A Revised Encoding Scheme for Hieroglyphic

Mark-Jan Nederhof*
University of Groningen
Faculty of Arts
P.O. Box 716
NL-9700 AS Groningen
The Netherlands
markjan@let.rug.nl

Abstract

We describe a new encoding scheme for hieroglyphic that is very different from that offered by the Manuel de Codage, in its functionality and its syntax. We discuss the problems of the Manuel de Codage, and the solutions to these problems offered by the new encoding scheme. We also discuss the wider RES project, of which development of the new encoding scheme is the main objective.

1 Introduction

The most widely known form of encoding for (Ancient Egyptian) hieroglyphic text is proposed by the Manuel de Codage [2], henceforth referred to as "MdC". The MdC has played an important role in typesetting of hieroglyphic in the past 20 years, and most encoding schemes for hieroglyphic today are dialects of that proposed by the MdC.

However, the encoding scheme has apparent problems, related to:

- its functionality, i.e. what kind of hieroglyphic can be described and what can be described apart from hieroglyphic,
- its syntax, i.e. how hieroglyphic constructions are described, and lastly,
- its lack of standardization.

In this paper we will address these issues and present the Revised Encoding Scheme (RES) for hieroglyphic that solves many of MdC's problems. The development of this encoding scheme is the main objective of a larger, ungoing project, the RES project, which, apart from devising the alternative encoding scheme itself, has the following four objectives:

1. To make an in-depth study of the adequacy of RES by looking at original hieroglyphic inscriptions, and at the requirements for hieroglyphic transcription of hieratic.

*Supported by the Royal Netherlands Academy of Arts and Sciences. Secondary affiliation is the German Research Center for Artificial Intelligence (DFKI).

2. To produce a formal description of the syntax and meaning of RES, and publish this on the World Wide Web. This also includes formal descriptions of relevant algorithms needed to process RES encoding, such as those for scaling and positioning.
3. To produce a prototype in C of software to render the encoding into graphical form, and to test this software on various platforms. The software is released under a GNU general public license, which will allow everyone to use it free of charge.
4. To make an inventory of constructions from different dialects of the Manuel de Codage format, and to implement software to convert MdC into RES, to prevent existing resources from becoming obsolete.

At the moment, work on the first three objectives is all but completed. The study of the adequacy of RES has been under way in an informal manner for at least two years now, but we plan to intensify this effort by systematically encoding a number of good monumental inscriptions on the basis of photographs.

The description of RES is virtually complete, and is available at:

`http://www.dfki.uni-sb.de/~nederhof/RES/index.html`

At the same site, we offer the C source code of the implementation, which can be compiled under UNIX. Provisions for compilation on other platforms will be available soon.

The conversion from MdC into RES however is still to be implemented.[1]

The structure of this paper is as follows. In Section 2 we motivate the necessity of the RES project by investigating some problems that the MdC has and the solutions offered by the Revised Encoding Scheme. Other design decisions of RES are discussed in Section 3. The final section discusses the implementation.

2 MdC versus RES

2.1 Relative positioning

The core assumption of hieroglyphic encoding following the MdC is that signs and groups of signs may be positioned either next to each other or above each other. The former is indicated by the operator `*` and the latter by `:`. One may also use `-`, which functions as `*` for horizontal text and as `:` for vertical text. (The two-fold meaning of `-` can be exploited to render in a horizontal way an encoding designed for vertical text, and vice versa.)

Although such simple relative positioning of signs may indeed be found in products of Egyptological typography, original hieroglyphic is by no means so constrained. In fact, already in Gardiner's Egyptian Grammar [3], which relies on technology more than half a century old, we find relative positioning that is doubtless closer to the original hieroglyphic than to anything that could be described within the bounds of the MdC.

