

Additional control characters for Ancient Egyptian hieroglyphic texts

Andrew Glass, Jorke Grotenhuis, Mark-Jan Nederhof,
Stéphane Polis, Serge Rosmorduc, Daniel A. Werning

Aug 11, 2021

Background documents

- [L2/17-112R](#) summarizes and finalizes ideas from earlier proposals, and is the most self-contained description of the existing nine control characters for Ancient Egyptian hieroglyphic text [\[14\]](#).
- [L2/18-236](#) discusses details of the syntax [\[23\]](#).
- [L2/19-331](#) discusses details of the semantics [\[24\]](#).
- [L2/20-176](#) discusses recent progress with implementation in OpenType [\[13\]](#).

Summary

Ancient Egyptian hieroglyphic texts consist of signs (hieroglyphs) arranged next to one another, above one another, or in a range of other spatial arrangements. Text can be horizontal (in rows), or vertical (in columns). Starting with Unicode 12 (March 2019), there are nine control characters to express the relative positioning of signs, listed in [Table 1](#) together with their syntax.

The two most self-evident control characters are the VERTICAL JOINER and the HORIZONTAL JOINER, which arrange signs above one another and beside one another, respectively. The BEGIN SEGMENT and END SEGMENT are needed to nest horizontal and vertical arrangements. Such nested arrangements are relatively common in Ancient Egyptian inscriptions, but there are few parallels in other writing systems, and consequently, introduction of such controls was a significant novelty in Unicode.

Signs can also be overlaid. Some combinations of overlaid signs are very common, and exist as atomic code points in Unicode. Because overlaying is largely productive, it is not feasible to enumerate all possible overlaid arrangements, which motivated OVERLAY MIDDLE.

The four remaining control characters represent insertion of a sign, or of a nested group of signs, in the empty corner of a larger sign. The present proposal makes a case for one additional insertion control, revisits the encoding of cartouches and similar enclosing signs, and proposes controls for rotation and mirroring. We further consider encoding of damaged and incomplete text. The core of our proposal is found in [Table 2](#) and [Table 3](#). However, various additional controls have been under consideration. These are found in the sections labelled **(Option A)**, **(Option B)**, etc. [Feedback from the Script Ad Hoc Committee is sought on the core part of the proposal as well on these additional controls.](#)

1 Introduction

The design of the existing nine control characters in 2016/2017 followed three main principles:

U+13430	⋮	EGYPTIAN HIEROGLYPH VERTICAL JOINER
U+13431	*	EGYPTIAN HIEROGLYPH HORIZONTAL JOINER
U+13432	⌈	EGYPTIAN HIEROGLYPH INSERT AT TOP START
U+13433	⌋	EGYPTIAN HIEROGLYPH INSERT AT BOTTOM START
U+13434	⌌	EGYPTIAN HIEROGLYPH INSERT AT TOP END
U+13435	⌍	EGYPTIAN HIEROGLYPH INSERT AT BOTTOM END
U+13436	⌎	EGYPTIAN HIEROGLYPH OVERLAY MIDDLE
U+13437	(EGYPTIAN HIEROGLYPH BEGIN SEGMENT
U+13438)	EGYPTIAN HIEROGLYPH END SEGMENT

```

fragment ::= ( group )+
group ::= vertical_group | horizontal_group | basic_group
vertical_group ::= ver_subgroup ( ⋮ ver_subgroup )+
ver_subgroup ::= horizontal_group | basic_group
horizontal_group ::= hor_subgroup ( * hor_subgroup )+
hor_subgroup ::= ( vertical_group ) | basic_group

basic_group ::= core_group | insertion_group
insertion_group ::= core_group insertions
insertions ::= ⌈ in_group [ ⌋ in_group ] [ ⌌ in_group ] [ ⌍ in_group ] |
              ⌋ in_group [ ⌌ in_group ] [ ⌍ in_group ] |
              ⌌ in_group [ ⌍ in_group ] |
              ⌍ in_group
in_group ::= ( vertical_group ) |
            ( horizontal_group ) |
            ( insertion_group ) |
            core_group
core_group ::= flat_hor_group ⌎ flat_ver_group | sign
flat_hor_group ::= ( sign ( * sign )+ ) | sign
flat_ver_group ::= ( sign ( ⋮ sign )+ ) | sign

```

Table 1: The existing control characters for Ancient Egyptian and their syntax.

- (1) Inspiration can be gleaned from existing forms of encoding of hieroglyphic text, but one cannot blindly adopt their primitives in Unicode.
- (2) The Unicode controls must have a straightforward meaning independent from the platform, font, and rendering engine.
- (3) There should be as few primitives as possible.

The first principle is motivated by the awareness that existing forms of encoding of hieroglyphic text, in particular the Manuel de Codage (MdC) [6], were designed for the purpose of preparing *printed* publications. A typical use case is as follows: For a hieroglyphic or hieratic text of interest, a specific tool, such as Glyph [5] or JSesh [30], is used to produce a transcription, using the primitives of the MdC, with custom settings of the tool, possibly extended with custom glyphs. One then lets the tool export an image, which can be embedded in a scientific article. After this, the encoding can be discarded.



















U+13439		EGYPTIAN HIEROGLYPH INSERT AT MIDDLE
U+1343A		EGYPTIAN HIEROGLYPH BEGIN ENCLOSURE
U+1343B		EGYPTIAN HIEROGLYPH END ENCLOSURE
U+1343C		EGYPTIAN HIEROGLYPH BEGIN WALLED ENCLOSURE
U+1343D		EGYPTIAN HIEROGLYPH END WALLED ENCLOSURE
U+1343E		EGYPTIAN HIEROGLYPH MIRROR
U+1343F		EGYPTIAN HIEROGLYPH QUARTER FORWARD ROTATE
U+13440		EGYPTIAN HIEROGLYPH QUARTER BACKWARD ROTATE
U+13441		EGYPTIAN HIEROGLYPH HALF ROTATE
U+13442		EGYPTIAN HIEROGLYPH SHADE GLYPH TOP START
U+13443		EGYPTIAN HIEROGLYPH SHADE GLYPH BOTTOM START
U+13444		EGYPTIAN HIEROGLYPH SHADE GLYPH TOP END
U+13445		EGYPTIAN HIEROGLYPH SHADE GLYPH BOTTOM END
U+1342F		EGYPTIAN HIEROGLYPH V011D
U+101B1		EGYPTIAN HIEROGLYPH FULL BLANK
U+101B2		EGYPTIAN HIEROGLYPH QUARTER BLANK
U+101B3		EGYPTIAN HIEROGLYPH SQUARE BRACKET OPEN
U+101B4		EGYPTIAN HIEROGLYPH SQUARE BRACKET CLOSE

Table 2: The core of the newly proposed control characters for Ancient Egyptian, and related new characters.

The considerations for Unicode are very different however, with its emphasis on *interchange* of textual data, which leads us to the second of the above principles. A Unicode encoding of hieroglyphic texts should not include any element whose meaning changes significantly depending on the chosen tool, its custom settings, or the choice of font, as this makes interchange of the encoding impossible; see [22] for further discussion.

A case in point is the ‘&’ operator, which is widely used in existing MdC tools such as JSesh. This operator can be applied on two or more signs to obtain a particular relative positioning of those signs. The problem is that the exact relative positioning for a particular choice of signs is determined by the used tool, possibly by customizing its settings. For example, an expression of the form $A \& B$ could mean that sign A should be placed in the upper-right corner of sign B in one tool, but in another tool, or in the same tool with different custom settings, it could mean something entirely different, perhaps that A should be placed in the lower-left corner of B . Whereas this is harmless if the encoding is solely used to produce a printed publication and is then discarded, the ‘&’ operator precludes interchange of the encoding without losing its meaning, all the more as the spatial arrangement may affect the linguistic interpretation of the encoding.

The third principle is motivated by the significant challenges that Ancient Egyptian poses to common font technology such as OpenType. In particular, the productive use of controls to form deeply nested groups stretches the capabilities of such technology to its limits. One should therefore find a compromise between expressiveness and technical feasibility. For example, there should be primitives to distinguish such spatial arrangements as overlay and corner insertions, as satisfactory transcriptions cannot be obtained without them, but we would not aim to fine-tune scaling of signs or the distances between them. This also means that we cannot expect Unicode to replace tools such as JSesh, as the power of such tools is beyond the capabilities of common font technology. For faithful transcriptions of hieroglyphic texts that are to be included in printed publications, JSesh and other such tools will still be needed in the future.

```

fragment ::= ( group | singleton_group )+
horizontal_group ::= [ ] hor_subgroup [ ] ( * [ ] hor_subgroup [ ] )* |
                    hor_subgroup [ ] ( * [ ] hor_subgroup [ ] )* |
                    hor_subgroup ( * [ ] hor_subgroup [ ] )+
basic_group ::= core_group | insertion_group | blank | enclosure
insertions ::= [ ] in_group [ ] in_group [ ] in_group [ ] in_group [ ] in_group |
               [ ] in_group [ ] in_group [ ] in_group [ ] in_group |
               [ ] in_group [ ] in_group [ ] in_group |
               [ ] in_group [ ] in_group |
               [ ] in_group
core_group ::= flat_hor_group [ ] flat_ver_group | literal
flat_hor_group ::= ( literal ( * literal )+ ) | literal
flat_ver_group ::= ( literal ( : literal )+ ) | literal
literal ::= sign [ ] [ ] [ ] shading
blank ::= ( [ ] | [ ] ) shading
enclosure ::= [ opening ] inner_enclosure [ closing ]
inner_enclosure ::= [ ] ( group )* [ ] | [ ] ( group )* [ ]
opening ::= opening_delimiter shading
closing ::= closing_delimiter shading
singleton_group ::= delimiter shading
shading ::= [ ] [ ] [ ] [ ]

```

Table 3: The syntax of newly proposed control characters, as far as it differs from Table 1.

