L2/17-112

A method for encoding Egyptian quadrats in Unicode

Andrew Glass, Ingelore Hafemann, Mark-Jan Nederhof, Stéphane Polis, Bob Richmond, Serge Rosmorduc, Simon Schweitzer

April 26th, 2017

Background documents:

N1944 Encoding Egyptian Hieroglyphs in Plane 1 of the UCS. Michael Everson. 1999-01-09.

L2/15-123 Proposal to encode three control characters for Egyptian Hieroglyphs (revised). Bob Richmond. 2015-05-04.

L2/16-018 Proposal to encode three control characters for Egyptian Hieroglyphs (revised). Bob Richmond, Andrew Glass. 2016-01-27.

L2/16-090 Comments on three control characters for Egyptian Hieroglyphs. Mark-Jan Nederhof, Vinodh Rajan. 2016-04-25.

L2/16-104 Observations: L2/16-090 [Egyptian]. Bob Richmond. 2016-05-02.

L2/16-177 A comprehensive system of control characters for Ancient Egyptian hieroglyphic text (preliminary version). Mark-Jan Nederhof, et al. 2016-06-30.

L2/16-210R A system of control characters for Ancient Egyptian hieroglyphic text. Mark-Jan Nederhof, et al. 2016-07-25, revised 2017-01-25.

L2/16-214 An Extension to the three control characters for Egyptian Hieroglyphs and some additional remarks. Bob Richmond. 2016-08-01.

L2/16-231 Proposal for Ancient Egyptian encoding in Unicode. Serge Rosmorduc, et al. 2016-08-04.

L2/16-232 Preliminary analysis of Egyptian Hieroglyph quadrat types. Andrew Glass. 2016-08-05.

L2/16-233 Addendum to: A system of control characters for Ancient Egyptian hieroglyphic text. Mark-Jan Nederhof, et al. 2016-08-05.

L2/16-257 Source analysis of an extended Egyptian Hieroglyphs repertoire (Hieroglyphica based). Michel Suignard. 2016-09-19.

L2/17-073 New draft for the encoding of an extended Egyptian Hieroglyphs repertoire (Hieroglyphica based) (17MB, and associated database PDF snapshot). Michel Suignard. 2017-03-23.

Summary

The proposal to encode three control characters for Egyptian Hieroglyphs was accepted at the February 2016 UTC meeting. Following that acceptance some members of the Egyptological community raised concerns that the accepted controls were insufficient for their requirements. This led to a series of discussions and document submissions between specialists and implementers that explored the issues relating to encoding the quadrat structures that are an inherent feature of Egyptian Hieroglyphic writing. The present document combines expert input on Egyptian Hieroglyphs and a detailed exploration of possible solutions. It does not attempt to summarize or repeat arguments and details from earlier documents. Cross-references to earlier submissions are provided where appropriate. The result is a draft proposal for a system of controls that will enable plain text encoding of Egyptian Hieroglyphs in quadrats.

Principles and requirements

Principles

The two principles that guided this effort to encode Egyptian quadrats are:

- Structural completeness — Unicode encoded text in Egyptian Hieroglyphs must be able to faithfully render the structures inherent to the writing system
- Feasibility — Rendering text must be possible using existing font technologies

Structural completeness

Structural completeness means that users of the Egyptian Hieroglyphic encoding can expect that quadrat structures they encounter in their work can be encoded correctly, i.e., the necessary controls are available to encode the structure. To begin with, all 45 of the attested quadrat structures documented in L2/16-232 can be encoded using this system of controls. Furthermore, the system supports many other quadrat structures. Therefore, if someone typing an Egyptian text encounters a quadrat structure that had not been anticipated by the developer of their Egyptian Hieroglyphic font, it is highly likely that they would be able to correctly render the intended form with a font update rather than being required to make a proposal to add additional quadrat controls to Unicode.

Note that this applies to structural forms only and does not apply to hieroglyphs themselves. If a user of Egyptian Hieroglyphs encounters signs that are not graphical variants of signs already in Unicode, there is no choice but to request support for a new character. This is no different than for any other writing system in Unicode. For example, new Chinese characters are routinely added to Unicode.

The middle ground and grey area between these two paths (font update vs. Unicode update) is when there is ambiguity between encoding a sign as an atomic character or as a combination of existing characters with some kind of control to join them. Participants in the field need to investigate guidelines for determining whether a given sign or quadrat should be encoded as an atomic character or enabled as a sequence of other characters.

Digital encoding of Egyptian hieroglyphic texts is currently dominated by the Manuel de Codage (MdC) encoding system (1985 and later revisions). Clarifications and extensions to MdC have been made in popular software solutions such as JSesh which is used by the TLA and Ramses corpus projects. PLOTTEXT (1983), introduced an earlier encoding scheme which has been widely used in the German-speaking Egyptological community.[1] It has elements in common with a more recent encoding called RES (Revised Encoding Scheme) which is used in the St Andrews corpus. The authors of this proposal have considered the established use of comparable controls in these systems in constructing the proposed control set. Comparisons between the controls of Unicode and those of existing encoding schemes are clarified through the various examples in L2/16-210R.

It is not claimed that the control primitives can capture all details of the appearance of the original manuscripts, but that enough of the appearance is captured so that Egyptologists have been satisfied that the resulting encodings offer an acceptable representation of original texts. This abstraction to an acceptable representation is consistent with Unicode's goal for character encoding in plain text which

[1] PLOTTEXT was used to render the Hieroglyphs in E. Graefe. *Mittelägyptische Grammatik für Anfänger.* Harrassowitz Verlag, Wiesbaden, 1994.

contrasts with paleographic or facsimile representation which are out-of-scope. For further discussion on what constitutes plain text encoding for Egyptian, see Nederhof et al. “Case studies” (L2/16-210R §13) and Richmond’s Irregular Hieroglyph clusters (accessed 4/2/2017). The former illustrates a few lines from two separate 12th Dynasty hieroglyphic texts. The latter presents examples from the 18th Dynasty Tomb of Rekhmire (TT100). Both discussions detail quadrats in the original texts that can be represented in the proposed encoding, and both point out several examples that contain details of presentation that are finer than the proposed system of control signs can capture. However, the consensus of specialists who have put their names to the current proposal is that the proposed controls provide a desirable and acceptable level of fidelity in rendering Egyptian Hieroglyphs in plain text.

Since the work on the Egyptian Hieroglyphic corpus is ongoing, it is expected that new hieroglyphic signs will be discovered. And these may be proposed as additions to Unicode. It is possible that new discoveries and investigations will improve the understanding of the script and may in turn reveal inherent features that warrant the attention of encoding specialists.

Feasibility

The feasibility of rendering quadrats for Egyptian Hieroglyphs using the proposed controls and OpenType has been demonstrated with three prototype OpenType fonts.

