

# TKSESH

## A hieroglyphic database system

[Serge Rosmorduc](#)

Équipe EC.art, laboratoire LRIA, Université Paris 8



### Abstract

We introduce tksesh, a multiplatform editor, database and dictionary software. Tksesh is both intended to be a toolkit to build applications for the philologist and an example of such application. The core of tksesh is a hieroglyphic editor which understands "Manuel de codage" encodings. The edited texts can be saved in a database, and referenced by the dictionary system, via hyperlinks. The dictionary can handle complex definitions by multiple authors. Most of the text in the dictionary has precise meaning, not only for the reader, but also for the computer. This allows automated treatments and possibly complex searches. An important feature of the dictionary is that it can contain references to the text database, in a readable way, and that clicking on these pops up the referenced text. Of course, exhaustive searches in the database text are also an option. The Tksesh system is written in the Tcl/Tk language, which is freely available for both Windows, Mac, and Unix systems, allowing it to be very portable.

### Introduction

The system we introduce here, called TKsesh, is a multi platform system (it works under windows 95, Unix, and should work on macintoshes as well) built around a hieroglyphic editor (compatible with the manuel de codage), and a database engine.

We started working on it quite a long time ago, but at that time it was a rather secondary work. Yet, it appeared that the system was potentially useful. Thus we decided to develop it further. What is presented here is a preliminary version of the software. We look forward for opinions and criticisms to improve it.

### Context and goals of the system

While working on our computer-science thesis, whose subject was automatic syntax analysis applied to middle Egyptian, the question of what was to be done with the texts we worked on was left in the background. However, we ended up with the idea of an integrated environment which would allow to store most information one reads and produce while working on a text, to share them, and, most important, to retrieve them. The system could also be a testbed for the Natural Language Processing systems we worked on.

The goal of Tksesh is both to help the realisation of a complete database of Ancient Egyptian texts, and to be a working tool for its user, where one can keep notes, lexical files, and so on.

## The components of the system

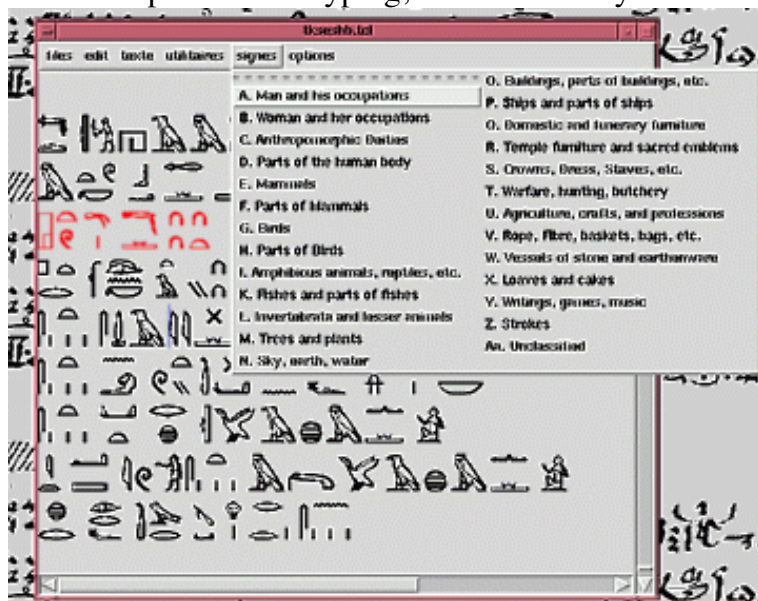
Tksesh is currently an integrated system : a number of quite different elements bound together in one program. We will now proceed to describe these elements, starting with the editor.


### The editor


The hieroglyphic editor is a central element of tksesh; its use will be described here, and information on its code will be given in the system's documentation. The editor's primary function was typing texts for databases, not for printing. So our primary goal was ease and speed of text typing, versus accuracy of printed representation.