That the MdC is inadequate for indicating relative positioning reflecting original hieroglyphic seems to be ignored or even denied by some, with the argument that hieroglyphic typically published in books stays within the bounds of the MdC. However, this argument is circular, since it is often MdC, or one of its dialects, that is used for typesetting hieroglyphic in books. This may explain the paradoxical situation that the misconception that

[1]This work will rely for a large part on previous efforts by Serge Rosmorduc, who has formulated a formal syntactic description that includes most dialects of the MdC; see `http://www.iut.univ-paris8.fr/~rosmord/HieroEncoding/MDC/mdc.html`.

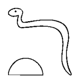
MdC provides satisfactory means of relative positioning seems to be growing, in spite of technological advances in typography that have widened the gap between what is desirable and technically possible on the one hand, and what the MdC offers on the other.

As far as the inadequacies of the MdC have been acknowledged, they have often been dealt with as follows. An MdC encoding is composed in a first phase to obtain the rough positioning of signs, and in a second phase the signs are given their final positions, by manually editing the output of the renderer, e.g. using the mouse. This approach is perhaps the only option in cases where relative positioning should very precisely agree with an original hieroglyphic text, e.g. for preparing a printed publication. Scientific requirements may demand such manual "fine-tuning" of positioning, and possibly scaling, and since a scientific publication may be printed in large editions, the additional efforts by the author may be well justified.

However, this approach is inappropriate or even useless for many other applications, most notably those involving the World Wide Web. E.g. a user may want to download an encoding of a hieroglyphic document, and view that using the fonts, the font size and screen size of his choice. He may also want to combine the hieroglyphic with other resources he may have found on the same document, such as translations and transliterations, possibly obtained from other, independent sites.

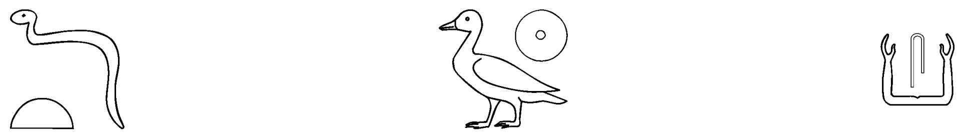
It is likely however that a manually fine-tuned positioning of signs that was composed on the basis of a fixed font, font size, page height and width, etc., will lead to unexpected, and moreover undesirable results if the text is rendered under other conditions. For example, two occurrences of signs that should occur close together, and that have been fine-tuned as such, may suddenly overlap when a different font is used in which these two signs are slightly bigger. It can therefore be concluded that manual fine-tuning is not a reasonable option for such applications.

There have been some attempts to extend the capabilities of relative positioning of the MdC. A common example is the use of the operator `&`, present in some MdC dialects. This can be used to form a kind of ligature. E.g. `D&t` can be defined to represent the sign I10 with sign X1 underneath, as follows:



This is a very questionable solution however. First, a casual inspection of actual hieroglyphic or hieratic texts should convince anyone that there is no natural bound to the number of pairs (or triples, ...) of signs with a relative positioning similar to that represented by `D&t`. For encoding a new text, one will likely find new such combinations of signs, which will first need to be defined before they can be used in the encoding. Secondly, defining combinations of signs such as `D&t` typically requires an effort that is not so different from manual fine-tuning, which we already argued above is not a viable option for many applications. Therefore, the use of operator `&` does not solve any problems, it just moves them from the encoding of the actual text to the definition of sign combinations.

The approach in RES is very different from the approaches described above, and consists in an extension of the very limited repertoire of `*`, `:` and `-` by other primitives for indicating relative positioning. We have found that only a few additional such primitives suffice to be able to represent many more types of relative positioning found in original hieroglyphic. Moreover, our primitives are generally font-independent. Although one can express fine-tuning in RES for specific fonts and parameters such as height of lines and



`insert[bs](I10,X1)` `insert[te](G39,N5)` `insert(D28,S29)`

Figure 1: The function 'insert'.

width of columns, replacing the font or changing these parameters does not lead to awkward results. In particular, two signs cannot normally be rendered in a way that they overlap, unless the encoding specifies them as such.