The first font, by Bob Richmond, uses a ligature substitution method that maps each encoded text sequence to a precomposed glyph for the intended quadrat. This approach targets a repertoire of quadrats based on corpus analysis. The practical limit for the method is the OpenType glyph limit of 64K, meaning that a maximum of 64K unique signs and quadrats could be supported by a font. This is easily sufficient to create a font that targets over 99% of the corpus of middle Egyptian texts (based on Richmond’s data analysis) using no more than about ten thousand precompositions. A prototype variation of this method uses high frequency quadrat structures and hieroglyph shape classification with marker glyphs to reduce the number of precomposed quadrat glyph layers required by layered polychromatic fonts. This method can also be used to optimize monochrome fonts or avoid the 64K limit (if this became an issue).

The second and third fonts are by Mark-Jan Nederhof and Andrew Glass. These fonts both use a dynamic OpenType approach to analyze encoded sequences and compute the intended quadrat structure and sign sizes. These fonts show that the goal of being able to generate quadrats for arbitrary structures and sign combinations is achievable within the size limits of OpenType lookup tables. The OpenType glyph limit isn’t a concern for this type of font.

A hybrid approach that combines features of the ligature and dynamic methods is also possible, as a straightforward extension of the dynamic approach. In this case, the ligature method would be used for the target repertoire of quadrats (which has performance and fidelity advantages), while the dynamic method would be reserved for quadrats that fell outside the target repertoire.

In addition to solutions using OpenType fonts, specialized software could leverage the proposed control characters as part of a dedicated system for rendering Egyptian hieroglyphic texts. Such a system may use higher-level protocols to render additional display features used by specialists, such as quadrat component shading. The JSesh and RES systems are examples of such solutions.

Further details of font solutions are given in the section on OpenType strategies below.

Documentation

The authors of this proposal appreciate that additional documentation is needed for implementers and end users of Egyptian Hieroglyphs in order to be able to use these control characters effectively. Two forms of documentation have been discussed and are planned as a follow-up to this proposal.

- The chapter on Egyptian Hieroglyphs in TUS will need to be revised to describe the effective usage of the quadrat forming control signs
- Font developers will need guidance on the quadrat repertoire, exceptional cases, and other guides to implementation. This may be published as a Unicode Technical Note

Beyond these, it is understood that tools and associated documentation will be valuable for users of the Egyptian encoding with quadrat control characters. For example, keyboard input methods and functions that can convert transliterations in popular transcription formats into valid Unicode sequences.

Encoding options

Based on an analysis of the corpus, the current requirement for embedding levels is three. It is possible that examples requiring more levels could be identified. A key challenge for the encoding therefore, is how to indicate the embedding levels in the encoding. Six basic options have been considered as follows:

Paired controls

In one of the options that has been pursued, horizontal joiners are represented by one pair of parentheses, and vertical joiners by another such pair. The advantage is the conceptual simplicity of the syntax, which is unambiguous without needing further provisions. The disadvantage is that no less than two control characters are needed for a horizontal or vertical arrangement of only two signs.

Polish notation

A syntax using a form of Polish notation has also been considered. Here one operator precedes two subexpressions, each of which may be a single sign, or may again be an operator preceding two subexpressions. As before, the notation is unambiguous and allows any number of levels of embedding. The downside is that this notation is less human readable and challenging to implement in fonts.

Explicit controls

Two controls, one for horizontal arrangement and one for vertical arrangement are encoded for each level. Within a level the horizontal control has precedence over the vertical one. Where deeper embedding is needed, for example, if two vertically arranged groups of signs are to be combined horizontally, then similar controls are used on the next level. Deeper levels have higher precedence than the controls on the lower levels. Hence the primitives for horizontal and vertical arrangements need to be represented by several groups of controls, one for each level.
The advantage of this solution is that the linear encoding for a quadrat is simpler. The principle disadvantage of this approach is that each level needs to be explicitly encoded in Unicode and implemented via updates to rendering software and fonts.

Combined controls

A variant of the preceding approach. It uses a pair of controls, one for the join and one for the level of the join. This approach requires the number of target levels to be established at the encoded level and

runs the risk of either some redundancy (if a level control is encoded but not required) or incompleteness (if a level beyond the encoded set is discovered).

Repeated controls

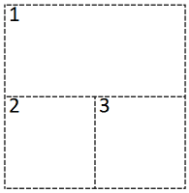
A second variant of the explicit controls approach. It indicates embedding level by repetition of the basic controls. This approach achieves arbitrary extension with some loss of readability. This approach is also prone to authoring difficulties as controls may be accidentally doubled, dropped, or split.

Binary controls with parentheses

A third variant of the explicit controls, but in this case pairs of parentheses are used to enclose encodings of embedded arrangements of signs. This notation allows any depth of embedding, requires only a small set of controls, and will be familiar and intuitive to users of the Manuel de Codage and similar forms of encoding. A negative aspect of the parentheses solution is that many quadrat sequences are necessarily longer than is the case with explicit controls. The implementation of parentheses in OpenType is possible but requires a few more rules as opposed to having explicit controls, however, the added overhead is small relative to the overall scale of an arbitrary OpenType solution.

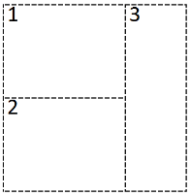
The encoding using binary controls with parentheses is the one used by this proposal.

Simple quadrat showing alternative encoding schemes



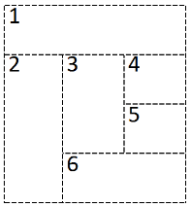
Source	
Symbolic	A15 : N23 * Z1
Characters	𓀭 𓇋 𓏺
Code points	U+13012 U+13207 U+133E4
Encoding	**Example**
Paired controls	[: 1 [* 2 3 *] :]
Polish notation	: 1 * 2 3
Explicit controls	1 : 2 * 3
Combined controls	1 : $_1$ 2 * $_1$ 3
Repeated controls	1 : 2 * 3
Parentheses	1 : 2 * 3

Two-level quadrat showing alternative encoding schemes



Source	
Symbolic	(M8 : G1) * M40
Characters	𓊪 𓅱 𓏺
Code points	U+131B7 U+1313F U+131E9
Encoding	**Example**
Paired controls	[* [: 1 2 :] 3 *]
Polish notation	* : 1 2 3
Explicit controls	1 : 2 > 3
Combined controls	1 : $_2$ 2 * $_1$ 3
Repeated controls	1 : : 2 * 3
Parentheses	(1 : 2) * 3

