The texts have been selected to simulate situations in which Kathrin has to talk about a topic which is new to her. In particular the texts have not been selected to include as few words as possible not included in the dictionary. We test 6 texts ranging from 301 to 1649 characters. The total number of characters of the corpus is 5032 characters. In order to get familiar with the test method she was presented with a test text which was not included in the analysis. All texts have been printed out and she was given as much time to read the texts before typing as she wanted to have.

In order to assure comparability and instantly reject misspellings the texts have been presented by a computer program showing the current paragraph with the current word marked (teleprompter). In a large font the speed measuring application stated the word the user has to type and the previous and the next word. The program only advances to the next word if the word is spelled correctly. This approach avoids the problem of handling misspellings as it simply does not accept them. This is superior to penalty factors, as with penalties for misspelling leaving out words might still be attractive for the user. Only the zero-misspelling policy assures that the penalty exactly corresponds to the time required to correct the misspelling. The texts have been written with both input methods: eye gaze and direct switch access. Three texts were written starting with gaze control and another three starting with switches.

For gaze input a dwell time of 0.5 seconds has been used. The virtual keyboard consisted of six letter controls, one menu control and six additional prediction controls.

For switch based input only three letter controls and one menu control could be used as Kathrin cannot efficiently use more than 4 switches.

Apart from the screen layout, the number of available controls (and of course the letter distribution) the software setup is identical.

Speed Comparison

The following table shows the results of the different texts typed with both input methods.

Text	Characters	Words	Average time per character [ms]		Median time per character [ms]	
			Gaze	Switch	Gaze	Switch
1	301	53	1559	1554	1299	1440
2	670	99	1381	1403	1106	1241
3	540	87	1492	1443	1334	1189
4	1649	234	1555	1383	1301	1048
5	783	110	1570	1501	1304	1180
6	1088	196	1895	1914	1322	1490
Corpus	5032	779	1514	1530	1251	1218

Table 5: Speed comparison of the different text written with gaze control and switches

On average Kathrin requires 1.52 seconds to write a character using gaze control and 1.55 seconds for the same task with switches. A significant amount of time is lost once she makes a mistake. In order to compare the performance of the input method itself the median is more appropriate. With the eyes the median time per character is 1.24 seconds, with switches it is 1.22 seconds. The numbers suggest that for Kathrin there is currently no significant difference in typing speed using gaze control or switches.

Three main reasons have been identified to slow down average performance.

1) Words not contained in the dictionary require much more time to be entered for the first time (after entering a new word it is automatically added to the dictionary).
2) Correcting mistakes takes quite a long time
3) Entering punctuation marks and numbers has a very bad efficiency

Although the German dictionary contains about 350,000 word forms it does, of course, not contain all words the user might want to write. Especially less common names of places and people are not included. However, since the dictionary is extended automatically this will not be an issue in practice.

The average time required to type a punctuation mark is 2.8 seconds. Writing a punctuation mark takes approximately twice as long as writing a letter. This is mainly because punctuation marks are not included in the prediction list and always have to be selected from the menu. For instance writing a comma and a space requires 4 selections.

The following table shows the number of letters (including spaces), punctuation marks (including spaces), words, syllables and the average time required to enter a punctuation mark. It also includes the rate of words per minute, actual words per minute and syllables per minute.

Text	Letters	PM	Words	Syllables	Average time per punctuation [ms]		Words per minute, actual (W5)		Actual syllables per minute	
					Gaze	Switch	Gaze	Switch	Gaze	Switch
1	285	16	53	83	2451	1868	7.9 (8.4)	7.9 (8.5)	12.3	12.4
2	640	30	99	185	2501	2280	7.4 (9.5)	7.3 (9.4)	13.8	13.6
3	518	22	87	144	2331	2609	7.5 (9.0)	8.0 (9.5)	12.5	13.3
4	1578	69	234	445	2530	2197	6.0 (8.1)	6.9 (9.4)	11.5	13.2
5	753	30	110	223	2384	2024	5.8 (8.0)	6.3 (8.6)	11.8	12.7
6	1006	82	196	257	3723	3014	8.0 (8.2)	7.5 (7.7)	10.5	9.8
Corpus	4780	249	779	1337	2879	2470	6.9 (8.5)	7.2 (8.8)	11,8	12,5

Table 6: Performance measures

Words and syllables per minute were only included as these numbers are frequently used to measure the speed of written and spoken language. The words per minute rate is given in two flavors: the number given in parenthesis adopts the common practice to use a standardized word length of 5 letters (including spaces) while the first number states the actual rate for the given text.

Using the constant of 5 letters per word particularly neglects the fact that for different languages the number of letters per word differs significantly. As a matter of fact, the constant is used to make the speed comparable across different languages. However, it uses the term “word” in an inadequate way and is highly misleading. The characters per minute rate and the reciprocal milliseconds per character value describe the same rate in a more adequate fashion.