#### typing text

As any hieroglyphic editor, tksesh allows entering the signs by a menu or by code. Simple sign grouping can be done by use of the Manuel de Codage symbols ":" and "\*", which allows fast typing; but complex grouping is done by menu. So it is impossible to enter incorrect codes in the system. The strong point of Tksesh as far as typing goes is that it is tolerant about codes. When a transliteration is typed, if the sign is not the expected one, a press on the spacebar will propose a new sign. For example, if I type "mr", I'll get



. If I want the pyramid-sign (O24), I'll press

"space" a few times, and get . Next time I'll enter "mr", the system will remember it and propose O24 first. Even better, when the list of possible signs is exhausted, the system looks in the [dictionary](#), for words having the said transliteration. Thus, in the present state of the dictionary, typing "iw" and spacebar will

propose :




### Grammatical informations

Supporting the grammatical codes of the *Manuel de codage* was essential for a system whose primary goal

was text databases. However, there's a problem with the manipulation of these codes. The display, and in general the way a user manipulate the text, is cadrat-oriented. Words separations and grammatical separations are sign oriented. Hence we have two problems : the first is to design how the user will manipulate the system to add these informations, and the second is how the system will extract words and the like. At the time being, grammatical markers are indicated by sign colors. Blue signs indicate word endings, yellow signs grammatical markers, and green signs word endings that are also grammatical markers (see Figure 1).



FIGURE 1 : WORDS ENDINGS IN LOUVRE C14 STELA

Currently, the user can only change the status of the last sign of a cadrat, typing "/" to make it a word ending, and "=" to make it a grammatical ending. If the marking is done at the time the text is entered, this is not a problem. However, it becomes one when the marking is done *a posteriori*. In these cases, one has to break the cadrats and rebuild them after marking. It is not convenient. Next version will include a sign-by-sign navigation mode, which will allow to navigate one sign at a time, and thus to change the status of *the current sign*.

A related problem is the difficulty to cut and paste *a word*. This possibility is highly desirable, since it would allow, for example, to add commands like "find the current word in the dictionary", enter the current word in the dictionary, and so on. To do this properly, we have to

1. be able to designate the current word --- and for this, the good unit is the sign, not the cadrat ;
2. be able to build a cadrat from parts of a cadrat.

This is not currently possible, but should be soon.

## References

As our goal was to build an "intelligent" text database system, with hypertext links all over the place, we needed a way to refer to particular points in a text in an efficient way.

We thought that readable references, very much akin to those used while referring to paper editions, would be fine. This has many advantages. First, it allows the references to be used outside the base : we support things like "o. DM 1567 verso, 2". A second point was that long texts are not always entered from the first line of the first page onward. In fact, you can start typing an interesting part of a text, enter information in the base about its contents, and some time later, decide, for completeness's sake, to type the rest. Our current system allows to explicitly give the position of the current part of the text. For example, in Figure 2, the whole content of P. L2 wasn't entered. But as the first line is explicitly "page 2, ligne 6", references to parts of the text will still be exact, even if we type the first pages afterwards.

