L2/19-331

Title: A note on OpenType implementation of Ancient Egyptian hieroglyphic text
From: Mark-Jan Nederhof (University of St Andrews)
To: UTC
Date: 2019-09-18

1 Introduction

This note concerns the control characters of Ancient Egyptian, specified in L2/17-112. An earlier note to the UTC, L2/18-236, discussed processing of the syntax, that is, how to find the internal structure of an expression that encodes a fragment of Ancient Egyptian hieroglyphic text. We also mentioned an implementation for use on web pages. This implementation translated Unicode expressions to RES, which is a form of encoding of Ancient Egyptian hieroglyphic text that is strictly more powerful than that of Unicode. The formatting was then realized by a JavaScript implementation of RES that had been developed earlier.[1]

In the current note, we discuss an independent implementation of the formatting of Unicode representations of Ancient Egyptian. It can be used in web pages without the need for client-side JavaScript processing, relying instead on the *liga* feature of OpenType fonts. At present, an OpenType font would be specific to a collection of documents. This makes our implementation only suitable for applications where the collection of documents is relatively stable, as each change to the hieroglyphic encodings in those documents requires recreation of the font. However, the (re)creation of the font is fully mechanical.

The Python implementation[2] presented here relies on FontForge[3] to create composite signs, and on AFDKO[4] to assemble an OpenType font out of a TrueType font and a feature file. Our open-source implementation can be used by those wanting to create their own fonts.

There is proof-of-concept, in L2/16-210R and L2/17-112, for a single OpenType font that could format any encoding, albeit restricted to a certain depth of nesting. This is the subject of ongoing research and will not be discussed further here.

2 Abstract syntax

Ignoring the intricacies of the concrete syntax of Ancient Egyptian in Unicode, we can say that a fragment of text consists of a sequence of outermost top groups. The abstract syntax is inductively defined by:

- A **top group** is a horizontal group, a vertical group, or a basic group.
- A **vertical group** consists of a list of two or more subgroups to be arranged vertically, each of which is a horizontal group or a basic group.
- A **horizontal group** consists of a list of two or more subgroups to be arranged horizontally, each of which is a vertical group or a basic group.
- A **basic group** has a core subgroup, which is either a single sign or an overlay, and, for each of the four corners, optionally a top group to be inserted in that corner.
- An **overlay** consists of two lists, each consisting of one or more signs. The signs in the first list are to be arranged horizontally and the signs in the second list are to be arranged vertically, and the two lists are then stacked upon one another, with their center points coinciding.

[1] `https://mjn.host.cs.st-andrews.ac.uk/egyptian/res`
[2] `https://github.com/nederhof/opentypehiero`
[3] `http://fontforge.github.io/en-US`
[4] `https://github.com/adobe-type-tools/afdko/`