Complex quadrat showing alternative encoding schemes



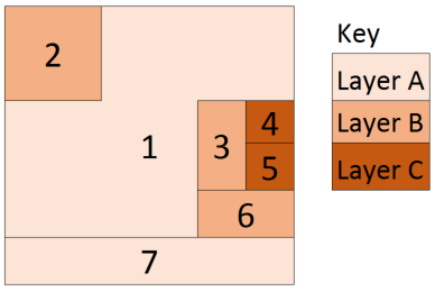
Source	Abydos temple of Ramesses II, p. 531–2
Symbolic	J15 : Z11 * (D2 * (D21 : X1) : N25)
Characters	
Code points	U+1341D U+133F6 U+13077 U+1308B U+133CF U+13209

Encoding	Example
Paired controls	[: 1 [* 2 [: [* 3 [: 4 5 :] *] 6 :] *] :]
Polish notation	: 1 * 2 : * 3 : 4 5 6
Explicit controls	1 = 2 > 3 * 4 . 5 : 6
Combined controls	1 : $_1$ 2 * $_1$ 3 * $_2$ 4 : $_3$ 5 : $_2$ 6
Repeated controls	1 : 2 * 3 * * 4 : : : 5 : : 6
Parentheses	1 : 2 * (3 * (4 : 5) : 6)

Text encoding sequence

In all the encoding systems described above, the sequence for encoding the text units of the quadrat (hieroglyphs and controls) follows a logical progression starting from the outermost embedding group. Within a single embedding group, encoding proceeds from left to right and top to bottom starting at the top left hieroglyph. When an embedding group is encountered, the same progression logic applies inside that group.

Here is an artificial left-to-right example to illustrate an extreme case:



This would be encoded in the proposed parentheses encoding as:

1 ◰ 2 ◲ (3 * (4 : 5) : 6) : 7

The encoding begins with top-left sign of the outer layer (layer A), i.e., sign "**1**". That sign has an insertion in the top left "◰". This inserted sign is sign "**2**" in the sequence because inserted groups are processed in the order, top-left, bottom-left, top-right, bottom-right. Sign 2 is in layer B but the linear encoding doesn't require parentheses because 2 is the only sign governed by that insertion control. There is a group inserted in the bottom right of sign 1. So, the next control is "◲". This group consists of multiple signs so a parenthesis is needed to mark the start of the layer B group "**(**". In this group, the top left sign is "**3**". The encoding progresses left-right then top to bottom. To the left of 3 is an embedded group, so the control asterisk "*****" is used to mark the horizontal connection. This is followed by an opening parenthesis "**(**" to mark the start of the layer C group. In this group, the top left sign is "**4**". There is no sign to the right, so the encoding progresses downward inside this group. The vertical join is

marked with colon "**:**" and the next sign is "**5**". Now the layer C group is complete so it is closed with a closing parenthesis "**)**". The encoding sequence reverts to the layer B group and progresses downward. The vertical join is again marked with colon "**:**". The next sign is "**6**". Now the layer B group is complete so it is closed with a closing parenthesis "**)**". The encoding sequence reverts to the layer A group and progresses downward. The vertical join is again marked with colon "**:**". The final sign is "**7**".

Any cases in which the logical reading order differs from the visual order must be encoded based on the visual appearance with mapping to the reading order handled as a higher-level protocol. A comparable division exists for Japanese text which must separate the encoded form from the collation form.

Proposed controls

Based on a detailed study of implementability (for details, see later sections), the following controls are proposed:

Default glyph	Code point	Character name
:	13430	EGYPTIAN HIEROGLYPH VERTICAL JOINER
*	13431	EGYPTIAN HIEROGLYPH HORIZONTAL JOINER
◰	13432	EGYPTIAN HIEROGLYPH INSERT TOP START
◱	13433	EGYPTIAN HIEROGLYPH INSERT BOTTOM START
◳	13434	EGYPTIAN HIEROGLYPH INSERT TOP END
◲	13435	EGYPTIAN HIEROGLYPH INSERT BOTTOM END
+	13436	EGYPTIAN HIEROGLYPH STACK MIDDLE
(	13437	EGYPTIAN HIEROGLYPH SEGMENT START
)	13438	EGYPTIAN HIEROGLYPH SEGMENT END

Details of the proposed controls

Joiners

There is agreement that the rectilinear characteristic of Egyptian quadrats requires paired controls for horizontal and vertical joins.

Vertical joiner (U+13430)

The seated man (A1) is vertically above the house (O1):

	Symbolic	A1 : O1
	Unicode	U+13000 **U+13430** U+13250

In vertical text, the form of the quadrat does not change.

Horizontal joiner (U+13431)

In left-to-right text, (which is the default encoding order for Egyptian Hieroglyphs), the item to the left of the horizontal control is drawn on the left. In this case, the bowl (W24), is logically before the abbreviated quail chick (Z7):

	Symbolic	W24 * Z7	LTR
	Unicode	U+133CC **U+13431** U+133F2	

When the text direction is forced to be right-to-left the order of the symbols is reversed but the encoding is unchanged:

	Symbolic	W24 * Z7	RTL
	Unicode	U+133CC **U+13431** U+133F2	

Insertions

Insertion of a hieroglyph within the rectangle of another hieroglyph is a common productive feature of Egyptian Hieroglyphic writing. Insertion may be in one of the four corners. When inserted in one of the corners, the inserted glyph may be offset to some extent depending on the form of the outer glyph. As with the horizontal joiner, the side (left/right) of the insertion is determined by the directionality of the writing line. Therefore, the corner insertions are named with start and end to identify their logical rather than visual position.

Insert top start (U+13432)

The bread sign (X1) is inserted in the top start quarter of the lion (F4):

	Symbolic	F4 ◰ X1	LTR
	Unicode	U+13102 **U+13432** U+133CF	

	Symbolic	F4 ◰ X1	RTL
	Unicode	U+13102 **U+13432** U+133CF	

Insert bottom start (U+13433)

The seated man (A1) is inserted in the bottom start corner of the cobra (I10):

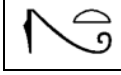
	Symbolic	I10 ◱ A1	LTR
	Unicode	U+13193 **U+13433** U+13000	

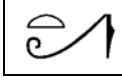
	Symbolic	I10 ◱ A1	RTL
	Unicode	U+13193 **U+13433** U+13000	

In the case of an enclosing cobra, the inserted hieroglyph(s) are centered in the space created by the cobra shape. When inset with some bird signs, the bottom-left insertion may be raised slightly to avoid occluding the feet of the bird. Similar approximations may apply with other base signs.