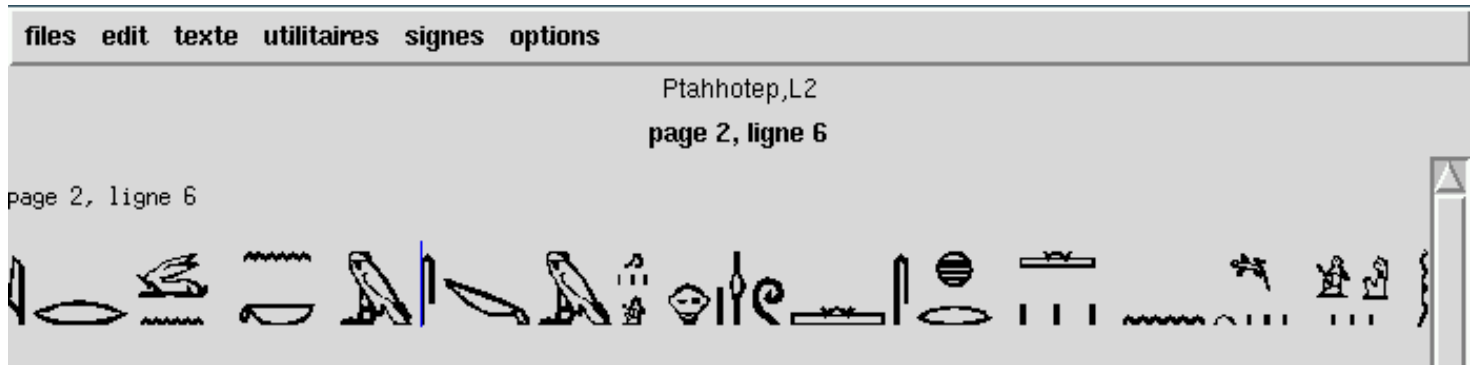


FIGURE 2 : EXPLICIT REFERENCES IN A TEXT

In order to create this system while keeping our files "Manuel de Codage"-compliant, we used the comment system. The first lines of L2 look like this once saved :

```
++TKSESH DATABASE FILE+s
++NAME Ptahhotep,L2+s
++COORDS=page 2, ligne 6+s
-i-r-wn:n-n:k\m-s-.sSm
```

Our main problem now with this reference system is to make it really usable by the end-user. The interface to the reference-setting system is probably not very user-friendly. As an example, I give in Figure 3, a picture of the system for the stela Berlin 1157, from an example file of Winglyph. The stela has four zones : three called A, B, C; and the main text below, which is not designated by a letter. So the text is separated in "lines" (which could be columns), grouped into zones. The lines are simply numbered (the value NUM for coord 1), and the zones (usually used for pages) are separated into A, B etc. Hence the numbering system is A1, A2, B1, C1, 1, 2, etc.