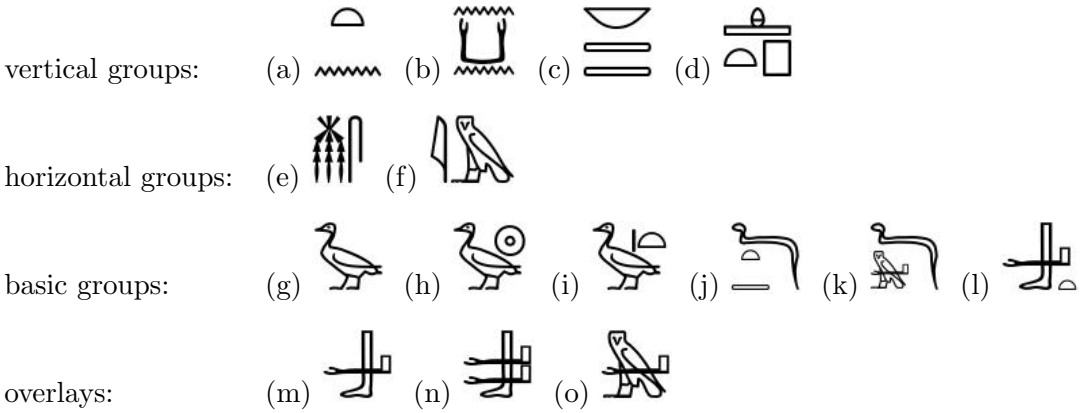

Figure 1: Examples of groups

Examples are given in Figure 1. Note that the vertical group (d) has a subgroup that is a horizontal group. The basic group (i) has an insertion in the top right corner that is an horizontal group. In the basic group (j), the insertion is a vertical group. The basic group (k) has an insertion in the bottom left corner that is a basic group whose core subgroup is an overlay. The basic group (l) has a core subgroup that is an overlay, and there is an insertion in the bottom right corner.

3 Formatting

Formatting of an outer top group is done in two steps:

- First, signs in that group are scaled down if necessary to fit within the available space. This process starts with the smallest nested subgroups.
- Second, as much whitespace as possible is added between subgroups without increasing the total surface area taken up by the outer top group. This process of *padding* is top-down.

In the following, we describe the scaling and padding informally. For details, we refer the reader to the Python code in our implementation.

3.1 Scaling

The space available to a group is determined by a maximum width and a maximum height. These maxima are either ∞ to indicate there is no restriction, or are positive numbers no greater than 1 EM, where EM is the height of a row of text, and of the unscaled 'sitting man' hieroglyph. (We do not consider hieroglyphic text in columns.)

An outermost top group has maximum height 1 EM and maximum width ∞. The latter means that if this outermost top group is an horizontal group consisting of many basic groups or vertical groups next to one another, then it can have any width. However, the maximum width of the subgroups of a vertical group is 1 EM and the maximum height of the subgroups of an horizontal group is 1 EM. The maximum width and maximum height of a top group inserted in a corner of a basic group is determined by the shape of the core subgroup. This is discussed further in Section 3.3.

The width of an horizontal group is the sum of the widths of the subgroups, plus a certain distance between consecutive subgroups, and its height is the maximum of the heights of the subgroups. The height and width of a vertical group are defined analogously.

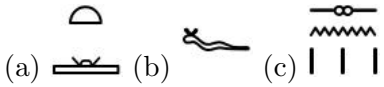

Figure 2: Examples of padding

As specified before, an overlay consists of a list of signs arranged horizontally, and a list of signs arranged vertically. Unlike ordinary horizontal and vertical groups, we do not insert space between the signs in either of these two lists. This means that the width of the horizontal list is simply the sum of the widths of the individual signs, and the height, as usual, is the maximum of the heights. The dimensions of the vertical list are analogous. The total width and height of an overlay are the maxima of the widths and heights, respectively, of the horizontal list and the vertical list. In the formatting, the geometric center of the horizontal list coincides with the geometric center of the vertical list.

If the width and/or height of a group exceed the available space, then all signs within it are scaled down, including the distances between consecutive subgroups, to exactly fit within the available space. Note that a single sign may be scaled down multiple times, as it may be part of a group that in turn is a subgroup of a larger group, etc., and each of these larger and larger groups may trigger down-scaling of all the signs in them. Signs are never scaled up, so they never appear bigger than their 'natural' size in the font.

3.2 Padding

After scaling, whitespace is added inside groups, to center individual signs, and spread out horizontally the subgroups of an horizontal group, and spread out vertically the subgroups of a vertical group.

More precisely, groups are formatted within a rectangle whose size is computed after scaling as in Section 3.1. For an outer top group, the height of this rectangle is the height of a line, so 1 EM, and the width is the width of that top group.

A horizontal group that is to be formatted within a certain rectangle is divided into smaller rectangles horizontally, one for each subgroup, and one for each occurrence of whitespace between subgroups; the width of the rectangle of a subgroup is the width of that subgroup, and leftover horizontal whitespace is equally divided over the whitespace that occurs between the subgroups. The formatting of a vertical group is analogous.