A fourth principle of lesser importance is:

- (4) Let each rendered text be expressible by only one encoding.

This principle is generally desirable in Unicode, as it simplifies text search, among other things. For many reasons however, this principle cannot be consistently applied on Ancient Egyptian. One reason is that some groups of signs that can be obtained using controls since Unicode 12 also exist as atomic code points since Unicode 5.2.

Given the gains made by applying the above principles, which resulted in a simple but powerful set of controls with well-defined meanings, a continuation in the same direction is advisable.

2 Middle insertion

The four existing corner insertion controls in Unicode were inspired by PLOTTEXT [36, 37], which has six related primitives, namely insertion in one of the four corners of a larger sign, insertion above the feet of a bird, and insertion in the middle of a larger sign. The first five correspond to the four insertion primitives in Unicode, with insertion above the feet of a bird unified with insertion in the lower-left corner. Table 4 provides examples, with their analogues in PLOTTEXT.

An earlier Unicode proposal [26, 27] also included a middle insertion, next to the four corner insertions, motivated by the corresponding primitive in PLOTTEXT, illustrated by Table 5 (a). A number of groups involving middle insertion exist as atomic signs, such as those in Table 5 (b). The phenomenon is highly

	PLOTTEXT	Unicode
	zA;/ra;	
	F4;t//;	
	w;t/;	
	D;n,d;	(:)
	A17;//Z2b;	
	w;t/;;/t;	

Table 4: Corner insertions.

	PLOTTEXT	Unicode
(a)	%010;/f/;	
(b)		
(c)		
(d)		
(e)		
(f)		

Table 5: Middle insertion, (a) the syntax in PLOTTEXT, and how it could potentially be encoded in Unicode, (b) examples of groups with middle insertion that exist as atomic signs in Unicode, (c) [20], (d) [16], (e) [12] examples from various sources and (f) the prospective encoding of middle insertion of a larger group in Unicode.



Table 6: Occurrence of *hwt* enclosure on the statue of royal scribe Youpa [38, Plate 22]; 19th dynasty.

productive however, and enumerating all such groups is not feasible. A small selection from a few sources is listed in Table 5 (c-e); see also Table 6.

As the examples show, a sign or group inserted in the middle of a bigger sign need not be strictly enclosed within the larger sign, and may even be outside the bounding box of the larger sign, analogous to corner insertion [24].

Objections against the middle insertion were raised in the context of the earlier proposals [26, 27], with the argument that implementation in existing font technology was going to be difficult. Recently



Table 7: Groups with corner insertion rendered by the OpenType implementation discussed by [13].

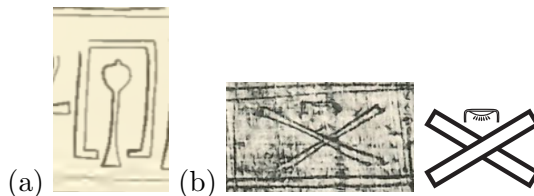


Table 8: Problems for middle insertion, (a) [17, p. 14], (b) [7, Taf. 2].

however, significant progress has been made with implementation of the four corner insertions in OpenType (Table 7), and middle insertion was found to be no harder to implement than the corner insertions. For all insertion primitives, the available space at the relevant position within a sign can be determined manually for individual signs, or can be determined semi-automatically [24]. One slight complication is that some common signs such as 𓏏 , 𓏑 and 𓏒 may need to be included in a font in a second, larger version, such as 𓏏 , 𓏑 and 𓏒 , for use with middle insertion. This is easy to realize within the existing implementations of Ancient Egyptian control characters in OpenType.

However, the last example of Table 5 (c) and the second and third examples from (d) pose technical challenges, even though they may look similar to the other examples and may not have been perceived as any different by the ancient scribes. The first issue is that the encoding becomes quite unwieldy, due to the need for several pairs of parentheses [] and [] and horizontal joiners * to connect subsequent subgroups. The second issue has to do with conversion between horizontal and vertical text to suit the purposes of publication. In particular, in grammar books, text direction is generally normalized to be horizontal. A *hwt* sign 𓏑 in vertical text enclosing several groups of signs underneath each other would be printed with those groups beside each other instead. The third and final issue is the difficulty of stretching out the shape of the enclosing sign depending on the dimensions of the enclosed groups. These considerations suggest an alternative encoding analogous to cartouches, as discussed in Section 3.

special cases

Inevitably, some cases will remain that cannot be handled perfectly with the four corner insertions and the one middle insertion. First, whether inserted groups ‘stick out’, as exemplified by some groups in Table 5 (e), is currently not encoded. One may argue that this is a shortcoming, as it does not allow us to distinguish between Table 8 (a) and the more usual form 𓏑 . This distinction may be felt to be relevant enough to epigraphy to merit encoding. One solution could be to use variation selectors to choose either the usual shape 𓏑 or a more square-shaped variation 𓏑 , the choice of which determines whether the inserted sign ‘sticks out’ of the bounding box: In the latter case, there is enough space inside the base sign to fully incorporate the inserted sign, while in the former case, the inserted sign inevitably needs to go partly outside the bounding box. An alternative is Option A below. A last resort may be atomic encoding.

Second, there are cases of insertion that are neither captured accurately by corner insertions nor by the middle insertion, such as Table 8 (b), which could be analyzed as ‘top insertion’. (One could additionally envisage ‘bottom insertion’, ‘left insertion’ and ‘right insertion’ for this sign; once more, see also Option A

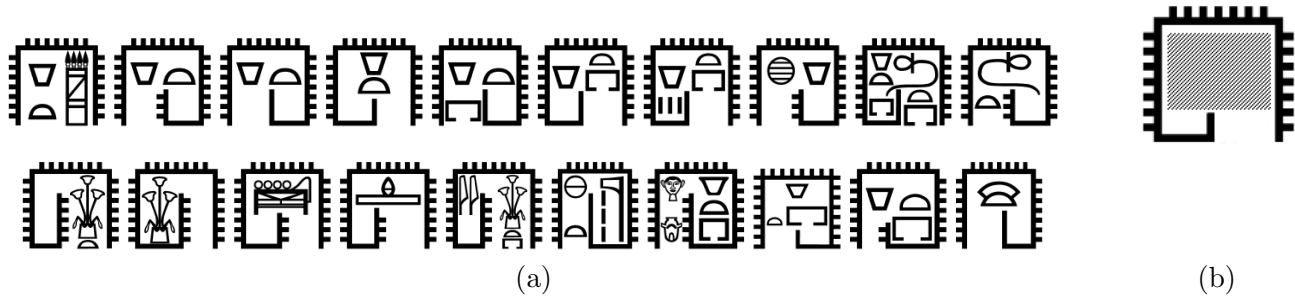


Table 9: (a) Special cases of middle insertion in variants of O013. (b) Single generic area for middle insertion.

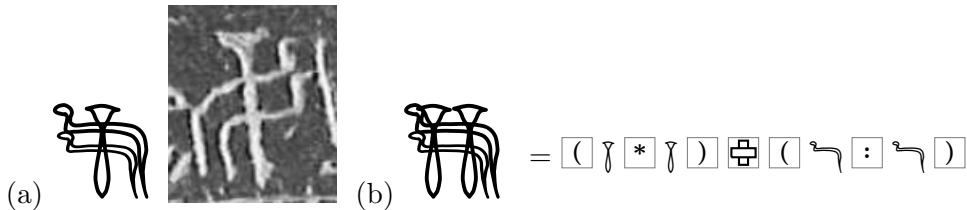


Table 10: Problems for overlays. For (a) see [18, p. 1], [15, pp. 103-104, Plate 29].

below). Any insertions in a sign such as \square would mean either ‘insertion into the left empty space’ or ‘insertion into the right empty space’. None of these are presently available as control characters.

Although these cases are rare, some thoughts are in order how to handle them. The most straightforward option is to use the middle insertion, interpreting ‘middle’ broadly as one particular empty space within the bounding box other than the four corners. There will be some cases where this does not result in the ideal rendering, but for most practical applications, an insertion in a slightly ‘wrong’ place may still be acceptable. Another option is to add a larger variety of insertion primitives at a later time. There will however be diminishing returns, with each added control character adding some burden on the design of fonts. Another option is atomic encoding, which should generally be the last resort.

Similar examples include the insertions into O013 illustrated in Table 9 (a). There are numerous special cases, some with a group inserted into the left half, some with a group inserted into the right half, some with a group inserted in the top half, and some with combinations of these. As no repertoire of controls can realistically be expected to faithfully capture all such cases, it seems best to opt for a shape of O013 in which there is a single large rectangle, indicated as the shaded area in Table 9 (b), which allows middle insertion of a single group.