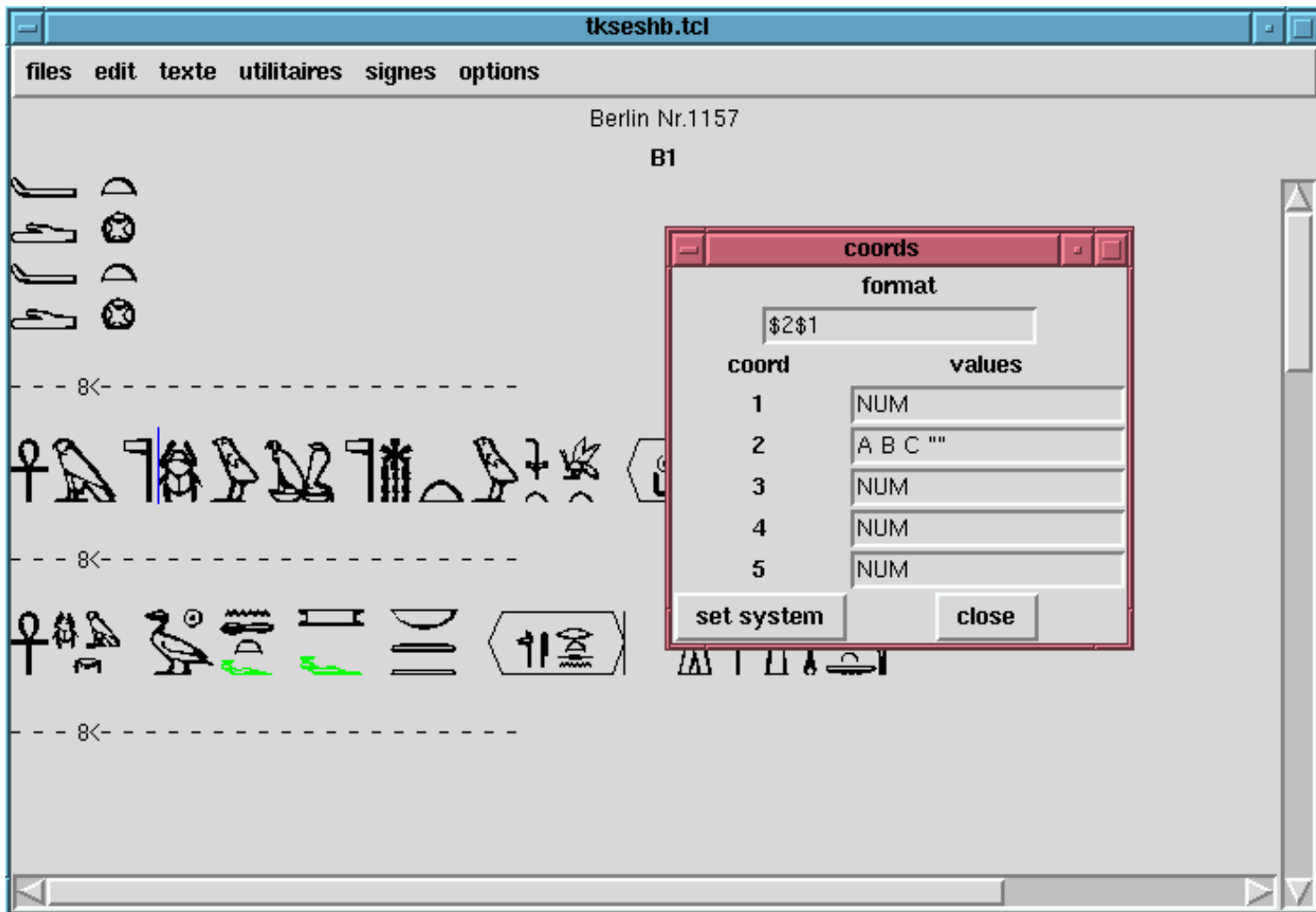


FIGURE 3 : REFERENCE CREATING SYSTEM

### Further needs

The main shortcomings of tksehb's editor stand in the domain of presentation. Improved cadrat rendering and column handling would be nice. More seriously, a font editor is absolutely necessary. We have already written one, but it runs only under UNIX, and thus can't be integrated in the whole system.

Another interesting addition would be the support of multiple reference systems. It might be interesting, for example, to be able to chose between a reference in the original source, or a reference in an edition (That is, for example, between *P. Leyde I 350 verso 13* and *KRI II,813,3*).

## The dictionary

### Introduction

When we decided to transform tksehb into a work environment for studying texts, the need for a linked dictionary arose naturally. In the first version of the dictionary, entries were quite simple : three fields : "transliteration", "spelling", "translation", the latter being free text. We included also the possibility to add hypertext references to the text database.

However, a real dictionary entry is something both complex and very structured. Entering it as free text is not a very good option, because the structure is lost to the computer. The human reader might be able to

reconstruct it, the system won't. Many automated processing that would be possible with a well structured lexicon are then impossible.

On the other hand, giving a very precise and rigid form to the dictionary would also be a problem, because it would force an artificial structure on all definitions.

Last, but not least, the structure proposed should be extensible, but no extension should break existing data.

Hence the structure we are going to describe now. This structure is supported by the editor built in the dictionary, which prevents unstructured entries to be made. It allows to enter many different style of dictionary entries, while keeping the maximum amount of structure information. We tested it by entering definitions from GARDINER's lexicon, FAULKNER, HANNIG, and of P. WILSON's *A Ptolemaic Lexicon*.

## Structure of the dictionary

The fields in the dictionary are of roughly three types : base fields, which contain one type and only one type of information (for example, a transliteration), complex fields, that can contain mixed information (for instance text in transliteration and hieroglyphs), and the group and comment fields.

### The group field

The group field is the main organizational device of the dictionary. Groups can be nested, to represent sub-meaning of a words, derived words, and so on. The basic point is that if a group contains multiple fields of the same kind, let's say multiple transliterations, they are supposed to be variants. In the case of translations, this would mean near-synonymous meanings. In Figure 4, all spellings for *ipt* are supposed to be equivalent. In a likewise case, the completion system described [above](#) should be able to propose all these writings.