One example of a primitive that leads to a significant increase in expressiveness is the function 'insert'. It takes two groups of signs, scales them independently, and inserts the second in the first at a specified location. The second group is then scaled down such that it fits in an appropriate way within the first group, and it may also float up or down, or left or right, if that helps to maximize its size.

A few examples are given in Figure 1. The first example shows the alternative to what would be written as `D&t` in MdC dialects. The argument `bs` means that the second group (`X1`) is to be placed in the bottom-left edge of the first group (`I10`). (`b` stands for 'bottom' and `s` stands for 'start', which means 'left' in a left-to-right text direction; the words 'left' and 'right' are absent from RES to avoid confusion in the case of a right-to-left text direction.)

It is important to stress that if a different font is taken and signs `I10` and `X1` obtain different sizes, then appropriate software will still be able to render the combination of these signs in a satisfying way, since it does not rely on coordinates that are fixed in the encoding. It scales and positions sign `X1` inside of `I10` given the sizes of those signs in the font, within the very global constraints provided by the encoder. Another way of looking at this is that the encoder can simply express in the encoding that which he sees in a text, which is here "an occurrence of `X1` in the bottom-left corner of `I10`", without having to use a ruler or specialized graphical software. This contrasts with most dialects of the MdC.

Another common example illustrated in Figure 1 is the sun (`N5`) in the empty space above the duck (`G39`). A last example shows insertion of `S29` in the middle of `D28`. Here, `S29` floats a little upward from the center of `D28`, while remaining within the bounding box of `D28`, in such a way that its size can be maximized.

Also different from the MdC is the *interpretation* of the basic repertoire of operators `*`, `:` and `-`, i.e. the kind of scaling and positioning of groups of signs that these operators lead to. The MdC assumed the existence of "quadrants". This is a familiar concept in Egyptology, but the interpretation of quadrants as blocks of uniform sizes is at best a very rough approximation of reality, and at worst an imaginary concept without any empirical basis. (See for example how [2] defines the operators for shading, which presupposes a fixed quadrant size.)

We have defined a scaling and positioning algorithm that is much more flexible. The exact meaning that this algorithm assigns to the operators allows us to parameterize them to influence scaling of signs and the distances between them, in ways that agree much better with observed arrangements of signs in original hieroglyphic.

A good example of the extended possibilities concerns a group of three occurrences of `Aa1`; we write `Aa1` here as the *mnemonic* `x` (see Section 3.2). A naive approach is to encode

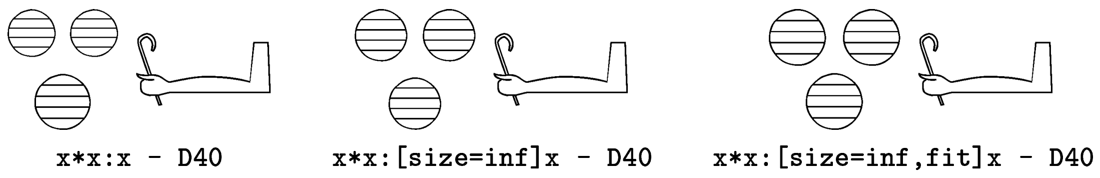


Figure 2: Parametrization of common operators.

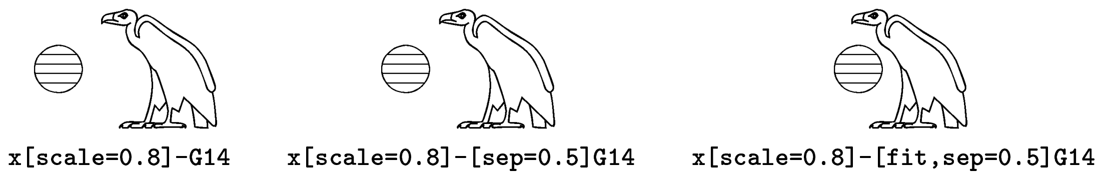


Figure 3: Other use of fitting.

this as in the first example of Figure 2. However, this does not lead to a satisfactory outcome, in that the upper two occurrences of `x` are made smaller than the lower one, which is inconsistent with what we may typically observe in original inscriptions, where all occurrences of `x` have the same size.