Rare special cases generally deserve a pragmatic solution allowing them to be encoded with the existing controls and their syntax. Another example is the problem posed to overlays by Table 10. The issue is that in the past it was assumed that overlays are limited to being applied on one ‘flat’ horizontal group and one ‘flat’ vertical group (cf. Table 1). This assumption was intended to keep implementations simple. Here however, one may argue that the two cobras are arranged by corner insertion. Rather than generalizing the syntax of overlays, a better solution may be to analyze the examples in Table 10 such that there is a flat vertical group consisting of two cobras, with ‘kerning’ to leave as little space in between as possible. This is in keeping with the observation that the flat horizontal and vertical groups in overlays are as a rule squeezed together. Moreover, the original inscription in (a) looks less like corner insertion and more like two equally sized cobras that are shifted together.

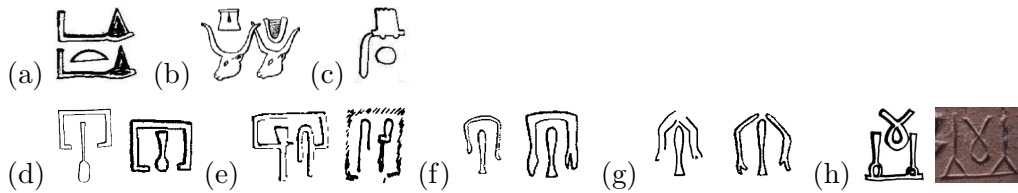




























Table 11: The bottom two signs in (a) can be composed by ; BM EA 143 [10, p. 110]. Similarly (b), BM EA 1242 [17]. Group (c) can be handled by ; BM EA 571 [10, p. 77]. In (d-g), a meaningful distinction can be made by the choice of either  or ; [17]. In (h), a meaningful distinction can be made by the choice of either  or ; BM EA 567 [10, p. 120] and BM EA 903.

(Option A) two side insertions

Our very first proposal in 2016, L2/16-177, had 9 insertion controls, namely four corner insertions, one middle insertion and four *side* insertions. In the face of heavy opposition against the perceived overwhelming number of such controls, later proposals were simplified to only include the four corner insertions, as these exist in PLOTTEXT and are easiest to justify. One of the four side insertions, the ‘insert at start’ , which is mainly needed to insert signs above the feet of a bird sign, and which also has an analogue in PLOTTEXT, was merged into ‘insert at bottom start’ .

In our above proposal of the middle insertion, we have once more tried to capture as many cases as possible, to reduce the need for further controls. However, in effect that means we have proposed the use of the middle insertion even for cases that geometrically are not exactly insertion in the middle of the base sign, and that could be more appropriately expressed by two of the side controls from our original proposal. For example,  could more suitably be encoded by an ‘insert at top’ control  and  could more suitably be encoded by an ‘insert at bottom’ control ; for similar groups see Table 8 (b) and Table 11 (a-c). The middle insertion  could then be restricted to cases where the inserted signs are properly inside the shape of the base sign.

This would also give us a more principled solution to distinguishing between , encoded with , and , encoded with . A similar distinction exists for  versus , and even for multiple inserted signs, such as  versus ; cf. Table 11 (d-h).

The case for introducing  and  is less compelling. We can continue to use  rather than  for ‘insert above the feet [of a bird]’. From among the current Unicode repertoire of 1071 signs, only \times , \otimes and \int have a shape that could plausibly allow for an ‘insert at end’ .


Summarizing the advantages of adding  and .

- More expressive power, solving various open problems with ‘insert at middle’ described earlier in this document.
- Owing to the experience with implementing the corner insertions and middle insertion, it is already clear how to implement side insertions.

Disadvantages:

- Need for two additional controls.

3 Cartouches and similar enclosures

A *cartouche*  encloses a name (in fact two out of the five names) of a king. The enclosed name can be lengthy, and consequently the cartouche has, in principle, unrestricted width (or height in the case of

	PLOTTEXT	MdC/JSesh	RES	Unicode
	%Z1 <i>etc.</i> %Z2	<- <i>etc.</i> ->	cartouche(<i>etc.</i>)	
	%Z3v <i>etc.</i> %033 (only in vertical text)	<S- <i>etc.</i> ->	serekh(<i>etc.</i>)	
	(not available)	<F- <i>etc.</i> ->	inb(<i>etc.</i>)	
	%Z3 <i>etc.</i> %Z3	<h1- <i>etc.</i> -h1>	rectangle(<i>etc.</i>)	
	%Z3 <i>etc.</i> %06	<h1- <i>etc.</i> -h3>	Hwtcloseover(<i>etc.</i>)	

Table 12: Encoding of cartouches and related signs, in PLOTTEXT, Manuel de Codage (MdC) [6, 30], RES [25], and as proposed in Unicode.

vertical text). There are a number of other enclosing signs whose dimensions vary in the same manner, as illustrated in Table 12. These also include the *hwt* sign discussed earlier, and with the same justification as before, we may want to avoid encoding a cartouche enclosing a long name as: , then a sequence of groups connected by *, and lastly), as this would pose significant technical challenges.

Traditionally, cartouches and related signs have been encoded by a pair of symbols, delimiting the start and the end of the enclosed text; see Table 12 for examples of encodings in PLOTTEXT and MdC, which contrast with the syntax of RES.

Having pairs of delimiters has two disadvantages:

- Two code points are needed for each enclosing sign.
- There is redundancy in the notation, or alternatively, it is unclear what the meaning should be of the combination of an opening delimiter and a closing delimiter that do not match.

One may argue that the free combination of opening and closing delimiters, as possible in MdC and PLOTTEXT, allows some flexibility, and may even help to keep the number of operators down. For example, by having three opening delimiters and and three closing delimiters and , one can form nine enclosing signs, such as and . However, some of these, such as , are uncommon. Moreover, it is not clear how an entirely implausible combination such as and should be rendered, and one may consider such a combination of two unrelated delimiters to be syntactically invalid.

Having pairs of delimiters, as opposed to an encoding using middle insertion, does have a number of clear advantages however:

- Fewer *occurrences* of control characters are typically needed.
- The enclosed text can be given a dedicated syntax, as sequence of groups, which are rendered beside one another or above one another, depending on whether the text direction is horizontal or vertical, respectively.
- Implementation is simplified, by a dedicated mechanism to render the enclosing sign, possibly with the enclosed text on the baseline, just like text would be that is not enclosed, and the top and bottom borders of the enclosing sign can be rendered above and below the enclosed text, within what would otherwise be the space between lines.
- The implementation can be further simplified by assuming that an enclosing sign is not part of a larger group.

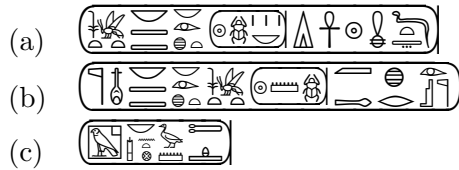


Table 13: Nested enclosures: (a) the original is vertical right-to-left [3, Plate V], (b) the original is vertical left-to-right (Tomb of Thutmosis III, Safari Afrika, <https://images.app.goo.gl/9xS7mFhtakuVEA6v8>), (c) *hwt* enclosure within cartouche [2, p. 194].

opening	closing
U+13258 O006A	U+1325B O006D
U+13259 O006B	U+1325C O006E
U+1325A O006C	U+1325D O006F
	U+13282 O033A
U+13379 V011A	U+1337A V011B
(**) V011D	U+1337B V011C
U+13286 O036A	U+13287 O036B
U+13288 O036C	U+13289 O036D

Table 14: List of code points since Unicode 5.2 (2009) that could act as opening and closing delimiters, and one desired additional such delimiter marked with ‘(**)’.

Not allowing an enclosure to be part of a larger group in particular excludes e.g. nested cartouches as in Table 13 (a-b), but such writings are very rare, so that this limitation is unlikely to hinder common usage.¹ The writing in (c) can be easily realized by virtue of having an atomic code point; an alternative solution could otherwise have been a middle insertion.

In the past few years, pairs of delimiters have already been used in some prototype implementations to render cartouches and similar enclosing signs, using the code points listed in Table 14. An additional opening delimiter V011D is needed to represent a mirrored cartouche; cf. Table 16. This is distinct from the independent hieroglyph V011

Before fixing the syntax of the use of the delimiters, a few issues deserve consideration. First, the delimiters and of cartouches also exist as separate, stand-alone signs in hieratic. Users who transcribe hieratic in Unicode will generally prefer that pairs of and are *not* interpreted as delimiters of ‘full-form’ cartouches that enclose text. Also signs such as have stand-alone occurrences, as in Table 15. Further note that the stand-alone delimiters are different from normal independent signs in that their orientation should depend on the text direction; they are rotated by a quarter turn in vertical text. Because of this, stand-alone occurrences of these signs must form a group (in the sense of Table 1) on their own, without

¹The outer occurrence of a nested cartouche often encloses the entire text, or an entire line of text, as in the case of Table 13 (a). Other examples include a pair of castanets from the Tutankhamun excavations, [http://www.griffith.ox.ac.uk/gri/carter/620\(13\)1-c620\(13\)1.html](http://www.griffith.ox.ac.uk/gri/carter/620(13)1-c620(13)1.html), and the sarcophagus of Hatshepsut, <https://collections.mfa.org/objects/130720?image=8>.