Insert top end (U+13434)

The bread sign (X1) is inserted in the top end corner of the *wḏꜣt*-eye (D17):

	Symbolic	D17 ◳ X1	LTR
	Unicode	U+13087 **U+13434** U+133CF	

	Symbolic	D17 ◳ X1	RTL
	Unicode	U+13087 **U+13434** U+133CF	

Insert bottom end (U+13435)

The bread sign (X1) is inserted in the bottom end corner of the Ibis (G25):

	Symbolic	G25 ◲ X1	LTR
	Unicode	U+1315C **U+13435** U+133CF	

	Symbolic	G25 ◲ X1	RTL
	Unicode	U+1315C **U+13435** U+133CF	

Compound insertions

When multiple insertions occur within the same outer glyph, they must be encoded in the standard order, top left, bottom left, top right, bottom right. Only one insertion control of each type can be used with the same outer glyph at the same embedding level. These restrictions are required to ensure consistency for searching and collation purposes. Enforcing this order is not expected to be supported at the rending engine level, but should be enforced by fonts. Orthography/spelling checkers may also be developed to validate quadrat structures.

Insertion sequence:

The loaf (X1) is inserted in three corners of the pintail duck (G39):

	Symbolic	G39 ◱ X1 ◳ X1 ◲ X1	LTR
	Unicode	U+1316D **U+13433** U+133CF **U+13434** U+133CF **U+13435** U+133CF	

	Symbolic	G39 ◱ X1 ◳ X1 ◲ X1	RTL
	Unicode	U+1316D **U+13433** U+133CF **U+13434** U+133CF **U+13435** U+133CF	

Inserted group:

Inserted items may themselves be compounded forms involving horizontal and vertical joins. Such cases require the inserted group to be bound using the segment begin and end controls. For example, two hieroglyphs of the sun rising over a mountain (N27) are stacked vertically and inserted in the top end corner of the falcon of Horus bearing the sun (G9):

	Symbolic	G9 ◳ (N27 : N27)	LTR
	Unicode	U+1314A **U+13434 U+13437** U+1320C **U+13430** U+1320C **U+13438**	

	Symbolic	G9 ◳ (N27 : N27)	RTL
	Unicode	U+1314A **U+13434 U+13437** U+1320C **U+13430** U+1320C **U+13438**	

Insertion within insertion:

Examples of insertions within insertions are known to exist, but are extremely rare.

	Symbolic	I10 ◱ (I10 ◱ I10)	LTR
	Unicode	U+13193 **U+13433 U+13437** U+13193 **U+13433** U+13193 **U+13438**	

	Symbolic	I10 ◱ (I10 ◱ I10)	RTL
	Unicode	U+13193 **U+13433 U+13437** U+13193 **U+13433** U+13193 **U+13438**	

Note that the sequence "I10 ◱ I10 ◱ I10" would be invalid since there can be only one insertion of the same type at the same level. Parentheses are required to indicate a deeper level embedding.

Other insertions

Other types of insertions have been discussed in L2/16-210R Nederhof et al., section 6. How to handle these cases remains an open question. The options are to encode them atomically or as a sequence using control signs.

	Symbolic	D60 ◱ X1 ?	LTR
	Unicode	TBD	

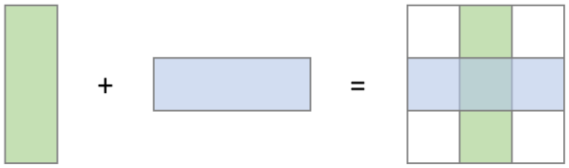
	Symbolic	D173	LTR
	Unicode	TBD	

Stacking

Stack middle (U+13436)

Stack middle is included since centered stacking is a common productive feature of the script. Including a stack middle control is expected to significantly reduce the need to bring additional requests for atomic stack characters to Unicode for encoding, and will speed up adoption of Egyptian Unicode.

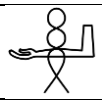
The stack middle control allows signs to be placed one on top of the other in the z-axis. The scope of this control is to allow a group of one or more hieroglyphs to stack on top of a second group of one or more other hieroglyphs. The arrangement of such stacks should be strictly limited to matching the mid-point of the lower group to the mid-point of the upper. When stacking two signs or groups of signs with unequal dimensions the resulting hieroglyph has a bounding box equal to the maximum height and width of the stacked groups. For example, a tall narrow glyph with dimensions 2 : 6, is stacked on top of a low wide glyph with dimensions 6 : 2. The resulting stacked form is a square with dimensions 6 : 6:



The stacker control has a higher precedence than the other signs. Groups are to be indicated with parentheses. Groups are stacked starting at the back and moving forward based on the encoded sequence.

Simple stack

The wick of twisted flax (V28) is stacked on top of the forearm (D36):

	Symbolic	D36 + V28	LTR
	Unicode	U+1309D **U+13436** U+1339B	

Compound stacks

Two forearms (D36) arranged vertically (y-axis) are stacked (z-axis) above two foot-hieroglyphs (D58) arranged horizontally (x-axis):

	Symbolic	(D58 * D58) + (D36 : D36)	LTR
	Unicode	**U+13437** U+130C0 **U+13431** U+130C0 **U+13438 U+13436 U+13437** U+1309D **U+13430** U+1309D **U+13438**	

Stack with insertion

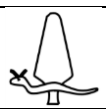
It is possible to stack and insert. In these cases, the stack should be composed before inserting. For example: A mouth (D21) is stacked on top of the head and neck of a canine (F12) and a bread sign (X1) is inserted in the bottom start corner. This enables the insertions to be positioned relative to the maximum height and width of the stack rather than relative to the height and width of one of the stacked groups.

	Symbolic	F12 + D21 ◱ X1	LTR
	Unicode	U+1310A **U+13436** U+1308B **U+13433** U+133CF	

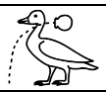
Offset stacks

Offset stacks are considerably rarer than centered stacks. Due to the complexity of specifying the details of an offset, the preferred option for offset stacks is to assign any offset to the font level or, if that is not appropriate or possible, encode them atomically.

For example, the viper (I9) stacked over the trunk of a tree (M1) may be rendered based on mid-point to mid-point alignment, or a ligature-based font might fix the offset of the viper on aesthetic grounds:

	Symbolic	M1 + I9 (not M48)	LTR
	Unicode	U+131AD **U+13436** U+13191	

On the other hand, a jug pouring liquid (W54) stacked at the top start of a white fronted goose (G38) might rather be encoded atomically.

	Symbolic	G70	LTR
	Unicode	***U+13F7A** (proposed code point, not final)	

Segment boundaries

The notation system requires bounding controls to indicate nested groups. In practice support for three levels of embedding is sufficient to encode the vast majority of quadrats. In principle, the mechanism supports deeper nesting which may be of greatest interest to specialized software. Controls that co-occur within a single embedding segment are interpreted in the precedence order: vertical joiner < horizontal joiner < insertion controls < stacker.

Segment start (U+13437) and Segment end (U+13438)

The earlier example from the Abydos temple of Ramesses shows three levels of embedding (e.g., ).