The explanation is that `x*x` is wider than 1 EM (the basic unit of length in typography), and a scaling algorithm (irrespective of whether it was designed for MdC or RES) works in such a way that it scales down the groups of signs on both sides of the operator `:` to be no more than 1 EM wide, by default.

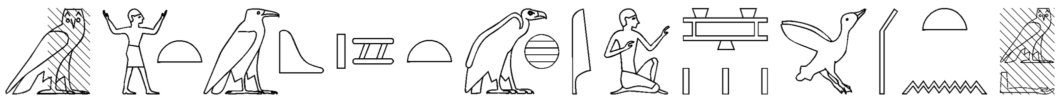
It has been suggested that this can be solved by manually scaling down the lower occurrence of `x` by a certain factor,[2] but this is not an acceptable solution, since first, this factor can only be determined by trial-and-error, and secondly, a different font may require a different factor.

The solution offered by RES is to parameterize the scaling algorithm in such a way that the maximum width of the groups on both sides of `:` can be set to any positive value, and even to `inf` ('infinity') if there should not be any restriction on the width. This is demonstrated by the second example in Figure 2.

Although now all occurrences of `x` have the same size, the result is still not quite correct. Since the amount of white space between the upper and lower groups is determined by the bounding boxes, the lower occurrence of `x` is further away from the two upper occurrences than the upper occurrences from each other. This is corrected by the use of what we call *fitting*. This means that the minimum distance between groups is not measured in terms of the distance between bounding boxes, but between pixels. This is exemplified in the third example of Figure 2.

Fitting is actually quite common in original hieroglyphic, as can be easily verified by investigating almost any inscription. A typical example is illustrated by the three encodings from Figure 3. The `x` (scaled to 0.8 times its natural size) can simply be put next to `G14`, but the result is not satisfactory, since the signs are too far apart. One may reduce the distance to only 0.5 times the normal size between bounding boxes, as in the second encoding in Figure 3, but that is still not as it may appear in an original inscription. Note that allowing negative 'sep' values would be highly problematic if we

[2]See `http://www.ccer.theo.uu.nl/codage/codage.htm` by Hans van den Berg. This page also mentions something called "WinGlyph's 'Wide groups' option", that should help to avoid this problem, but this is not explained further.



```
[rl] ![shade] m:a ![noshade] - t:n -[sep=0.3] T14*G41 - N31:Z2 - A1*i -
x[scale=0.8]*[fit,sep=0.5]G14 - t*N23*[sep=0.5]Z1 -
q*[fit]A*[fit]t -[fit] A28*m[s]
```

Figure 4: Eloquent Peasant B1 37, after [7].

seek a font-independent way of encoding. More realistic is the third encoding in Figure 3, where the distance between `x` and `G14` is measured between pixels, and `x` moves as close to `G14` as the shape of `G14` and the specified 'sep' value 0.5 allows.

In RES, the direction of text can be specified at the beginning of a piece of encoding, by `[lr]`, i.e. left-to-right, which is the default, `[rl]`, i.e. right-to-left, `[tblr]`, i.e. top-to-bottom left-to-right, or `[tbrl]`, i.e. top-to-bottom right-to-left. Note that very few dialects of MdC, if any, allow text direction to be specified in the encoding itself.

2.2 Syntax

Another problem of the MdC is its bizarre syntax. For example, operator precedence is counter-intuitive. Most notably, `#` is often mistakenly assumed to have a higher precedence than `*` and `:`. E.g. `A1*B1#` means `(A1*B1)#` rather than `A1*(B1#)`.[3] Furthermore, the signs < and >, which the MdC uses to delimit cartouches and similar types of containing structures, are an impediment to applications involving XML and HTML, and \, which is used for indicating mirroring and rotation in MdC, complicates integration into e.g. LaTeX (not considering far-reaching extensions to LaTeX such as HieroTeX[4]).