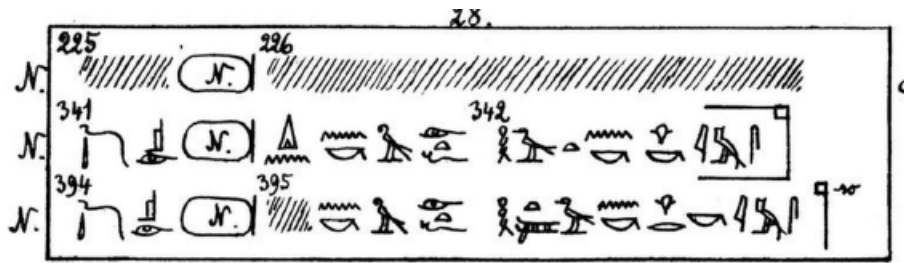


Table 15: Partial enclosure and stand-alone delimiter] [33, p. 12].



Table 16: Mirrored cartouche; stela of Intef (Leiden V 6) [21, 190-192 (Nr. 56), line 1].

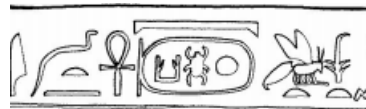



















Table 17: Complex enclosure; stela of Ity And Iuri (British Museum EA586) [10, p. 90].

further joiners, which becomes a singleton_group in Table 3.

One should be aware that O033  is also an independent hieroglyph, used without a matching opening delimiter, as determiner or logogram [16, p. 731]. In this use, we would not wish the sign to rotate spontaneously upon a change of text direction, which sets it apart from O033A .

A different issue is an enclosing sign of which the closing or opening is damaged (see also Section 5), which could be printed as for example  or . Here we would need a mechanism to signal the start or the end of the enclosing shape without a delimiter. The most economical way to deal with all of the above is to introduce two new characters  and  that are solely responsible for drawing the connecting lines of an enclosure. If used in combination with, for example,  then this creates a full-form enclosing cartouche, but  or  or both may be missing. If  or  are used without  and , then they act as stand-alone signs. The characters  and  must always occur in pairs to ensure an unambiguous syntax. To ensure unambiguity, the tokenizer is further required to analyze an **opening_delimiter** immediately preceding  to open an enclosure, and a **closing_delimiter** immediately following  to close it; see Table 3.

With the proposed encoding, we get for example:

$$\begin{aligned}
 \textcircled{\uparrow} &= (\uparrow \textcircled{\uparrow} \uparrow \uparrow \downarrow) \\
 \textcircled{\uparrow} &= (\uparrow \textcircled{\uparrow} \downarrow \downarrow \square \square \square \square) \\
 \textcircled{\uparrow} &= (\square \square \square \square \square \uparrow \uparrow \downarrow) \\
 \textcircled{\uparrow} &= (\square \square \square \square \square \uparrow \downarrow \square \square \square \square) \\
 (\textcircled{\uparrow}) &= (\textcircled{\uparrow})
 \end{aligned}$$

The characters for blanks and shading are discussed in Section 5.



Table 18: A ‘vertical’ cartouche in horizontal text; [17, Plate VI]

The pair and can be used for cartouches, serekhs and *hwt* enclosures. We need a separate pair of characters and for walled enclosures. It is conceivable that more such characters are needed in the future, but what we described fails to cover only extremely rare cases of enclosures, such as the ‘heaven’ sign above the cartouche in Table 17, where both shapes are stretched out together to the full length of the enclosed text, so that they could potentially be analyzed as a single complex enclosing shape.

middle insertion versus pairs of delimiters

It is tempting to *disallow* use of middle insertion (Section 2) where the same appearance can also be achieved using an enclosure as above. We would hesitate to impose that restriction at this time however, as middle insertion has a number of advantages over enclosures.

First, the delimiters of an enclosure are interpreted according to text direction, that is, the delimiters are rotated by a quarter turn if the direction is vertical. Connected to this, we assume that the implementation in say OpenType does not require remembering the opening delimiter until the moment when the closing shape is being drawn. However, it is generally undesirable that a simple composition like would turn into upon a change of text direction, which would unavoidably be the case if pairs of delimiters are used. A related issue is the existence of ‘vertical’ cartouches in horizontal text, as in Table 18, where use of the middle insertion, rather than a pair of delimiters, would prevent an undesirable normalization to a horizontal cartouche.

Second, we would not expect *nested* enclosures to be implemented, of the kind exemplified in Table 13 (a-b), as this would require considerable effort to handle a case that is very rare. However, corner and middle insertion would normally be available within enclosures, so that for example Table 13 (c) could be encoded in any case, even if were not an atomic code point.

Allowing say to be expressed using middle insertion, next to encoding as enclosure, does not require additional effort, beyond the implementation of middle insertion in general.

In practice, most users may prefer middle insertion over enclosure if only one small group is enclosed. If several groups are enclosed, then middle insertion is not applicable.

Next to O006 we would need graphical variants with the square in the other three corners. We will leave this to the work on the sign list that is going on in parallel.

4 Mirroring and rotation

A small number of signs change their meaning when mirrored, becoming thereby essentially distinct signs. For example, is among other things a logogram in *ju*, ‘come’, whereas is determinative in words related to ‘going backwards’. The majority of signs however can be mirrored for reasons of symmetry and of interactions between the animated hieroglyphs, frequently in cartouches as in Table 19 (a-b). A number of signs that represent inanimate objects and that have no clear ‘front’ or ‘back’ can be arbitrarily mirrored even outside cartouches; cf. Table 19 (c). In the Thesaurus Linguae Aegyptiae² (TLA) we counted over 230

²<http://aew.bbaw.de/tla/>

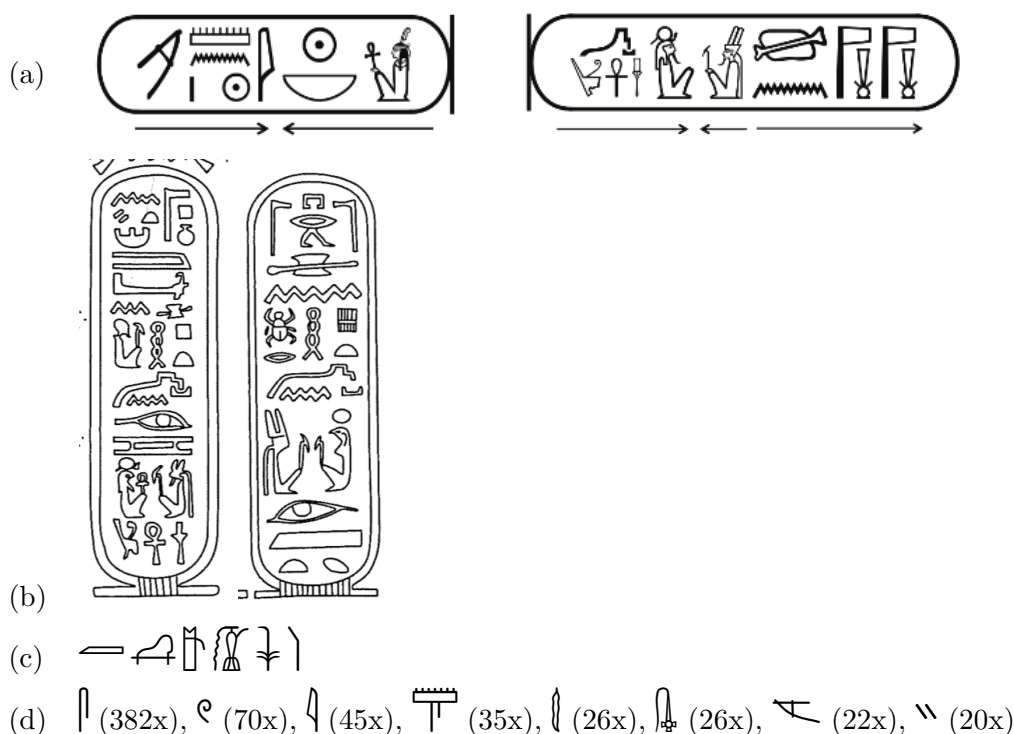


Table 19: (a) Mirroring for reasons of symmetry [28, p. 333], and (b) [4, p. 34 and p. 66], (c) a selection of signs that appear mirrored in [34] relative to their representation in Unicode, (d) the signs most frequently mirrored in the TLA.

distinct signs that were mirrored at least once. The most frequent ones are listed in Table 19 (d), excluding signs that are not in the current Unicode repertoire.

Introducing an additional code point for each sign that has been attested to have been mirrored in some text would be problematic, as it appears that a fair portion of signs, perhaps as much as one tenth, has a known occurrence in mirrored form.

Similarly, a small number of signs change their meaning when rotated by an eighth, a quarter or half a turn, becoming thereby essentially distinct signs. Cf. ⤵, ‘mace’, versus ↘, ‘smite’, and ⤵, ‘wall’, versus ⤵, ‘tilt’. A great number of signs however can be rotated for aesthetical reasons, especially if their shape is narrow and tall, or low and wide, so they may fit better into the composition of a group. This means that the rotated forms are mere graphical variants, and introducing additional code points would be problematic. At the same time, using the wrong orientation of such a sign yields an unsatisfactory composition, which moreover typical users would consider to be an incorrect representation of the original inscription; cf. Table 21 (a).