	Symbolic	J15 : Z11 * (D2 * (D21 : X1) : N25)
	Unicode	U+1341D U+13430 U+133F6 U+13431 **U+13437** U+13077 U+13431 **U+13437** U+1308B U+13430 U+133CF **U+13438** U+13430 U+13209 **U+13438**

Character properties

General properties

```
13430;EGYPTIAN HIEROGLYPH VERTICAL JOINER;Mn;0;NSM;;;;;N;;;;;
13431;EGYPTIAN HIEROGLYPH HORIZONTAL JOINER;Mn;0;NSM;;;;;N;;;;;
13432;EGYPTIAN HIEROGLYPH INSERT TOP START;Mn;0;NSM;;;;;N;;;;;
13433;EGYPTIAN HIEROGLYPH INSERT BOTTOM START;Mn;0;NSM;;;;;N;;;;;
13434;EGYPTIAN HIEROGLYPH INSERT TOP END;Mn;0;NSM;;;;;N;;;;;
13435;EGYPTIAN HIEROGLYPH INSERT BOTTOM END;Mn;0;NSM;;;;;N;;;;;
13436;EGYPTIAN HIEROGLYPH STACK MIDDLE;Mn;0;NSM;;;;;N;;;;;
13437;EGYPTIAN HIEROGLYPH SEGMENT START;Mn;0;NSM;;;;;N;;;;;
13438;EGYPTIAN HIEROGLYPH SEGMENT END;Mn;0;NSM;;;;;N;;;;;
```

Line breaking

Line breaks should occur at quadrat boundaries and not within a quadrat. Therefore, the JOINER characters should act as glue to connect EGYPTIAN HIEROGYPHS, thus in the format of LineBreak.txt:

```
13430..13438; GL # [9] EGYPTIAN HIEROGLYPH VERTICAL JOINER..EGYPTIAN HIEROGLYPH SEGMENT END
```

Text segmentation

Grapheme cluster boundaries, equate to quadrat boundaries. Word and sentence boundaries could occur within a quadrat. In such cases, word selection and other word boundary effects would not be able to automatically break inside a quadrat. Comparable behavior occurs in implementations of Indic writing systems where word boundaries can occur within a ligature or conjunct consonant sign. The proposed joiner characters should be given the value in PropList.txt:

```
13430..13438; Extender # Mn [9] EGYPTIAN HIEROGLYPH VERTICAL JOINER..EGYPTIAN HIEROGLYPH
SEGMENT END
```

Syllabic category

Joiner characters should be given properties in IndicSyllabicCategory.txt so that shaping and caret advancement can be handled correctly:

```
13430..13438; Structure_Control # Mn [9] EGYPTIAN HIEROGLYPH VERTICAL JOINER..EGYPTIAN
HIEROGLYPH SEGMENT END
```

The property value Structure_Control is a proposed value that may also be used for Mayan and other languages with two-dimensional joining behaviors. It should be used to fuse adjacent characters on both sides of the control into a single orthographic unit for the purposes of shaping with per-cluster features. Caret placement would be prohibited between characters joined with a Structure_Control in standard software implementations.

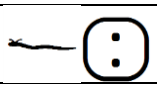
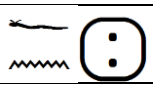
Collation

The proposed structure controls should sort in the order of their Unicode values.

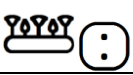
Due to variation in the quadrat structures used for different instances of the same word, it will be desirable to be able to find quadrats that include the same basic signs but use different structure controls. It is also desirable to be able to do exact matching for a quadrat with the same structure. As such, the structure controls may be given a diacritic weight that can be optionally included or ignored by the search mechanism.

Incomplete quadrats

The control characters proposed in this document should render invisibly if they are in a valid well-formed sequence that the font supports. If a quadrat sequence is not supported by the selected font, or if a control occurs in isolation, the controls should render visibly so that someone reading the text can interpret the intent of the sequence or identify an error in the sequence encoding. This also applies to controls entered during the build-up of a quadrat sequence. For example:

Sequence	1	2	3	4	5	6	7
Character	U+13191	U+13430	U+13216	U+13430	U+13113	U+13433	U+13000
Code point	𓆑	:	𓆖	:	𓄓	𓐫	𓀀
Display							

Because only an opening parenthesis is required to signal a deeper embedding level, quadrat buildup can proceed in the same way with parenthetical notation.

Sequence	1	2	3	4	5	6	7
Character	U+13437	U+131B7	U+13430	U+1313F	U+13438	U+13431	U+131E9
Code point	(	𓅷	:	𓄿	)	*	𓇉
Display							

Note that a limitation of this model is that there is no visible difference between steps 4 and 5 in the above example. That is to say, there is no visible enforcement of paired parentheses if the quadrat ends

with the closure of one or more parentheses. It is conceivable that this could be managed at the font layer by inserting a glyph to indicate the need for a closing parenthesis.

Structure validation

Beyond simply failing to render with the expected form, validation of quadrat sequences is not the responsibility of fonts. Some cluster validation has been an important responsibility of shaping engines for some writing systems. This is usually indicated by the insertion of a dotted circle glyph (U+25CC ◌) in cases where the shaping engine's cluster validation process found an error in the text run. More generalized validation of sequences is the domain of orthographic consistency checkers such as spelling and grammar checkers. Given the requirement that the encoding solution work with existing software, the responsibility for structure validation must be optional at the rendering engine level.

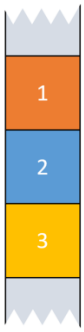
Horizontal and vertical layout

Vertical mode	Horizontal mode
	
Vertical and horizontal writing on the Stela of Ity, 12th Dynasty, c. 1951 BCE. (BM EA 586)	

A fundamental flexibility of Egyptian Hieroglyphic writing is that text can be laid out horizontally or vertically. The impact of this on quadrat structures is considered in this section. Nederhof et al. contains additional details and background (see L2/16-210R §§ 10.2 and 11.1). Existing encoding schemes have all assumed that horizontal and vertical text directions are symmetric in the primitives they require. The present proposal is no different in this respect. However, there are considerations to be taken into account.

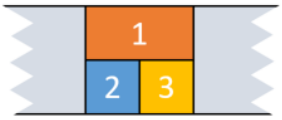
- Quadrat structures themselves do not rotate
- Quadrats may restructure in some cases
- Writing line constrains quadrat boundaries
- Kerning effects apply across quadrats

In the sections that follow, the points being made are illustrated using abstracted structures based on an archetypal writing line in gray with colored boxes for individual hieroglyphic signs or quadrats.

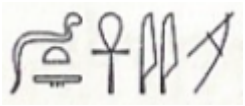
Vertical mode (LTR)	Horizontal mode (LTR)
	
Signs 1, 2, and 3 occur in sequence in a vertical line.	Signs 1, 2, and 3 occur in the same sequence in a line that progresses left-to-right.