A top group that is an insertion in a basic group is formatted within a rectangle determined by the shape of the core group in that basic group. We will return to this in Section 3.3.

A sign that is to be formatted within a certain rectangle that is bigger than the sign itself is centered horizontally and vertically, that is, any leftover horizontal whitespace is divided equally over padding to the left and to the right of the sign, and any leftover vertical whitespace is divided equally over padding above and below.

We give examples to illustrate this. In the vertical group of Figure 2(a), extra whitespace is added between the top-most and the bottom-most signs, to make the total padded height 1 EM. The top-most sign is centered horizontally, with equal amounts of added whitespace on the left and on the right. Similarly, in Figure 2(b) padding is added above and below the flat sign to center it within the row of text. In another vertical group, in Figure 2(c), the bottom-most subgroup is a horizontal group consisting of three strokes, represented as individual signs. Extra whitespace is equally divided over the two occurrences of whitespace between the strokes, so that the padded width of that subgroup becomes equal to the maximum of the widths of the two signs above it.

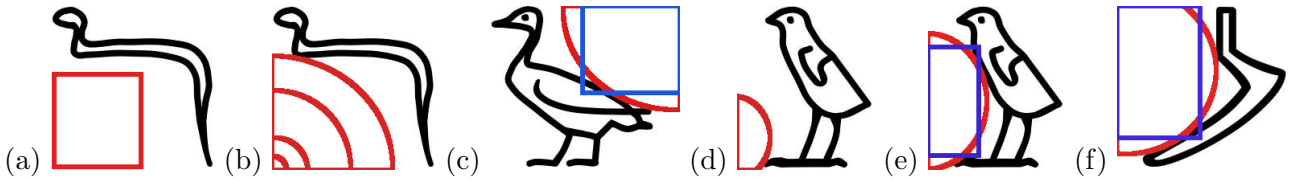

Figure 3: The space available for insertion

3.3 Insertion

For each of the four corner insertions, the available space is fixed for each individual sign that may appear as core subgroup. For example, for the cobra, the available space may be as indicated in Figure 3(a). Note that this space is here a square that is strictly in the bottom left corner of the bounding box of the cobra.

As it would be overly cumbersome to determine this space for each corner of each sign manually, there is an automatic analysis of the shapes of the signs, which is run once for each font. We will refer to this as a *static* analysis. This differs from the semantics of RES, mentioned before, in which this analysis of shapes is done *dynamically*, that is, upon rendering of a given group. In RES, this analysis cannot be done statically, as the core of a group with corner insertions can be an arbitrary group itself.

In the case of bottom-left corner insertion in the cobra, the static analysis draws circles of increasing radius with the bottom left corner as center. The radius of the smallest such circle that intersects with the cobra, as illustrated in Figure 3(b), determines the available space for corner insertion. The square that is then reserved for the insertion has a size that is chosen to be slightly smaller than that radius, as illustrated earlier in Figure 3(a).

However, the cobra is somewhat exceptional in that, as it occurs in most fonts, it offers sufficient space for typical insertions. For many other signs, not enough space is available if the shape is determined as above. For this reason, we add a strip of whitespace to the signs, on the left and/or on the right, if there are insertions on the left and/or on the right, respectively. Put differently, we artificially add space to the bounding box around the shape. This is done during the static analysis, and also when the groups are formatted. This is illustrated by Figure 3(c), for top-right corner insertion, where as before, the size of the square that is reserved for the inserted group is slightly smaller than the radius of the smallest circle intersecting the sign. It should be observed that this is not inconsistent with the Ancient Egyptian writing systems. The notion of 'bounding box' is a modern invention, which would have been alien to the artists in ancient times, and there is therefore no necessity to adhere to strict bounding boxes in formatting hieroglyphic text.

For a number of signs, we make a further adjustment, which is that the centers of the circles need not stay exactly in one of the four corners, but may be moved slightly up or down, from a given initial position, in order to maximize the radius. The initial position is one fifth of the height below the top of the bounding box for top-left corner insertion and one fifth of the height above the bottom of the bounding box for bottom-left corner insertion. This is illustrated starting with Figure 3(d). The analysis incrementally moves the center of the circles up, as it finds that the minimum radius to intersect with the sign can in this way be increased, which eventually results in Figure 3(e).