In contrast, RES uses a minimum of special symbols. There are a few functions and attribute names, some of which we have already seen above, but they are written entirely in the Roman alphabet and are intuitive and can be easily remembered. Furthermore, many constructions have a uniform syntax. In particular, all parameters of signs and operators are written in a list between square brackets, which is appended behind the element in question. For example, `A1[red,shade]` denotes a red and shaded version of `A1`, and `cartouche[red,shade](N5-N28:f)` denotes a red and shaded version of `cartouche(N5-N28:f)`.

In the same vein, all manipulations of global properties such as color and shading can be written in the same way, viz. as an exclamation mark followed by a list of global properties that need to be changed, enclosed in square brackets. E.g. `![red,shade]` means that the following text is to be printed in red, and is to be shaded, and `![noshade]` switches off global shading. Retrograde text can be obtained by the switch `![mirror]`.

Furthermore, operator precedence is relatively straightforward, since the only binary operators are `-`, `*` and `:`, and these have the same operator precedence they have in the MdC.[5]

Figure 4 shows an extended example.

[3]There is a rational explanation for how this situation evolved, via a binary operator `#` that was used to superimpose groups of signs (personal communication with Serge Rosmorduc).

[4]See the work by Rosmorduc, at `http://www.iut.univ-paris8.fr/~rosmord/Sesh/Sesh.html`.

[5]There is actually a fourth operator, viz. ^, which is used to attach footnote markers and short comments to hieroglyphs, but since its second argument is a string and its first argument is a single hieroglyph, there cannot be any confusion with regard to operator precedence.

2.3 Beyond hieroglyphic

Another notable defect of the MdC stems from the unfortunate design decision to let encodings represent complete documents, so that the encoding scheme offers (and constrains) much more than just hieroglyphic. Creating software to handle such complete documents satisfying the full standard requires much effort. For example, the software designer has to write code to divide documents into pages, sections and lines, choosing fonts for different types of header, etc., although all of this is standardly available in each off-the-shelf text-processing or document-preparation system.

That the MdC defines a complete document type may also have had much adverse consequences for users, who have found themselves subject to the possibly misplaced assumptions by its designers about what they need or do not need. For example, one common dialect of the MdC offers a possibility to include text in the Roman[6], Greek and Coptic alphabets, but not the use of, say, Cuneiform writing systems. There is no obvious justification however for the assumption that any fixed repertoire of writing systems that a dialect of the MdC provides are all that a user may need.

For these reasons, we have chosen to let the RES encoding scheme include nothing other than hieroglyphic, save a few additions for text-critical notation such as brackets and footnote markers. This significantly simplifies the implementation of hieroglyphic-processing systems, since we establish a separation of concerns whereby everything that is related to graphical aspects of hieroglyphic, in particular scaling and positioning of signs, can be realized by software that is independent of the application. This software can be linked to other software to include hieroglyphic into many different types of text processing and document preparation systems, in a way that fits the application at hand.

Note that our approach implies that any form of lexical, syntactic and semantic annotation is banned from the hieroglyphic encoding itself. Far from restricting the users, this design decision liberates them from the constraints imposed by the MdC, and much more appropriate ways of annotation can be found, for example by defining a layer of XML markup on top of RES encoding.[7] By specifying a particular DTD, the users may choose forms of annotation that fit the specific needs of their applications. If an application requires grammatical tags to be represented by particular colors (cf. some existing implementations of MdC dialects), this may be solved by style sheets that are fully under control of the user.

To give a simple example, one can imagine an XML format defined on top of RES that allows expressions such as:

```
<verb> sDm-m </verb> - <suffix> f </suffix>
```

In the implementation, hidden from the user, this could for example be translated to:

```
![blue] sDm-m ![black] - ![green] f ![black]
```

In other words, verbs would be printed in blue, suffixes in green and everything else by default in black.