Non-rotation

The phenomenon of writing both horizontally and vertically is shared with East Asian writing systems, notably Chinese, Japanese, and Korean. Like these systems, the orthographic units themselves do not rotate when switching between horizontal and vertical layouts.[2] Like the East Asian systems, Egyptian quadrats remain upright regardless of text direction (vertical or horizontal). A quadrat, being and an orthographic unit, retains its internal layout whether in vertical or horizontal mode.

Vertical mode (LTR)	Horizontal mode (LTR)
	
A quadrat with three signs 1, 2, and 3 retains its internal structure in vertical and horizontal layout.	

For example, the cluster  is consistent in both modes:

Vertical mode (RTL)	Horizontal mode (RTL)
	
Extracts from the Stela of Bouto (Dessoudeix, Michel. 2012. *Lettres Égyptiennes II*. Arles: Actes Sud. p. 308)	

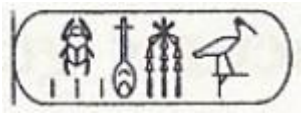
[2] This is not the same as saying that one simply transposes a text unit for unit between horizontal and vertical writing. For example, see W3C Working Group, April 2012, Requirements for Japanese Text Layout. (http://www.w3.org/TR/jlreq/#vertical_writing_mode_and_horizontal_writing_mode), accessed 4/14/2017.

Restructuring

Since the quadrat structure controls do not themselves vary between horizontal and vertical rendering of a text (see previous section), any restructuring required by a context must be done explicitly in the document encoding rather than being expected by the rendering layer. Examples of the need for restructuring occur in both ancient and modern contexts.

Ancient

The structure of a hieroglyph may be reconfigured between horizontal and vertical sections in the same text. In the following example, the fact that the quadrat needs to be explicitly restructured is made more apparent by the inclusion of an additional sign in the horizontal instance:

Vertical mode (LTR)	Horizontal mode (RTL)
	
Extracts from the Stela of Bouto (Dessoudeix 2012: 308)	
	
(1 : 2) * (3 : 4)	1 * 2 * 3 * (4 : 5)

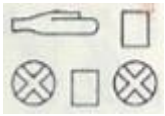
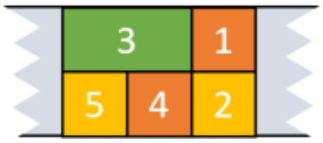
Ancient restructuring is analogous to cases in which a word appears in different spelling in the same text:

First occurrence (LTR)	Second occurrence (RTL)
 *wꜣḏy.t*	 *wꜣḏy.t*
Extracts from the Stela of Bouto (Dessoudeix 2012: 308)	

Such cases require that the two instances of the word are encoded differently.

Modern

Modern editions of Egyptian Hieroglyphic texts frequently use horizontal layout. Editions may also provide a version of the text in vertical layout. This is the case with Dessoudeix's edition of the Stela of Bouto (2012: 308 ff.). The following example shows an extract in the vertical edition and Dessoudeix's horizontal presentation of the very same sign group. The point of note here is that the relationship between the first two signs changes from a horizontal connection to vertical connection while the relationship between the second pair of the same signs remains horizontal.

Vertical mode (RTL)	Horizontal mode (RTL)
	
Extracts from the Stela of Bouto (Dessoudeix 2012: 308, 309)	
	
1 * 2 3 : 4 * 5	1 : 2 3 : 4 * 5

Because the normal case is that quadrat structures are maintained between vertical and horizontal presentations, restructuring cannot be done automatically as a property of the same control set. Therefore, any required restructuring must be done in the document encoding. It is, therefore, out-of-scope for the current proposal.

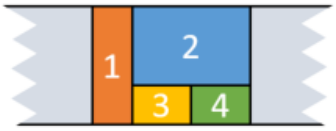
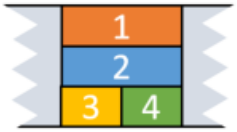
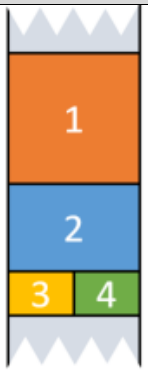
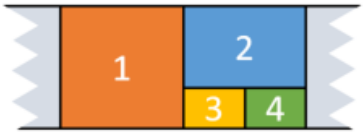
Layout mode indication

The fact that an author of a document may encode a text differently to reflect an original vertical structure or an original horizontal structure raises the need to capture that information somehow. If plain text supported vertical mode in a manner equivalent to the way plain text does support bidirectional text (LTR and RTL), one would expect common directional controls to specify vertical encoding in ways comparable to RIGHT-TO-LEFT OVERRIDE (U+202E), RIGHT-TO-LEFT EMBEDDING (U+202B) etc., our author would be able to use such controls to indicate the original directionality. However, this is not current practice in Unicode and vertical layout is consigned to higher-level protocols. Therefore, indicating the original direction of a text must be done as a higher-level protocol, or as an explicit statement to the reader of a plain text document.

Size constraints

When writing vertically, signs and quadrats are constrained by their width. When writing horizontally, they are constrained by their height. This has an impact on how quadrats are constructed in these modes. When there is a separation between signs that is perpendicular to the line of writing it is a matter of interpretation whether there is a quadrat boundary or a join between adjacent quadrat

members. In the following examples the distinct text encodings for the quadrats are identified with the signs ❶, ❷, and ❸.

Vertical mode (LTR)	Horizontal mode (LTR)
	
The quadrat 1-2-3-4 is compressed to fit within the width of the writing line.	Sufficient width is available so that the quadrat 1-2-3-4 can use its natural proportions.
Structure: ❶ 1 * (2 : 3 * 4)	Structure: ❶ 1 * (2 : 3 * 4) *OR* ❷ 1 2 : 3 * 4
	
Sufficient height is available so that the quadrat 1-2-3-4 can use its natural proportions.	The quadrat 1-2-3-4 is compressed to fit within the height of the writing line.
Structure: ❸ 1 : 2 : 3 * 4 *OR* ❷ 1 2 : 3 * 4	Structure: ❸ 1 : 2 : 3 * 4
	
Due to width constraints, sign 1 is separated from the quadrat 2-3-4.	Due to height constraints, sign 1 is separated from the quadrat 2-3-4.
Structure: ❷ 1 2 : 3 * 4	Structure: ❷ 1 2 : 3 * 4

The three possible structures for the above arrangements are:

- 🅐 1 * (2 : 3 * 4)

  One quadrat: the connection between sign 1 and the rest of the quadrat is horizontal. Fonts that support this structure would display in this form in either vertical or horizontal layout.