It should be noted that this adjustment is partly a consequence of merging two of the *five* kinds of insertions of PLOTTEXT, which had a dedicated kind of bottom-left corner insertion for birds, where signs were placed just *above* the birds' feet, so not strictly in the bottom-left corner.[5] However, we use the same principle for top-left corner insertion for a number of signs, as exemplified by Figure 3(f). Note that the

[5]Norbert Stief. *Hieroglyphen, Koptisch, Umschrift, u.a. – Ein Textausgabesystem* –, in: Göttinger Miszellen 86, p. 37-44, 1985

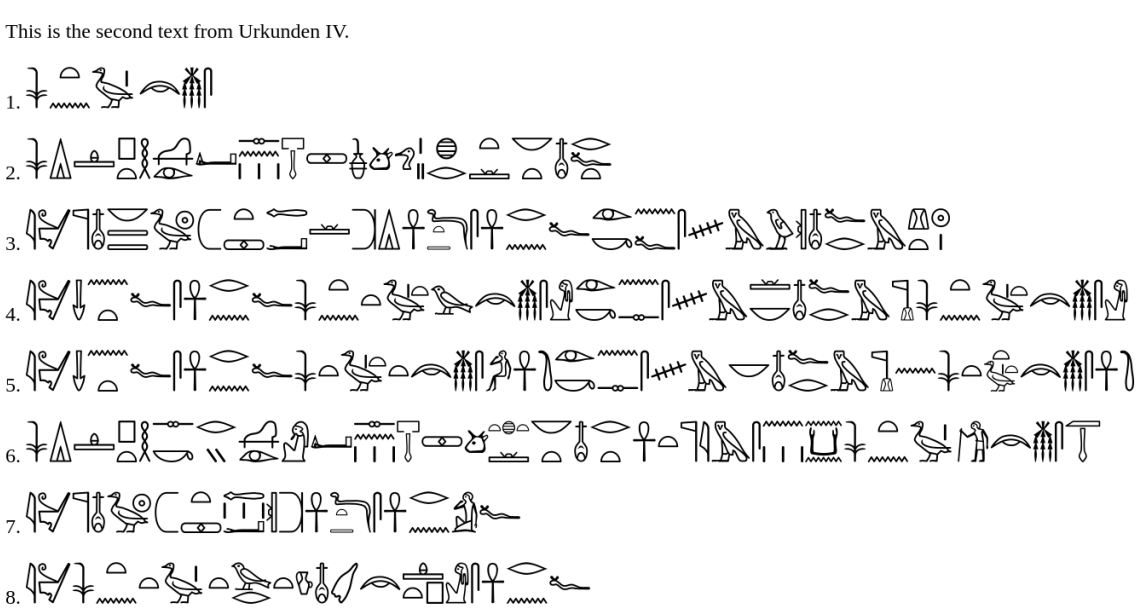

Figure 4: Screenshot of Chromium (version 76 on Ubuntu 16.04).

rectangles now reserved for the insertion need no longer be square.

The above static analysis is not applicable however if the core of a basic group is an overlay, as the overlaid horizontal and vertical lists of signs can be arbitrary. One option would be to introduce dynamic analysis as in the case of RES, but in the interest of processing speed, the current implementation applies the following heuristics:

- The bounding box around the overlay is extended by extra whitespace at the left and/or at the right if there are corner insertions at the left and/or right, much as before.
- The width and height of each rectangle reserved for a corner insertion are one fourth of the width and the height, respectively, of the extended bounding box.

Figure 1(l) is an example.

4 Extended example

A screenshot of an hieroglyphic text is given in Figure 4. The web page, viewed using Chromium, contains the Unicode representation of an hieroglyphic text, rendered using an OpenType font that was created by our implementation.