Concerning text-critical notation, in particular different types of brackets, RES does not make unnecessary assumptions about the style the user might prefer, but one may define an XML format on top of RES allowing expressions such as:

[6]However, the MdC [2] never specified whether diacritical signs were allowed, and if so which ones.

[7]For XML, see `http://www.w3.org/XML/`.

```
<damaged> sDm-m </damaged>
```

which could be translated into pure RES of the form:

```
"[" - ![shade] sDm-m ![noshade] - "]"
```

which results in shaded text enclosed in square brackets, or to a different form that an individual user may prefer.

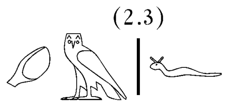
Another possible use of XML tags in RES encoding is for specifying positions in a text. This may look like:

```
sDm-m <position page="2" column="3"/> f
```

It is very easy to write a script that replaces such tags by pure RES expressions, so that we obtain something like:

```
sDm-m - "|"^"(2.3)" - f
```

which appears as:



Although some RES constructions here are used for a slightly different purpose than they were designed for, this is not objectionable, since the last expression above is hidden from the user.

2.4 Standardization

One last serious defect of the MdC is the lack of an international standard. The last printed form of the MdC [2] was far from explicit about the exact syntax and meaning of constructions, and since its possibilities were restricted, various pieces of software that have appeared since 1988 have extended the encoding scheme, but not in any organized way, which resulted in several different, incompatible dialects, defined only indirectly by what the corresponding software accepts.[8]

Many of our efforts have been directed towards documenting the syntax and the meaning of the Revised Encoding Scheme, and this material has been made available on our Web pages. When changes to the encoding scheme are ever found to be necessary, we will record these meticulously, on the basis of version numbers assigned to the various stages of evolvement of the encoding scheme. Such changes will then be implemented in the software.

[8]The highly objectionable practice of developing a standard as an a posteriori description of what a certain piece of software can handle, rather than developing an a priori suitable standard for which one then tries to build accurate implementations, is a blatant element of [2]. Cf. p. 15: "[...] the Glyph programme [sic], linked to this enterprise from the beginning, has been improved, which had to be included in the Manual".

3 Further design decisions

3.1 Sign lists

Apart from the sign list by Gardiner [3] there is no stable collection of hieroglyphs. Respective versions of the so called 'extended library' [2, 5] suffer from unfathomable inconsistencies, both internally and with respect to each other.[9] For this reason, RES does not commit to any fixed sign list. We assume external font files and an external index that maps the names of available hieroglyphs, the so called Gardiner codes, to signs in those font files. We do commit to a certain format of names, namely strings consisting of a category (letters `A` thru `Z` with the exception of `J`, and `Aa`), followed by a number between 1 and 999, boundaries included, followed by an optional affix.

Earlier versions of the Manuel de Codage used lower-case letters for affixes, but the last printed version made the transition to upper-case letters.[10] Despite of this, virtually all modern publications (e.g. [1, 4, 6]) prefer lower-case letters. Although there is little reason to favour one over the other, we adhere to the most common notation, that of lower-case letters, since allowing both is an unnecessary source of confusion.

Page 29 of [2] suggests some affixes have a special meaning, namely that of rotating signs, but this seems to be allowed for only a select number of signs. Since rotation has been indicated in other ways in RES and in more recent dialects of the MdC, we have ignored this matter, and any (lower-case) letter can be used as affix, without any special meaning.

Both the original sign list from [3] and different versions of the extended library contain many signs that are graphical combinations of other signs. In a majority of cases, such a composite sign also has a meaning that is the composite of the meanings of the constituent signs.[11] Whereas half a century ago Gardiner could perhaps justify inclusion of such composite signs in his list on the basis of the primitive technical possibilities at the time, it seems that in the last decade or so the inclusion of such signs is motivated by the limited possibilities of the MdC. In other words, it appears that the shortcomings of the MdC have contributed to an avoidable proliferation of composite signs in the extended library.