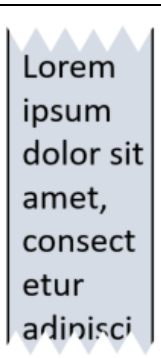
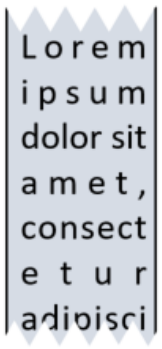
- 🅑 1 2 : 3 * 4

  Two quadrats: sign 1 and signs 2-3-4 are in separate quadrats from the perspective of the text encoding. The layout adapts between horizontal and vertical layout. The size of sign 1 is determined by the font.

- 🅒 1 : 2 : 3 * 4

  One quadrat: the connection between sign 1 and the rest of the quadrat is vertical. Fonts that support this structure would display in this form in either vertical or horizontal layout.

If an edit control treated vertical Egyptian text as narrow columns of horizontal text then it would be possible for the edit control to determine whether two or more quadrats would fit into the available column width. Such a control would be able to achieve the appearance of the structure A example using the structure B text encoding. In this respect, it is similar to layout of horizontal writing systems in narrow columns where text progresses in short horizontal rows:

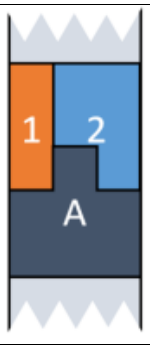
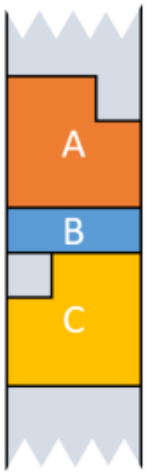
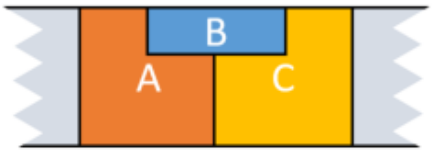
	
Left aligned	Distributed

In some cases, there is sufficient width to accommodate two words in a single line (*dolor sit*), in most cases there is not. It may also be necessary to do emergency line-breaking when a single unit cannot fit within the available width (*consectetur*). For aesthetic purposes, it may be preferable to normalize the width of orthographic units (i.e. words in the above example). To do this, one can either compress or expand the units. In Latin script, such processing is handled at the layout level and not at the font level. However, there is room for the layout to impact the font rendering through justification effects, such as *kashida*[3]. Such effects are out-of-scope for rendering quadrat structures themselves and are not covered by the current proposal.

Kerning

Egyptian scribes were very conscious of the whitespace in their texts. They employed kerning to achieve aesthetic balance. Because adjacency differs between vertical and horizontal layout, kerning effects cannot be the same when the same text is transposed between horizontal and vertical forms.

[3] For details, see https://w3c.github.io/alreq/#h_justification_kashida.

Vertical mode	Horizontal mode
	
Quadrat A protrudes into an available space at the bottom left of sign 2.	Quadrat A aligns next to sign 2 and has space at the top left and top right.
	
Quadrats A, B, and C layout out in vertical sequence without any kerning effect.	Quadrats A and C have space at the top right and top left of their signs respectively. This enables quadrat B to be positioned between them.

Kerning is a layout effect determined by the shapes of adjacent orthographic units in an encoded text. It is out-of-scope for rendering quadrat structures and is not covered by the current proposal.

Examples of kerning

Vertical mode (RTL)	Horizontal mode (RTL)
	
18th Dynasty Tomb of Rekhmire (TT100)	Stela if Ity (BM EA 586)

OpenType strategies

Three strategies for implementing Quadrat structures have been explored as part of this proposal. These are described in this section.

Ligatures solution

For a font that targets a specific repertoire of signs, the easiest approach to the OpenType tables is using ligatures that target each quadrat in the repertoire. The ligature would match the full expression for each quadrat and replace it with a single glyph designed for that quadrat. The ligatures must be ordered so that shorter sequences do not apply before longer sequences. This approach is likely to produce the best quality renderings of each quadrat as they can be designed individually.

For example:

To form the quadrat  , illustrated above, one could use the following <rlig> lookup (in VOLT syntax):

```
G25 TopEnd Aa1 BottomEnd X1 -> G25_TopEnd_Aa1_BottomEnd_X1
```

Dynamic solution

A font that aims to provide maximum coverage of possible quadrats without targeting a specific repertoire must be able to analyze the linear format and produce an acceptable rendering within the font's OpenType layout. Two such prototype fonts have been attempted, one by Mark-Jan Nederhof, the other by Andrew Glass. The description that follows here describes the approach used by Glass, though there are significant overlap in the two solutions. The Glass prototype has been carried far enough to determine that this approach is feasible and can produce acceptable results within the different table size limits of OpenType for the horizontal and vertical joiners as well as the corner insertions. Arbitrary stacking has not yet been implemented in this prototype, but it would be possible to extend it to do so. The Glass prototype has four main phases as follow. For a description of the Nederhof solution see L2/16-210R, appendix C (p. 24 ff.).

Structure processing

The purpose of this phase is to convert the encoded sequences into quadrat groups with all embedding levels marked using control glyphs that are internal to the font.

Parentheses conversion

These rules identify controls bounded by segment begin and end glyphs and mark them with the appropriate embedding level.

Hieroglyph type marking

This rule is one of two substitution rules that touch the entire set of Unicode encoded hieroglyphs. It inserts a marker glyph that identifies the ideal size and aspect ratio of each hieroglyph. Glyph sizes are identified with a two-digit integer that identifies the number of horizontal and vertical units that the glyph measures. Each digit ranges from 1 to 6. The first digit indicates the number of horizontal units. The second denotes the number of vertical units. The Glass prototype uses a unit of 1/6 of a standard glyph width and height. The horizontal unit is slightly larger than the vertical unit.

In the hybrid solution, this lookup can be extended to include the output of ligature rules so that they may be included in subsequent processing.

Quadrat bounding

A small number of rules are needed to mark the beginning and end of each quadrat. These include rules to remove false-positive quadrat beginning and ending glyphs.

Level definition

These rules insert row and column control glyphs to identify the level structures in each quadrat.

Level processing

The purpose of this phase is to establish the target size of all glyphs one embedding level at a time.

Each level in the quadrat is processed in much the same way. The innermost layer, level 2 in the prototype, has reduced number of rules because the size of the target glyphs has limited flexibility. The process for levels 1 and 0 is as follows:

- Count the number of columns in each row of each level
- Map the ideal widths of each glyph to a target size based on the number of columns in the row. These rules take into account the preferred width of other columns
- Count the number of rows in each level
- Map the ideal height of each glyph to a target size based on the number of rows in the quadrat. These rules take into account the preferred height of other rows and constraints on compressing a row if it has inserted signs
- Size row beginning markers to the height of the glyphs in the row
- Size column beginning markers to the width of the glyph in the column
- Sum the widths of all columns in each row
- Determine the maximum width of each row
- Round up columns to the width of the widest row
- Determine the maximum height of the columns of each row
- Sum the heights of each row
- Round up the heights of each target cell to the max height for the row
- Map the size of each target cell to a target glyph size

Final processing

The purpose of this phase is to finalize the glyph sequence based on the entire quadrat as well as to do clean-up of extraneous control glyphs.