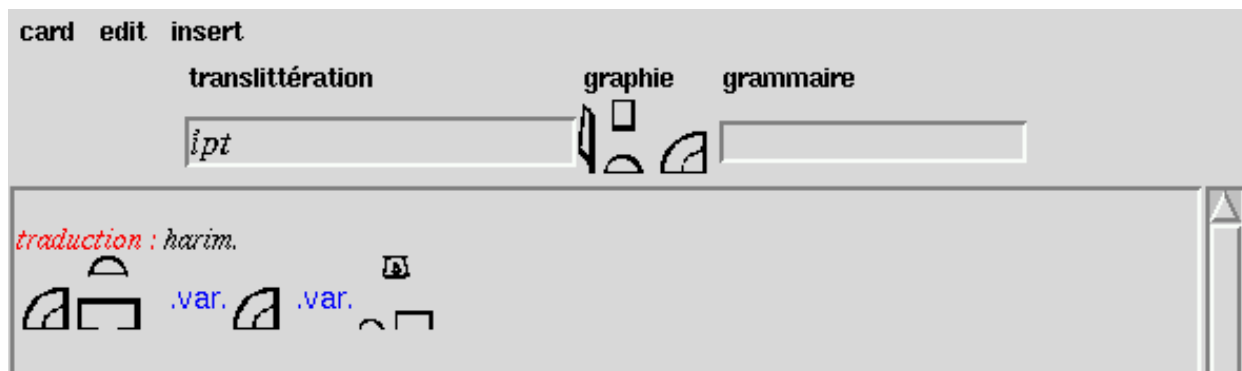


FIGURE 4 : REPRESENTATION OF VARIANT SPELLINGS

When some important information (transliteration, spelling, for example) is not available in a group, it is supposed to be inherited from its parent group.

Let's take, for example, the entry for *Aw* (Figure 5). The groups (indicated by french quotes << >>) delimit a number of new entries. The first one is for the adjective-verb meaning *be long*, which has the same transliteration as the head word, but different determinatives. It inherits its transliteration, but changes the translation and the spelling.

Then comes the expression "ib=f Aw". Expressions and composite words are a tricky problem, whose representation might need some improvement. At the time being, we have a number of tags that can be used in expressions to represent the currently defined word, an animate, or an inanimate. The other words might be free text, or explicitly transliterations of words (in which cases they are indexed. For instance, in the

current case, a search for the word "ib" will retrieve this definition).



FIGURE 5 : COMPLEX DEFINITIONS

### The comment field

In some cases, a dictionary definition can be a true little monograph on a word. In this case, the dictionary entry structure is not very efficient. This is the reason for the comment field's existence. It is there for anything that can't fit in a definition. Many fields can appear in it, like in a true little text editor.

### References

References are hypertext links to the text database. They are readable by the human reader, like in the example on the right, taken from Amenemope. These links are made quite easily, by using the "copy reference" menu option in the editor, and pasting the reference in the dictionary. Afterwards, a click on the reference will load the text at the proper place.

«  *expr* : <mot> *ht=f*.  
*traduction* : s'épancher.  
[Amenemope@page 22, ligne 11]»

### Signature

An important feature for further use of the system will be the possibility to share texts and dictionary entries, and conversely to identify the author of these entries. The Signature field is supposed to be used for this. A further development would be to fill it automatically -- at least, at data exchange time.

## **Indexation and search**

The system builds an index for each dictionary entry, into which it writes references for each transliteration, spelling, and translation. Any field of these types in dictionary entries is indexed, so even sub-definitions are entered in the database.

An important practical point is that text is treated to ease the search. For example, the hieroglyphs are save as Gardiner code, no matter their original form, and only the list of signs is saved. So, someone looking for "p\*t:pt" and typing "p:t-pt" will find his word. The transformation could even be improved by suppressing redundant phonetic complements and the like, but this is future work.

Transliteration are also simplified for searches : all 'j' are made into 'i', all points suppressed, etc. Note that this is only made at search time. Any point entered in the dictionary entries will be retained.

## **Extensions**

Looking at the current state of the dictionary, we see a possible generalization : the same mechanism can be used for freer text, for example for notes and the like. So we intend to reuse the code for the dictionary to allow the edition of general notes, which will benefit from the indexation mechanism of the dictionary.