For example, the sign `M163` from the extended library can be written in RES as `M23:[sep=0,fix]Aa1[scale=0.8]`, which states that `M23` should occur on top of `Aa1`, which is scaled down somewhat, and there is to be no space between the two signs. The sign `D184` from the extended library can be written in RES as `insert(D28,S29)`, as already illustrated in Figure 1.

3.2 Mnemonics

There are sign names such as `mSa` that can be used as alternatives to Gardiner codes such as `A12`. We will call such names *mnemonics*, since these names are analogues of mnemonics in assembly language, where they are alternatives to numbered instructions in machine code [8].

[9]See `http://www.dfki.uni-sb.de/~nederhof/AEL/signlists.html` for a comparison of different sign lists.

[10]Cf. p. 15 of [2]: "Several details of the former edition have been altered. The small letters replacing the asterisks used by Gardiner have also been replaced by capitals".

[11]This is stated for the sake for completeness, but may be argued to be not very relevant. I believe that the position that a sign list of hieroglyphs should be based on characters (in the formal sense of the word, cf. Unicode) rather than on signs (abstractions of physical appearances of hieroglyphs) is untenable for any practical application in Egyptology.

Mnemonics seem indispensable for editing encodings of hieroglyphic, but they are not unproblematic. Some problems stem from two unfortunate design decisions in the MdC. The first is that there are signs that are associated with more than one mnemonic. The second is that, taking [2] literally at least, Gardiner codes are forbidden for those signs that are also associated with one or more mnemonics.[12] If we take these two aspects to their logical consequence, then it is not possible to encode hieroglyphs on the basis of their physical appearance alone, without interpretation of the text. For example, for determining whether a sign given by Gardiner code `S3` should be encoded as `dSrt` or `N`, one needs to determine the meaning of the sign in the context. In the case of e.g. `S15` it can be uncertain whether to read `tHn` or `THn` even when the meaning of the sign in the context is clear. In our opinion this is unacceptable, and therefore we also allow Gardiner codes to be used next to and in place of mnemonics.

The fact that there are signs with more than one mnemonic is by itself a very troublesome aspect. It means that mnemonics are necessarily more than a matter of the interface. If each sign had at most one mnemonic, one could internally store signs by their Gardiner codes, and translate these to mnemonics back and forth for purposes of editing. This would open the possibility to introduce user-defined lists of mnemonics, without the risk of incompatibility of encodings obtained through different such lists. However, if one stores, say, `dSrt` by `S3`, one cannot know whether the latter should be mapped back to `dSrt` or to `N`. Therefore, the internal encoding should maintain the mnemonics.

The reason why one would want to introduce user-defined lists of mnemonics is that the list of mnemonics from the MdC (or any such list for that matter) makes some arbitrary choices. First, Egyptologists trained in the German school, where they tend to write `j` instead of `i`, will certainly be confused when they have to write transliteration with `j` and hieroglyphic encoding with `i`, and the distinction between `z` and `s` is not always regarded as essential. Secondly, what is the most common Gardiner code that matches a combination of consonants is something that changed in the course of Egyptian history, so the list may be more appropriate for some periods than for others.

At the moment, we adhere to the list of mnemonics from the MdC (after correcting some obvious errors), but suggest that in the future it is desirable to disconnect the mnemonics from RES, or at least to create some provision for introducing user-defined mnemonics at the level of the interface, which are translated to standard Gardiner codes in the internal format that is actually stored and exchanged. User-defined lists of mnemonics should then allow at most one mnemonic for each Gardiner code.

3.3 Why not XML?

There are some proposals to encode hieroglyphic using XML, e.g. by Serge Rosmorduc[13] and Spencer Tasker[14]. Although XML has useful features, such as the possibility to formally define the syntax of a format through a DTD, it seems to have more disadvantages than advantages for hieroglyphic. The main disadvantage is that XML is not normally readable by people, and therefore specialized editors are necessary to edit text. Although

[12]Cf. p. 17 of [2]: "The two and three-consonantal characters should be encoded according to their phonetic value". Cf. also p. 15: "The codes M22a, O38a and R8a have been replaced by their phonetic value [...]".