Anchor insertion

An anchor glyph is inserted so it will occur before every sized hieroglyph. Having just one anchor glyph reduces the number of rules required to position all of the sized hieroglyphs.

Substitution to sized glyphs

This rule converts all size variants of every Unicode encoded hieroglyph into the target size based on an abstracted target size marker glyph calculated in the level processing steps.

Quadrat width

A marker glyph, calculated in earlier processing is used to substitute the glyph that marks the beginning of a quadrat into a base glyph with the required width.

Corner insertion

These rules insert glyphs that will serve as carriers the corner insertions.

Enclosing

These rules swap empty quadrat width glyphs into cartouche or other enclosing sign extenders if the current quadrat continues a cartouche/enclosure sequence.

Positioning

The purpose of this phase is to assemble all of the quadrat components into the display form.

Row and column scaffolding

Row and column markers are pinned to each other (row to row-end, column to row top, column to column end) to form a structure on which invisible rectangles that will host the sized hieroglyphs are pinned.

Corner insertion

The rectangles that will host the inserted glyphs and glyph groups are pinned to the corners of their hosting rectangles.

Anchor positioning

The anchor glyph is pinned to the center of every hosting rectangle.

Glyph anchoring

This is the final rule, and the only GPOS rule to touch all sized glyphs. It positions the sized-glyphs center to center on the preceding anchor glyph.

Hybrid solution

It is also possible to combine the two preceding solutions to achieve a best of both solution. In this approach, a defined repertoire of quadrats is supported via precomposed target glyphs. These are formed using ligatures processed before any features of the arbitrary solution are triggered, i.e., using the <ccmp> feature. Then all of the lookups of the arbitrary solution are applied, such that any unprocessed sequences may form quadrats using those steps. As a further benefit, the glyphs that result from precomposed glyphs may themselves participate in the lookups used later in the steps of the arbitrary solution.

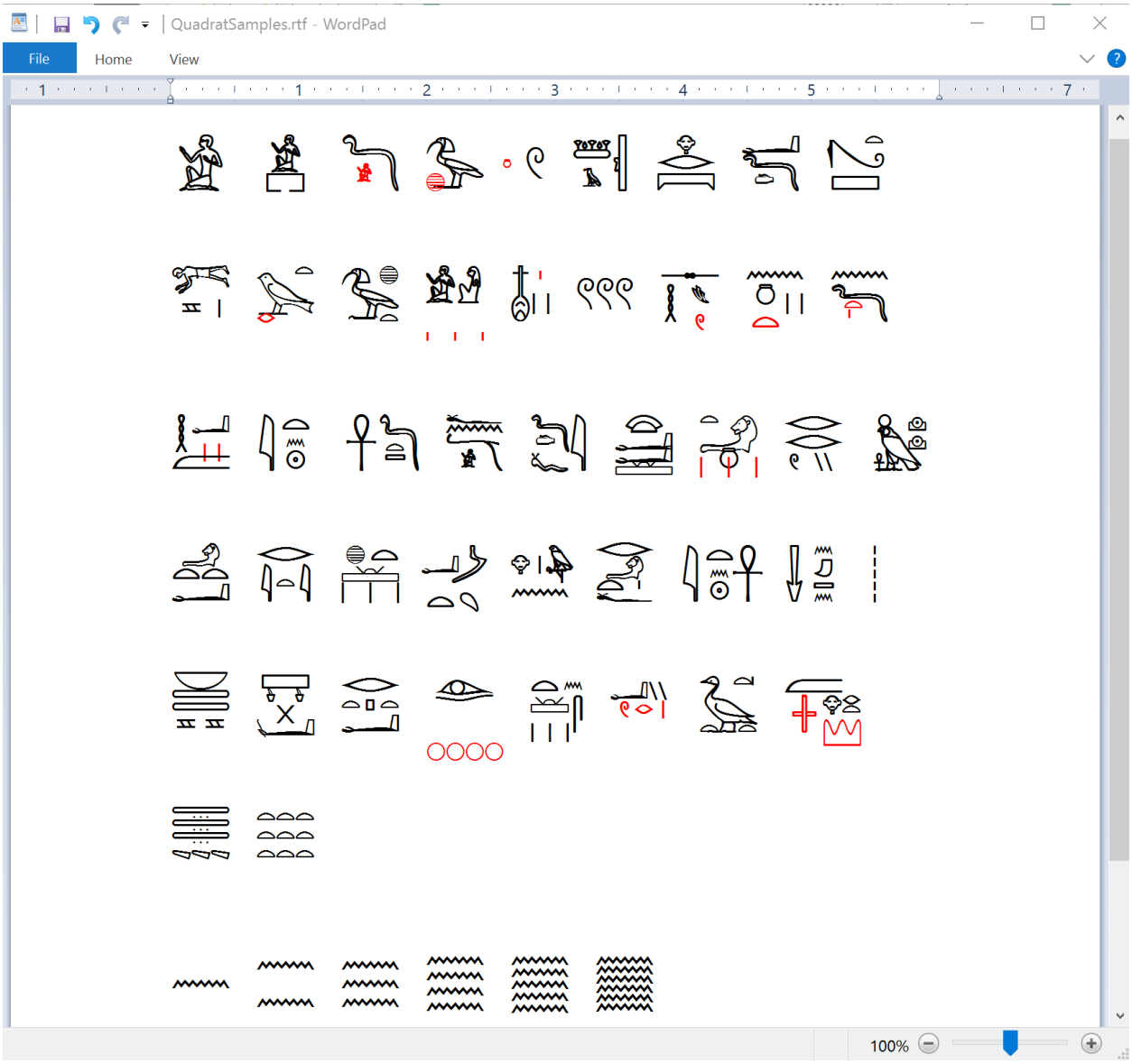
Rendering examples using prototype fonts

Ligature style font

(Source http://hieroglyphseverywhere.blogspot.co.uk/2016/12/web-browser-test-for-hieroglyphic.html)

Dynamic style font

In the following image, Glass's prototype dynamic font has been used to render the set of sample quadrats illustrated in L2/16-232. The font mimics the proposed encoding using Microsoft WordPad on Windows 10 Anniversary Update. The quadrat components shown in red indicate minor font bugs that Glass has not had time to resolve in the font.



Open issues

Future character encoding proposals for Egyptian Hieroglyphs need to consider the option of encoding signs via a sequence of other signs using the structure controls vs. atomic encoding. Participants in the field should engage in a dialog to determine guidelines for when to use a structure and when atomic encoding is preferable.