One strong piece of evidence that signs could be rotated for purely aesthetical reasons is found in Table 20. More examples of rotation are listed in Table 21 (b-e). In the TLA, about 80 signs occur in rotated form, some with more than one rotation, and some in combination with mirroring.

Most frequent seem to be rotations by a quarter turn clockwise of low and wide signs, and rotations by a quarter turn counter-clockwise of narrow and tall signs (assuming a left-to-right text direction). Other rotations exist however, such as rotations by half a turn, which for a portion of signs does not change the meaning. It is further worth noting that signs such as ⤵ (‘harpoon’) and ⤵ (‘platform’) are rotated *despite*, rather than because of, their iconic value.

The proposed solution is introduction of one control character ⇄ for mirroring and three control char-



Table 20: Tomb of Horemheb. Compare the red and black text of the left column: the scribe/draughtsman rotated ☞ for the final version of the layout, presumably as that would fit better in the composition.

acters ☞ , ☜ , ☛ for rotation by a quarter turn clockwise and counter-clockwise, respectively, (assuming left-to-right text; for right-to-left text, the direction of rotation is reversed) and rotation by half a turn. We assume that if a sign is both mirrored and rotated, then mirroring is done before rotation. For example, ☞ ☜ is rendered as ☞ .

Implementing the mirroring control has a relatively small cost in terms of the size of fonts. The reason is that mirrored copies of all signs need to be included in any case in order to render both left-to-right and right-to-left text. For rotation however, one can limit implementation to those signs that are known to occur frequently in rotated form, or that could reasonably be expected to occur in rotated form, such as inanimate objects that are low and wide or narrow and tall. Lists of signs for which certain rotations are assumed to be implemented can be maintained in a central place, analogous to the tables mentioned by [13].

5 Text-critical marks

In the corpus of Ancient Egyptian hieroglyphic inscriptions, undamaged and complete texts from beginning to end are the exception. The current set of Unicode characters is inadequate therefore to express the great majority of texts and parts of texts, on stone fragments, potsherds and snippets of papyrus.

For encoding an incomplete text, we first need to be able to indicate surfaces within an inscription where signs are no longer legible. Such a surface is normally drawn as a rectangle, filled by a light gray pattern, or by drawing diagonal lines across the surface, which is known as *shading* or *hatching*. For typical users, it is highly informative to know how large such a surface is, to make an educated guess about the signs that are lost. Furthermore, signs occur in groups, and the relative size of the shaded area can significantly affect the scaling and positioning of the remaining signs. This suggests the need to form shaded areas of different

	best achievable	original
(a)		and

(b) (21x), (20x), (7x), (6x), (5x), (4x)

(c) (18x), (18x), (5x), (2x)

(d) (4x), (2x), (2x), (2x), (1x)

(e) (4x), (3x), (2x), (1x), (1x)

Table 21: (a) Rotated signs within a group from [34], preceded by the best achievable encoding at present. (b) Some of the most frequent clockwise quarter-turn rotations in the TLA, (c) the same counter-clockwise, (d) the same half a turn. Omitted are signs not in Unicode and rotations in combination with mirroring. (e) A selection of signs in the TLA that are rotated by an eighth turn or less, clockwise or counter-clockwise.

dimensions. Examples are found in Table 22 (b), (c), (e), (f) and (j).

A second requirement is to be able to indicate that extant signs are damaged or partly illegible. This is once more done by shading, but now the shading overlays the affected signs, as exemplified once more by Table 22. Where this is combined with a pair of square brackets, this generally indicates that the enclosed shaded signs are a reconstruction, as opposed to being individually discernible.

It is important to be able to indicate that a part of a sign is damaged. For example, suggests that the actual sign may be or or , and suggests that the actual sign may be or , which are two different signs with non-overlapping meanings, most easily distinguishable by the shape of the tail.

A convenient granularity for shading of a sign divides its bounding box into four quarters. If we were to indicate the pattern of shading of a sign by a single post-modifying character, then that would require 15 such characters . A more practical solution is to allow up to 4 post-modifying characters per glyph, denoted here by . If all four are used, then this shades the entire glyph. In the naming, we use ‘START’ and ‘END’ rather than ‘LEFT’ and ‘RIGHT’, because hieroglyphic text can be written left-to-right or right-to-left, with mirrored signs and mirrored composition of groups, so that ‘START’ corresponds to either ‘left’ or ‘right’ depending on the text direction; a similar naming was adopted in the past for the four corner insertion controls.

In addition, we may want to add square brackets as in Table 22 (a), (d), (g) and (h), and in Table 23 without shading. These brackets would ideally take up no additional space as far as the scaling and positioning of signs within groups is concerned. The special behaviour of the brackets within groups of hieroglyphs motivates two dedicated code points and . These can precede or follow a `basic_group` (in the simplest case a single sign), or precede or follow a `vertical_group`, as specified in Table 3. An example is:

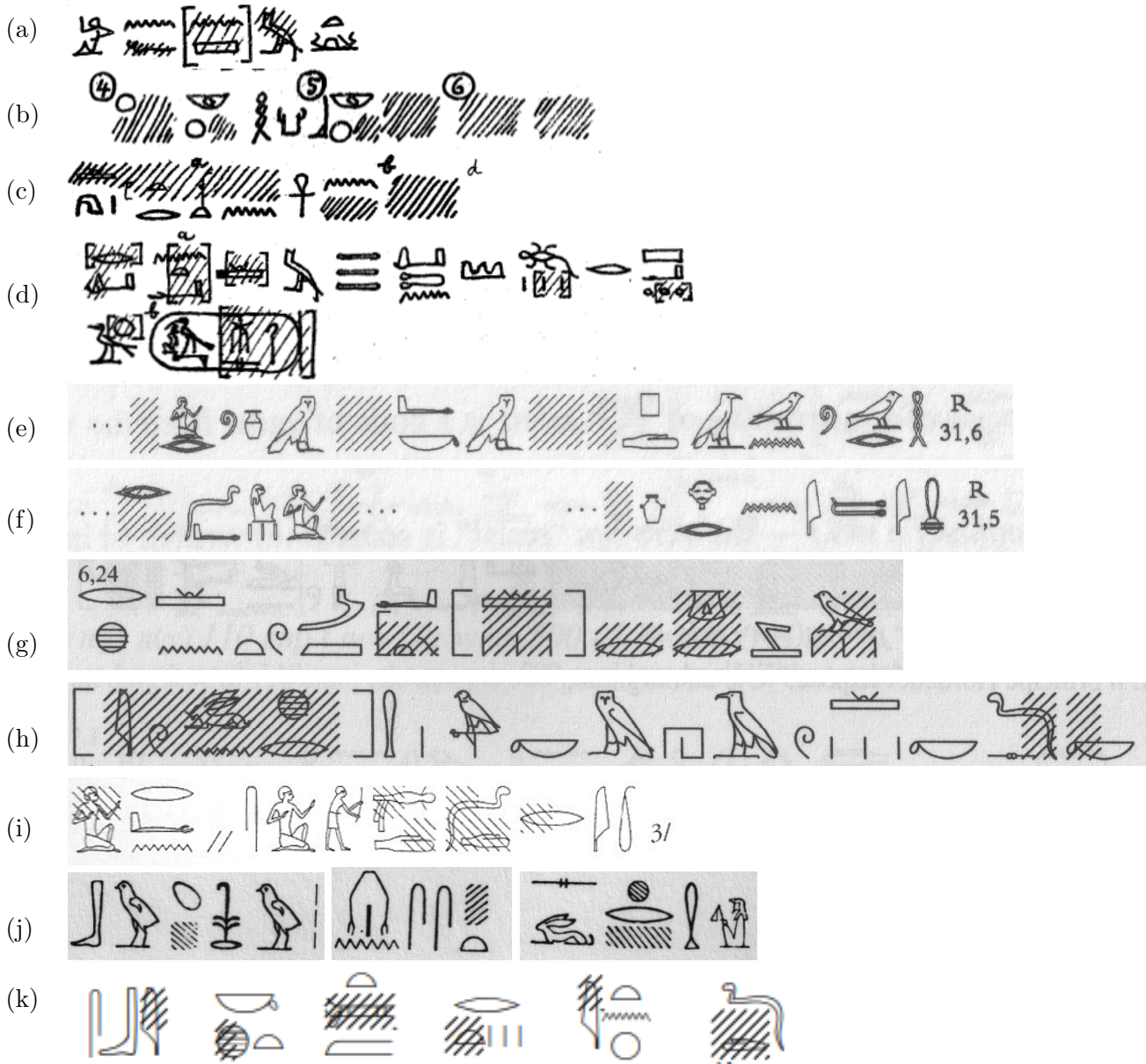


Table 22: Examples of shading: (a-c) [35], (d) [32], (e-f) [1], (g-h) [9], (i) [11], (j) [29], (k) [39].

$$\left[\begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \end{array} \right] = \left[\left[\left(\leftarrow \right) : \leftarrow * \right) \right] \right]$$

A third requirement is to be able to indicate *absence* of signs on some surface area of an inscription, which is different from damage to the surface area. There are several cases where this is needed, in transcriptions of both hieroglyphic and hieratic texts. A non-exhaustive list is:

1. A blank in the middle of a text was left in order to eventually fill in a name or a date, which never happened, as in phrases of the form “This book belongs to []” or “In year 3, [] month of the inundation, day []”.
2. A cartouche \square can contain blank space as placeholder for a king’s name, as in Table 24 (b). This is very different from the independent cartouche sign U+13377 \square .