Another interesting extension would be to add an reference system to parts of the dictionary. This would allow referencing a definition in another one. It would be very interesting in expression definitions, as it would allow to reference in a precise and explicit way the words which appear in the definition.

## **The transliteration and translation editor**

This facility will be a central working point of the system once finished. What we present now is just a model. It allows parallel edition of the text translation and transliteration, which can be saved separately. The advantage is that this allows multiple translations to be edited for one text.



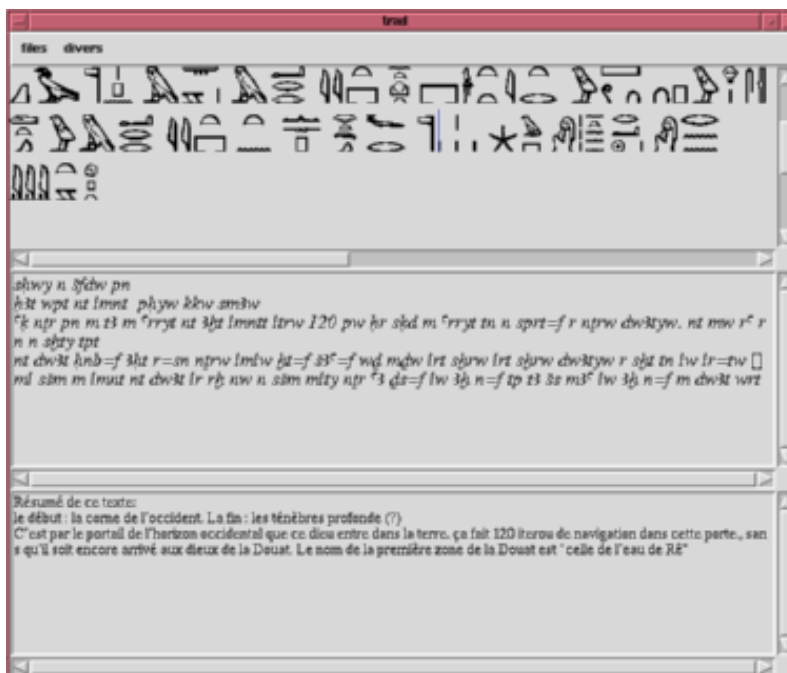


FIGURE 6 : TRANSLATION EDITOR

In the editor, all texts are synchronized : the edited line is always displayed in translation, transliteration, and hieroglyphs. We worked a little with the model, and it seems to be quite suitable. We linked it with the dictionary, and it is now possible to look for the word selected in the hieroglyphic window.

## Search facilities

One of the most important facility a database can provide is the possibility to retrieve the information it contains. So our base allows to find a words (given in transliteration) in the texts. For texts which have been manually transliterated, it should look in the man-made transliteration, because it's supposed to be accurate. But (as we'll see later), we have a automatic transliteration program that produce a rough transliteration. It's quite fast on a basic Pentium computer, and can be used if no time is available to write a transliteration. The search made in this case is not complete nor sure : some occurrences might be lost, and some words found can be errors. Yet, it can give a fast initial working base, and it should improve with our transliteration system. In Figure [7](#), we have the result of the search for the word *Axt*, and an example of a solution.