[13]See `http://www.iut.univ-paris8.fr/~rosmord/HieroEncoding/DTD/`.

[14]At IAE Computer Working Group Table Ronde Informatique et Egyptologie, Würzburg, 11-14 Juli 2000. See `http://www.uni-wuerzburg.de/aegyptologie/ritualszenen/Abstracts1.html`.

specialized editors may be used in any case, we feel that encodings that are readable without specialized software are preferable over those that are not.

Note that the possibility to provide a precise description of an encoding scheme, e.g. through a context-free grammar, does not depend on the scheme being based on XML. This is illustrated by the formal description of the syntax of RES that can be found on the Web site of the RES project.

Further note that by defining an XML format on top of RES, as illustrated in Section 2.3, we obtain a convenient distinction between two kinds of information, the first being what the hieroglyphic itself looks like, and the second involves interpretation of the hieroglyphic.

4 The implementation

The main purpose of the implementation was to verify that the more innovative features of RES could indeed be realized. A second motivation was to advance the state-of-the-art in hieroglyphic typesetting, by offering a free C program, called res2image, that can compete with the best software that is currently available. The advantage of C is that it can be compiled on any modern computer platform.

The functionality of res2image covers what many different systems for processing hieroglyphic would have in common, but it is not itself a complete system. In particular, res2image cannot be readily used for turning an entire Egyptian text into graphical form, but it can be very easily extended to do just this, by building a simple script around it. This is in agreement with the nature of RES, which does not make unnecessary assumptions about the needs of users, especially not those that go beyond hieroglyphic itself.

The program res2image is based on TrueType technology, and can turn a string of RES encoding into TIFF, (encapsulated) postscript, or PNM. By piping the output through external programs, one can also add e.g. GIF and JPEG to the list of available output formats.

One very useful feature of res2image is that it is capable of splitting a string of RES encoding into two strings, the first of which is the longest prefix that fits within a given length restriction. This can be used e.g. to include hieroglyphic into running text (i.e. in a Latin script) with pages of a fixed width; the first string is taken to fill up the current line on a page, and the second can be placed at the beginning of the next line.

The program has many options, e.g. to choose device resolution, to use colors or to indicate colored text by underlining, to render text in rows or columns possibly overriding the text direction specified in the encoding, to influence the appearance of shading, etc.

Acknowledgements

Frank Nusselder spent countless hours helping me with graphical formats and the intricacies of C and makefiles. I am indebted to Michel Mudde for intensive discussions in the early stages of developing the encoding scheme. I am also grateful to many other people who contributed to this work, among whom are Serge Rosmorduc, Paul Sciortino and Daniel Tuijnman.

References

[1] J.P. Allen. *Middle Egyptian: An Introduction to the Language and Culture of Hieroglyphs.* Cambridge University Press, 2000.

[2] J. Buurman et al. *Inventaire des signes hieroglyphiques en vue de leur saisie informatique.* Institut de France, Paris, 1988.

[3] A. Gardiner. *Egyptian Grammar.* Griffith Institute, Ashmolean Museum, Oxford, 1957.

[4] E. Graefe. *Mittelägyptische Grammatik für Anfänger.* Harrassowitz Verlag, Wiesbaden, 1994.

[5] N. Grimal J. Hallof and D. van der Plass. *Hieroglyphica.* Publications Interuniversitaires de Recherches Égyptologiques Informatisées, Utrecht, Paris, 1993.

[6] R. Hannig. *Grosses Handwörterbuch Ägyptisch-Deutsch: die Sprache der Pharaonen (2800-950 v.Chr.).* Verlag Philipp von Zabern, 1995.

[7] R.B. Parkinson. *The Tale of the Eloquent Peasant.* Griffith Institute, Ashmolean Museum, Oxford, 1991.

[8] E.F. Sowell. *Programming in Assembly Language: MACRO-11.* Addison-Wesley, 1984.