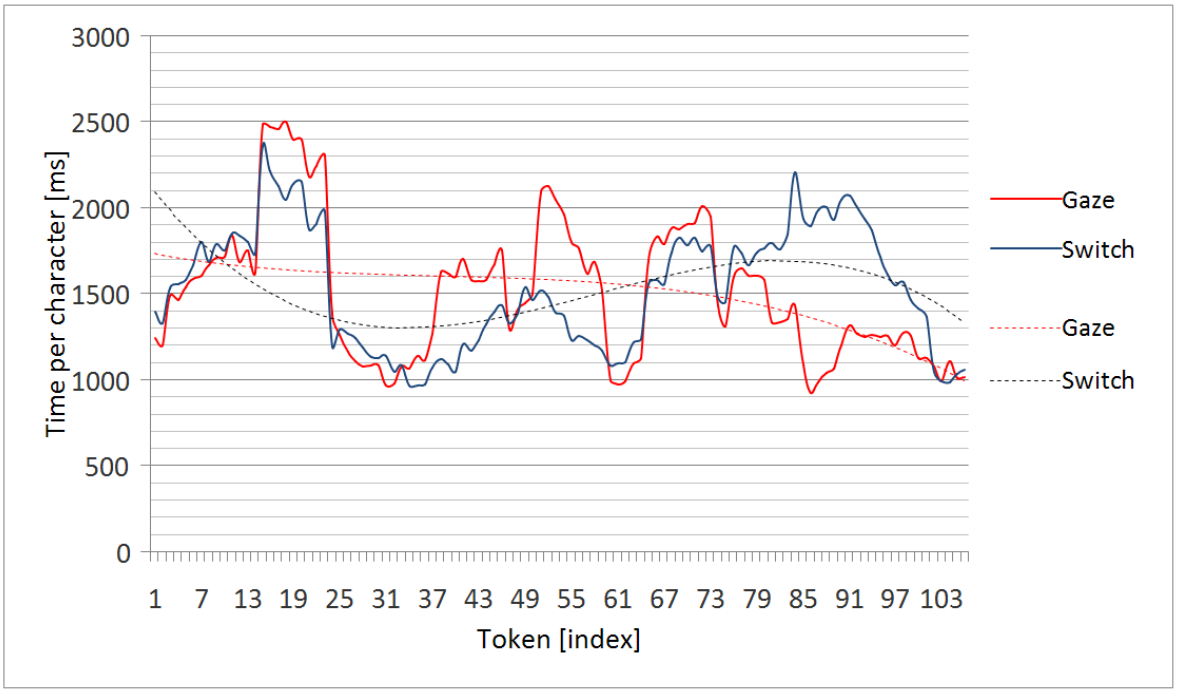

Diagram 7: Moving average (9 tokens) of the time per character for typing text 2 and cubic approximation.

Diagram 8: Moving average (9 tokens) of the time per character for typing text 6 and cubic approximation.

Effort Comparison

Some tests are filmed on video in order to analyze how calm Kathrin stays and to estimate how much effort is required to get the text typed. In addition she is interviewed after each session. From the video it can be seen that Kathrin stays much calmer when using gaze control. She also reported that for her it requires less effort writing with her eyes. From the video it can clearly be seen that the number of involuntary movements is much lower and that Kathrin stays more relaxed while using the gaze control system.

It is interesting to see if the speed of typing changes while she is writing longer texts. This would indicate fatigue. In Diagrams 7 and 8 the time to enter a character is shown as a moving average over 9 tokens. A token is a word, number or punctuation mark. It can be seen that the overall performance improves until a level of approximately 1200 milliseconds per character is reached. Even for longer texts Kathrin can keep this pace. From Diagram 8 it can be seen that correcting mistakes and typing words not contained in the dictionary takes significantly longer (peaks). The frequency of mistakes does not increased when writing longer texts. However, when writing for very long periods of time concentration is of course reduced resulting in more mistakes. The length of the texts was carefully chosen not to reach a point where concentration deteriorates. The period of concentrated writing seems to be longer with gaze control. The main reason for this is that the physical effort is reduced.

Conclusions

Even after working for more than 5 years with the switch-based solution Kathrin reaches comparable speeds working with gaze control. Her speed with gaze control still improves. Also the current dwell time (500 ms) is much higher than required. It is only kept so high as the dwell time for the letter controls cannot be chosen independently from the dwell time of the prediction list. For the prediction list a longer time is required as the items change dynamically. Once this issue is resolved we anticipate that she could ultimately reach a dwell time as low as 200 milliseconds increasing the input speed probably significantly below the switch based input.

Much more important is the required effort to type text. Kathrin clearly states that using gaze control the effort to enter text is significantly lower. Her impression (which is the most important factor) is also supported by the video analysis.

References

[Bates 02]
Richard Bates, Howell Istance
Motor adaptations: Zooming interfaces!: enhancing the performance of eye controlled pointing devices
Proceedings of the fifth international ACM conference on Assistive technologies, 2002; pp 119-126

[Garbe 01]
Jörn U. Garbe – Optimizing the Layout of an Ambiguous Keyboard Using a Genetic Algorithm
Diploma thesis, Universität Koblenz-Landau, 2001

[Lankford 00]
Chris Lankford
Effective eye-gaze input into Windows
Proceedings of the 2000 symposium on Eye tracking research & applications ETRA '00, 2000; pp 23-27

[Meyreis 00]
Meyreis, Frank Cathrin – Communication Aid with Thesaurus and Reduced Input. Diploma thesis, Universität Koblenz-Landau, 2000