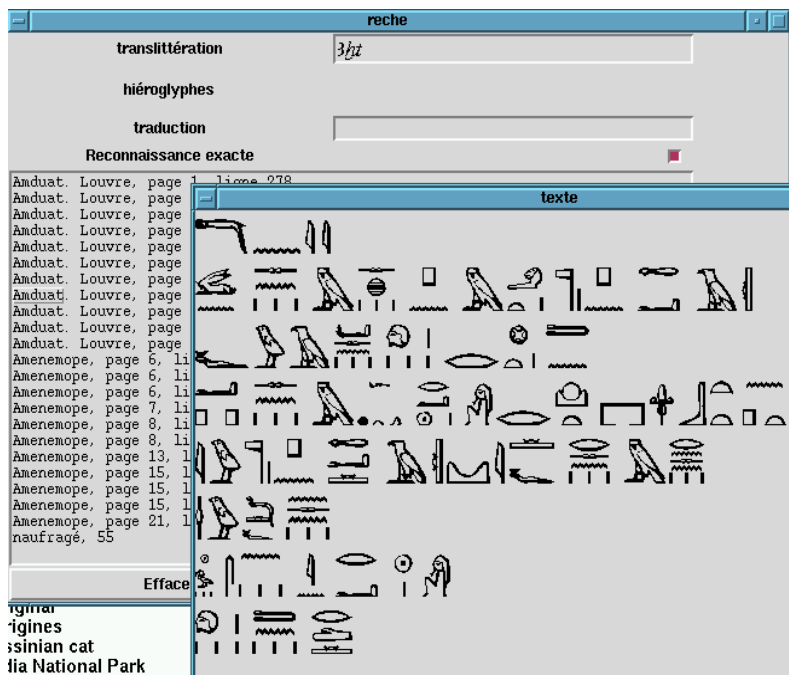


FIGURE 7 : SEARCH RESULT

## Natural Language processing

The system includes (currently only in its UNIX version) a prolog interpreter which allows us to use natural language processing techniques more easily. We had sketched in [ROS94] a transliteration system. After having worked more on syntactic analysis problems, we had a student, L. KERBOUL, working on the subject [KER97]. As the result was interesting, we decided to work again on transliteration, and if possible, to end up with a usable system. A detailed technical description does not fit here; let's only say that the basic principle is still the one described previously, but the performances have much improved. The main problem now is word cutting, which is a difficult problem. For this, the best solution would be an interactive one, the system proposing word-cuttings, and the user changing them. It is, interestingly, the solution used by some in editing Asian language (an example for Thai in MCB97)

## Further developments

The developments axis of the system will be :

- improvement of the editing system
- improvement of the natural language processing system -- ultimately with the incorporation of grammatical information
- creation of an exchange module --- to allow easy information sharing and exchange, using disks or even the net.

## Conclusions

The system I've just described is still in his infancy. Its foundations are however quite sound, and it works well. Now what it needs is users, to make it live, to propose friendlier interfaces, and useful additions.

# Appendix

## TCL and TK

TCL/TK is a programming language developed by John OUSTERHOUT at Cambridge University (USA) and then in the SUN research department. It is a simple, powerful, cross-platform language, specially designed to be embedable in programs written in compiled languages like C or Pascal. Tksesh is based on a number of extensions, written in C, to TCL/TK. It runs under UNIX and Windows 95. Due to the flexible nature of TCL, it is possible to use tksesh to write little application that would need to display hieroglyphs.

## Availability

The system will be available free of financial charges, as "textware": if the system is of some use to you, please contribute some texts. It would be definitely better to ask which texts are needed before sending them. At the time being, the software is quite young, and has had very few users. For this reason, I don't release it by simply putting it on a ftp server. You will have to register first. The details about obtaining the system will be available by next autumn on <http://www.iut.univ-paris8.fr/~rosmord/EgyptienE.html>.

## References

- **BIL95** S. BILLET, 1995, *Apports à l'acquisition interactive de connaissances contextuelles* Thèse de doctorat de l'Université Montpellier II (another approach of automated transliteration).
- **KER97** F. KERBOUL, 1997, *Translittération automatique des hiéroglyphes* Rapport de stage de l'ENSTA
- **MCB97** S. MEKNAVIN, P. CHAREONPORNSAWAT and B. KIJSIRIKUL, 1997, *Feature-based Thai Word Segmentation* In [Natural Language Processing Pacific Rim Symposium 1997, Phuket, Thailand](#)
- **ROS94** S. ROSMORDUC, 1996, [Traitement automatique du langage naturel en moyen égyptien](#) In Robert VERGNIEUX, editor, [Xieme conférence Informatique et Égyptologie](#)
- **ROS96** S. ROSMORDUC, 1996, [Analyse morpho-syntaxique de textes non ponctués](#) Thèse de doctorat, École normale supérieure de Cachan,

[Serge Rosmorduc](#)