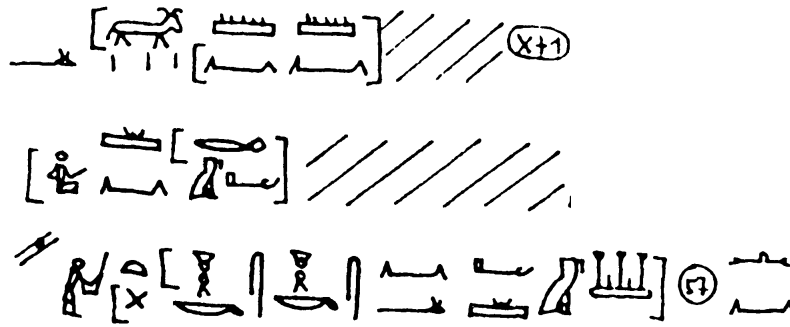


Table 23: Square brackets in transcriptions [19].

3. A scribe copying a text left a corresponding amount of blank space where the original was damaged, as in Table 24 (c), where the blank space corresponds to the missing grammatical subject in “[...] Ra! [] says to Ra: [...]”. The name of a goddess is expected in the light of the feminine suffix pronoun.
4. Hieroglyphs on walls of tombs were first drawn in ink and later carved. Sometimes the artists forgot to carve a sign and the ink faded away or was painted over with a background color. This resulted in a blank in place of a sign, as in Table 24 (d).
5. An empty space refers to the object or material on which the inscription is written, as in the enigmatic spelling near the centre of Table 24 (a) of ‘loving myrrh’, with *mr* (‘loving’) written with a blank, referring to the palette made of softwood, which is also *mr* in Ancient Egyptian.

In all these cases, the exact size of a blank surface is highly suggestive of its interpretation in context. Relying on existing space or blank symbols in Unicode is little effective, as we need to rely on definite sizes relative to 1 em (the height of the unscaled ☞ in the font). Furthermore, the blank signs are specifically expected to interact with the horizontal and vertical joiners and other control characters of Ancient Egyptian, and would be subject to the same rules of scaling as hieroglyphs would. A pragmatic choice is to introduce two characters for blank surfaces of sizes 1 em × 1 em and 0.5 em × 0.5 em.

Note that U+2001 (em quad) and U+2000 (en quad = 0.5 em quad), which by definition are two-dimensional, cannot be re-used for our purposes, as we need blanks that are subject to scaling.

As the two prospective hieroglyphic blanks largely behave like normal signs, they can be combined with ◻ ◻ ◻ ◻ to obtain shaded rectangles that represent surfaces within an inscription where signs are no longer legible. As explained in Appendix A, such rectangles need not be square.

See Table 25 for examples of encodings in PLOTTEXT, MdC, RES and prospectively Unicode. The extant documentation of PLOTTEXT does not provide a fully unambiguous description of the syntax, but it appears that shading is available for individual signs, by enclosing them within a pair of %S, as well as for parts of a group, by using an overlay with a shading symbol. There are four such shading symbols %S1, %S2, %S3, %S4, and there are four blank signs %B1, %B2, %B3, %B4, two of which match the prospective blank signs in Unicode, and two more are a narrow and tall blank and a low and wide blank. Similarly, the MdC allows shading of individual quarters of glyphs, as well as shading of quarters of groups. It has four individual shading symbols //, /, h/, v/, of the same dimensions as those of PLOTTEXT. It has two blank signs . . and . comprising a full square and a quarter square, respectively. RES allows shading of glyphs and of the spaces between glyphs, to unbounded granularity, by which an appearance can be obtained that is very close to shading of whole groups. (In Table 25 (d), the top half of the inter-glyph space is shaded by adding [t] after the colon.)

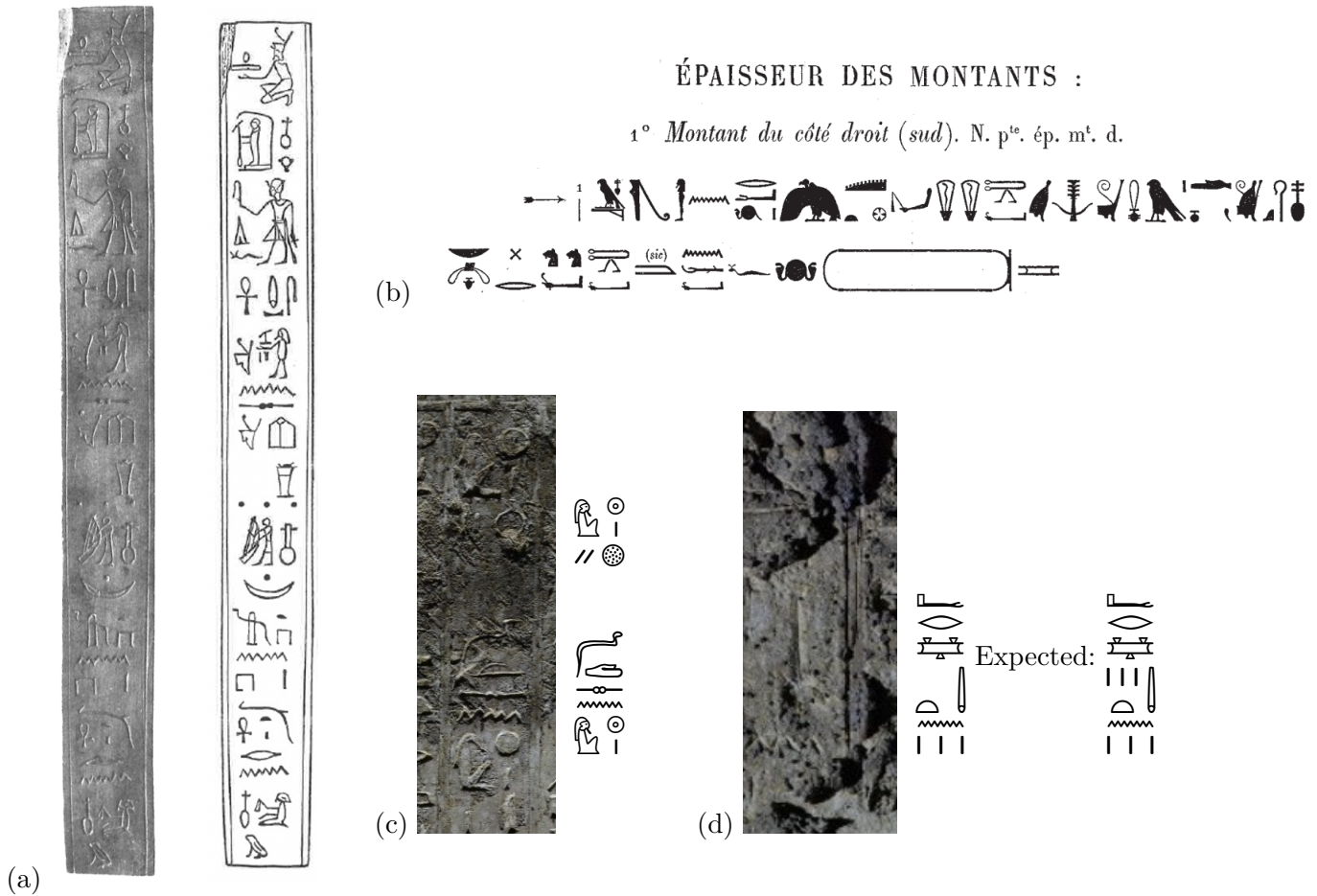


Table 24: Blank spaces in inscriptions: (a) from the scribal palette Frankfurt Liebighaus IN 1899 [31, p. 321], (b) temple of Dendara [8], (c) Theban Tomb 33, Book of Caverns, col. 1287, Hb. 103,21, photo DOI 10.17171/2-8-358-2, [39, p. 251], (d) Theban Tomb 33, Book of Caverns, col. 809, Hb. 76.37,Z, photo DOI 10.17171/2-8-334-2, [39, p. 183].

It should be pointed out that shading of individual signs was not available in earlier variants of the Manuel de Codage. Where typeset publications only have group shading, as for example in Table 22 (g-h), this is almost always due to limitations of the used software, in this case Glyph for Windows, which does not implement sign shading. However, for many applications, group shading is far more appropriate than sign shading. For the sake of presentational convenience, this section has so far only discussed how sign shading can be introduced in Unicode. We will later return to the strong need for introducing group shading, next to or as alternative to sign shading, in Options G, H, and I below.

(Option B) whole sign shading as extra control

The four controls that shade the four quarters of a sign will most often be used together to shade the entire sign. A suitable input method may offer a single key or single button to shade all quarters together, saving the user from the burden of selecting each of the quarters individually. Nonetheless, having a single control to shade the entire sign could be convenient.

Another argument is that some widely used shaping engines impose a limit on how many characters can together form a group. Through experimentation, we found that the DWrite shaper has a limit of 31






















	PLOTTEXT	MdC/JSesh	RES	Unicode
(a) 	N35,%S I9 %S (?)	n:f\shading1234	n:f[shade]	 :    
(b) 	G17+%S3 (?)	m\shading12	m[t]	  
(c) 	N35,%S4 (?)	n://	n:empty[shade]	 :    
(d) 	"t,r"+%S3 (?)	t:r#12	t[shade]:[t]r	 :    (?)












Table 25: Encoding of shading; for (d) see Option H below.

characters. To illustrate how restrictive this is, a group of 6 signs and 5 joiners in which all signs are to be shaded by four quarter sign shading controls would require 35 characters, which is already beyond this limit.

Hence, the advantages are:

- It allows us to form much larger shaded groups with current font technology.
- In the absence of specialized input methods, input of shaded signs requires fewer characters to be entered.

Disadvantages:

- One more code point, and slight added complexity in the implementation.
- An encoding such as   would result in the same rendering as     , unless we introduce a syntactic restriction to forbid the full sequence of characters    .

(Option C) whole sign shading as only sign shading control

One could also introduce whole sign shading, as in Option B, and omit quarter sign shading.





Advantages:

- Saves four controls for quarter sign shading, and implementation becomes easier.
- The problem of existence of two distinct encodings with the same rendering goes away.
- Users typically search for damaged versus undamaged occurrences of signs, and may be less interested in how much of a sign is damaged. Hence, for the purposes of search at least, sign shading at the granularity of quarters is dispensible.





Disadvantages:

- Users cannot encode that only a part of a sign is damaged and that the remainder is still visible.

(Option D) two more blanks

Next to the two square blanks  and , one could introduce two more blanks  and , of sizes 0.5 em × 1 em and 1 em × 0.5 em, respectively, as they exist in PLOTTEXT (unlike MdC, which only has two blanks).



Advantages:

- Convenience, avoiding the need for joiners as in  :  and  * .
- Just as easy to implement as ordinary signs.

Disadvantages:

- Two more code points.
- Two different encodings are rendered the same.
- The aspect ratio of the blank surface in the rendering may not correspond to the aspect ratio of the blank in isolation, due to the scaling and positioning of groups; see also Option F and Appendix A.















(Option E) independent shade characters

Users may wish to search for groups of signs specifically with presence or absence of damaged areas, which is different from searching for missing signs represented by blanks. There is therefore a case to be made for having independent shade characters, in place of shaded blanks. A pragmatic choice would be to have two such shade characters  and  of sizes 1 em × 1 em and 0.5 em × 0.5 em, respectively, next to the two blanks of the same dimensions.

Advantages:





- Convenience, requiring shorter character sequences.
- Makes formulating search queries easier by a completely separate encoding of independent shade characters versus shaded blanks.

Disadvantages:

- Two more code points.
-  is rendered the same as     . This problem goes away *if* one could introduce a syntactic restriction that blanks cannot be shaded, i.e. sequences such as      and    would be declared invalid. This syntactic restriction may or may not be enforced by fonts.

[Feedback from the Script Ad Hoc Committee how to proceed for this matter specifically would be much appreciated.](#)

(Option F) two more shade characters

Next to the two square shade characters  and  from Option E, one could add two more shade characters  and  of sizes 0.5 em × 1 em and 1 em × 0.5 em, respectively, as they exist in MdC and PLOTTEXT.³ The resulting four shade characters would match the four blanks if Option D is followed.

Advantages:

³The original intended uses of the four shading operators in MdC, apparent from e.g. [6, p. 21], include overlay of a hieroglyph with a shading shape. For example, overlay of a 0.5 em × 1.0 em hieroglyph and a 0.5 em × 1.0 em shading shape results in the sign being completely shaded. If the shading is smaller than the hieroglyph in absolute terms, then only part of the hieroglyph is shaded, and if the shading is larger, then some portion of the shading is rendered outside the bounding box of the hieroglyph. Here the need for several shapes and sizes of shading is clearer than it is for Unicode encodings of Ancient Egyptian.

- Convenience, avoiding the need for joiners as in ◻ ◻ : ◻ ◻ and ◻ ◻ * ◻ ◻.
- Fairly easy to implement next to the two square shade characters from Option E.

Disadvantages:

- Two more code points.
- Two different encodings are rendered the same.
- The aspect ratio of the shaded surface in the rendering may not correspond to the aspect ratio of the chosen shade character in isolation, due to the scaling and positioning of groups and the prospective semantics of shading in Unicode, which has the potential to confuse users. See also Option D and Appendix A.

(Option G) whole group shading

One may introduce a control ◻ ◻ ◻ ◻ that shades the entire surface of the preceding group of signs, as exemplified by:



Advantages:

- Convenience, requiring fewer occurrences of controls for shading all signs within a group.

Disadvantages:

- One more code point.
- One rendering can be achieved in two ways, by a single group shading control or by shading each sign in that group individually.
- Precluding shading of blanks (cf. Option E) by a syntactic restriction becomes impracticable, as a blank may be nested deep inside a group.

(Option H) quarter group shading

In addition to, or in place of Option G, one may introduce four controls ◻ ◻ ◻ ◻, ◻ ◻ ◻ ◻, ◻ ◻ ◻ ◻, ◻ ◻ ◻ ◻ that each shade a quarter of the preceding group. Table 25 (d) exemplifies their use.

Advantages:

- For encoders it is more convenient and more in line with common practices to roughly indicate the damaged area of a group, without needing to indicate for each sign individually whether it is damaged.
- The rendering is often less ragged and more natural than an encoding where individual signs (or quarters thereof) are shaded.
- The hieratic script has *ligatures*, which are single shapes that are transcribed as multiple hieroglyphs. If such a ligature is partially damaged, then shading at the granularity of (quarters of) individual hieroglyphs becomes contrived, and group shading can provide a more appropriate abstraction of the state of the encoded text.

Disadvantages:

- There would then be two competing kinds of encoding of shading. In particular, if a group consists of a single sign, then sign shading and group shading are rendered the same.
- Which signs fall within shaded quarters of a group may depend on the font and on the rendering algorithm, as explained in Appendix B.
- Connected to this, search for damaged or undamaged occurrences of signs is made more difficult, because the encoding does not contain the information which individual signs are damaged.
- Precluding shading of blanks (cf. Option E) by a syntactic restriction becomes impossible, as a blank may be nested deep inside a group, and whether a blank falls within the shaded surface cannot be determined with certainty.

This is another matter where feedback from the Script Ad Hoc Committee would be particularly welcome.

(Option I) quarter group shading without sign shading

Given that quarter group shading (Option H) is widely used in publications of hieroglyphic and hieratic texts today, one may further consider adopting group shading while dropping sign shading from the proposal altogether. The advantages over Option H are:

- Implementation becomes simpler.
- If a text were to combine sign shading and group shading, then text search for damaged signs would succeed if sign shading is used and fail if group shading is used, which is likely to confuse users, as sign shading and group shading could be rendered the same in many instances. This confusion is avoided if sign shading does not exist.

Disadvantages:

- There is limited granularity to indicate which (parts of) signs are damaged. This limitation may be felt especially in Ptolemaic inscriptions, in which groups tend to consist of many signs.
- Shading can generally not be taken into account in text search.

References

- [1] J.P. Allen. *Middle Egyptian Literature*. Cambridge University Press, 2015.
- [2] J. von Beckerath. *Handbuch der ägyptischen Königsnamen*. Deutscher Kunstverlag, 1984.
- [3] H. Beinlich and M. Saleh. *Corpus der hieroglyphischen Inschriften aus dem Grab des Tutanchamun*. Griffith Institute, Ashmolean Museum, Oxford, 1989.
- [4] S. Biston-Moulin and C. Thiers. *Le temple de Ptah à Karnak*, volume 49 of *Bibliothèque générale*. Institut français d'archéologie orientale, Le Caire, 2016.
- [5] J. Buurman and E. de Moel. GLYPH – computer programs for Egyptologists. In *Informatique et Égyptologie 3*, pages 4–7, 1987.

- [6] J. Buurman, N. Grimal, M. Hainsworth, J. Hallof, and D. van der Plas. *Inventaire des signes hiéroglyphiques en vue de leur saisie informatique – Informatique et Égyptologie 2*. Institut de France, Paris, 3rd edition, 1988.
- [7] J. Capart. Une liste damulettes. *Zeitschrift für Ägyptische Sprache und Altertumskunde*, 45:pp. 14–21 & pl. 1–2, 1908/1909.
- [8] E. Chassinat. *Le temple de Dendara*. Institut français d’archéologie orientale du Caire, 1934.
- [9] M.E. Chioffi and G. Rigamonti. *I Racconti di Re Kheope – The Tales of king Kheops*. Duat Edizioni, 2005.
- [10] M. Collier and B. Manley. *How to read Egyptian hieroglyphs: A step-by-step guide to teach yourself*. British Museum Press, 1998.
- [11] M. Dessoudeix. *Lettres égyptiennes*, volume 2. Actes Sud, 2012.
- [12] A. Erman and H. Grapow. *Wörterbuch der Ägyptischen Sprache*. Akademie-Verlag, Berlin, 1926–1961.
- [13] A. Glass. Cluster model for Egyptian hieroglyphic quadrats. <http://www.unicode.org/L2/L2020/20176-hierogyph-cluster.pdf>, 2020.
- [14] A. Glass, I. Hafemann, M.-J. Nederhof, S. Polis, B. Richmond, S. Rosmorduc, and S. Schweitzer. A method for encoding Egyptian quadrats in Unicode. <http://www.unicode.org/L2/L2017/17112r-quadrat-encoding.pdf>, 2017.
- [15] G. Goyon. *Nouvelles inscriptions rupestres du Wadi Hammamat*. Imprimerie Nationale, Paris, 1957.
- [16] R. Hannig. *Grosses Handwörterbuch Ägyptisch-Deutsch: die Sprache der Pharaonen (2800-950 v. Chr.)*. Verlag Philipp von Zabern, Mainz, 1995.
- [17] T.G.H. James. *Hieroglyphic texts from Egyptian stelae, etc., Part I*. British Museum, 2nd edition, 1961.
- [18] K.A. Kitchen. *Ramesside Inscriptions*, volume 6. B.H. Blackwell, 1983.
- [19] R. Koch. *Die Erzählung des Sinuhe*. Fondation Égyptologique Reine Élisabeth, 1990.
- [20] D. Kurth. *Einführung ins Ptolemäische – Eine Grammatik mit Zeichenliste und Übungsstücken*. Backe-Verlag, 2008.
- [21] R. Landgrafova. *It is My Good Name that You Should Remember*. Charles University in Prague, 2011.
- [22] M.-J. Nederhof. The Manuel de Codage encoding of hieroglyphs impedes development of corpora. In S. Polis and J. Winand, editors, *Texts, Languages & Information Technology in Egyptology*, pages 103–110. Presses Universitaires de Liège, 2013.
- [23] M.-J. Nederhof. A note on the syntax of Ancient Egyptian hieroglyphic control characters. <https://www.unicode.org/L2/L2018/18236-nederhof.pdf>, 2018.
- [24] M.-J. Nederhof. A note on OpenType implementation of Ancient Egyptian hieroglyphic text. <https://www.unicode.org/L2/L2019/19331-egyptian-opentype.pdf>, 2019.
- [25] M.-J. Nederhof. RES (Revised Encoding Scheme). <http://mjn.host.cs.st-andrews.ac.uk/egyptian/res/>, 2019.

- [26] M.-J. Nederhof, V. Rajan, J. Lang, S. Polis, S. Rosmorduc, T.S. Richter, I. Hafemann, and S. Schweitzer. A comprehensive system of control characters for Ancient Egyptian hieroglyphic text (preliminary version). <http://www.unicode.org/L2/L2016/16177-egyptian.pdf>, 2016.
- [27] M.-J. Nederhof, V. Rajan, J. Lang, S. Polis, S. Rosmorduc, T.S. Richter, I. Hafemann, and S. Schweitzer. A system of control characters for Ancient Egyptian hieroglyphic text (updated version). <http://www.unicode.org/L2/L2016/16210r-egyptian-control.pdf>, 2017.
- [28] S. Polis. The functions and toposyntax of Ancient Egyptian hieroglyphs. *SIGNATA*, 9:291–363, 2018.
- [29] G.A. Reisner and M.B. Reisner. Inscribed monuments from Gebel Barkal. *Zeitschrift für Ägyptische Sprache und Altertumskunde*, 69:24–39, 1933.
- [30] S. Rosmorduc. JSesh. <http://jseshdoc.qenherkhopeshef.org>, 2020.
- [31] S.J. Seidlmayer. Eine Schreiberpalette mit ägyptischer Aufschrift. *Mitteilungen des deutschen archäologischen Instituts, Abteilung Kairo*, 47:319–330, 1991.
- [32] K. Sethe. *Urkunden der 18. Dynastie, Fascicle 8*. Hinrichs, Leipzig, 1906.
- [33] K. Sethe. *Die altaegyptischen Pyramidentexte*, volume 1. Hinrichs, Leipzig, 1908.
- [34] K. Sethe. *Urkunden der 18. Dynastie, Fascicle 1*. Hinrichs, Leipzig, 1927.
- [35] K. Sethe. *Urkunden der 18. Dynastie, Fascicle 2*. Hinrichs, Leipzig, 1927.
- [36] N. Stief. Hieroglyphen, Koptisch, Umschrift, u.a. – ein Textausgabesystem. *Göttinger Miscellen*, 86:37–44, 1985.
- [37] N. Stief. PLOTTEXT – ein Programmsystem zur Ausgabe von Texten. Version 4.09, Regionales Hochschulrechenzentrum (RHRZ) der Universität Bonn, 2001.
- [38] R.A. Wahid, M.-C. Bruwier, N. Gauthier, and M. Haggag. *Antiquités égyptiennes de la Préhistoire à la Basse Époque*. Centre d’Etudes Alexandrines, 2019.
- [39] D.A. Werning. *Das Höhlenbuch im Grab des Petamenophis (TT33)*. Propylaeum, 2020.

A Sign shading

We have discussed the importance of being able to indicate which signs are damaged, by the use of shading. Indicating damaged areas *between* signs seems outside the concerns of Unicode however, and this will therefore not be encoded explicitly. Where several signs within one group are damaged, the most desirable rendering is such that the shaded areas belonging to the different signs are connected. We illustrate this on the basis of Table 26 (a), for a group of the form $A \boxed{*} B \boxed{:} C \boxed{:} D \boxed{*} E \boxed{*} F$. The total surface is divided into connected rectangles, indicated by dashed lines. If for example A and B are both entirely shaded, then this forms one full shaded rectangle around A and B , including the top half of the space between $A \boxed{*} B$ and C . If all signs in the group are shaded, then the full square is shaded, without leaving seams in between the signs.

This semantics implies that the aspect ratio of a shaded area (i.e. the ratio between width and height) need not correspond to the aspect ratio of the shaded sign(s) individually. This is illustrated by Table 26 (b), which corresponds to an encoding $A \boxed{*} \boxed{\blacksquare} \boxed{\blacksquare} \boxed{\blacksquare} \boxed{\blacksquare} \boxed{\blacksquare} \boxed{:} B$, involving a fully shaded blank. Although the blank is square, the shaded surface is more generally a rectangle with underspecified aspect ratio. Converse

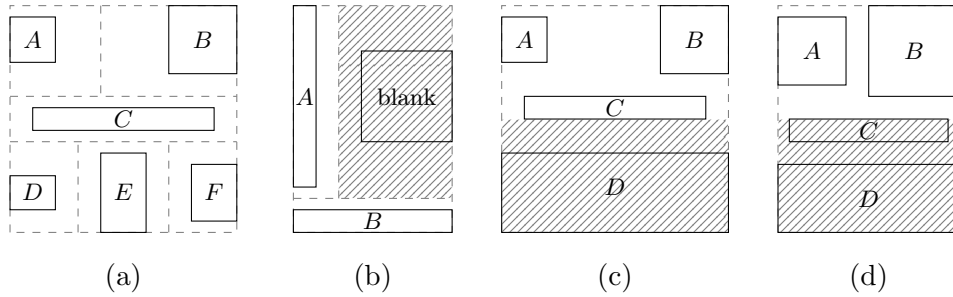


Table 26: Semantics of shading.

examples could be found of shaded tall and narrow signs, or low and wide signs, that within a group result in a shaded area that is roughly square.

This observation also has implications for considering further blank signs, as in Option D above; see also Option F. This is because in the general case, the aspect ratio of the rendered blank area (whether or not it is shaded as in the above example) is not solely determined by the aspect ratio of the blank itself, but also by the dimensions of other signs within the same group, as well as by the implemented rendering algorithm. This could lead to confusion among users, who may expect that by choosing a specific shape of blank, the rendered form will contain that same shape. For implementation-dependent behaviour, see also below.

B Quarter group shading

Here we consider the quarter group shading controls of Option H. One issue is that these controls underspecify which signs in a group are damaged. This is illustrated by Table 26 (c-d), for an encoding of the form:

$A * B : C : D$

where the bottom half of the preceding group is shaded by the two shading controls. This may be roughly equivalent to:

$A * B : C : D$

or equivalent to:

$A * B : C$ $: D$

or perhaps it is equivalent to:

$A * B : C$ $: D$

In other words, the rendering of the quarter group shading here may suggest that C is entirely undamaged in the inscription or that it is entirely damaged, or perhaps that only a portion of C is damaged. This depends on the relative dimensions of the signs in the font and on the used rendering algorithm. It should here be pointed out that rendering as discussed by [24], which is realized in a general-purpose programming language that can perform arithmetic operations, is very different from the implementation of [13], in which controls are ‘interpreted’ by OpenType, which is not capable of arithmetic operations, or can at best approximate such operations by using a small number of discrete values. The former implementation is recursive, making two passes through a group and its subgroups, first bottom-up and then top-down, formatting smaller subgroups first and then rescaling them in their entirety depending on available space in the larger subgroups. This

may result in a deeply nested subgroup being scaled down repeatedly. The OpenType implementation is necessarily more ‘holistic’, due to its inability to implement any notion of recursion, and would generally scale down nested subgroups less aggressively. This can result in vastly different scaling and positioning between the two implementations, which nonetheless both respect the intended semantics of the joining controls.