The eutofont font format

William Overington

29 December 2003

The eutofont format is an experimental font format which permits a font to be encoded as a sequence of Private Use Area characters in the range U+EAC0 to U+EAFF within a Unicode plain text file.

The eutofont format is devised with two niche application areas in mind, though the eutofont format may potentially find other applications. The first is for providing user defined fonts for use with the DVB-MHP (Digital Video Broadcasting - Multimedia Home Platform) system. A eutofont format font can be produced using a plain text editor on a PC and can be broadcast in a plain text file. The DVB-MHP system uses programs written in the Java programming language, those programs being broadcast, so a eutofont format font can be processed by such a Java program so as to provide font capability within the program. The second is for general computing applications where there is a need for fonts where the glyphs are not monochrome. The eutofont format allows colouring of individual contours, either in indexed palette form, so that background colour, foreground colour, first decoration colour, second decoration colour and so on are specified externally by the application or by absolute colours in rgb (red, green, blue) format. The former is for general use. The latter permits items such as ornaments for holly with berries to be specified with green holly and red berries.

Please know that at the time of writing the eutofont format has not been tested in use. It is a theoretical development. However, it has been designed with the experience of producing the eutocode graphics format, which has been tested in practice.

Accordingly, it is unclear whether the format in this document will need to be altered or whether it will be suitable as a long term document explaining the structure of a eutofont format font. However, in view of the fact that publication of this document is towards the end of 2003, it is convenient to regard this format as "eutofont 2003 format" and whether eutofont 2003 format becomes just a stage in research and development of a later format or whether eutofont 2003 format has long term usefulness as a font format can depend upon how eutofont 2003 format performs in application. However, the name eutofont 2003 format can be used to refer to the format described in this document unambiguously.

The code points have been allocated within five groups.

U+EAC0 .. U+EAC7 Codes affecting the whole font.
U+EAC8 .. U+EACF Codes affecting a whole glyph.
U+EAD0 .. U+EADF Codes affecting a whole contour.
U+EAE0 .. U+EAEF Decimal data and minus sign.
U+EAF0 .. U+EAFF Hexadecimal data.

The eutofont format is designed to have the following registers which can have data entered into them.

The ad register.

This starts with a value of 0.

The codes U+EAE0 .. U+EAE9 cause the value of ad to be multiplied by 10 and then have the appropriate value (0 .. 9) added.

The ad_sign register.

This starts with a value of '+'.

The code U+EAEF causes the value of ad_sign to become '-'.

The ah register.

This starts with a value of 0.

The codes U+EAF0 .. U+EAFF cause the value of ah to be multiplied by 16 and then have the appropriate value (0 .. 15) added.

Various codes use the current values of ad and ad_sign or the current value of ah. Some are used to provide data directly and some set other eutofont variables.

Some other codes use data from those other eutofont variables.

Some other codes do not use data at all, but simply have an action.

Using ad causes ad to be set to 0 and ad_sign to be set to '+'.

Using ah causes ah to be set to 0.

The other eutofont variables are x, y, r, g, b. The x and y variables are used by the commands to start a contour, place an on-curve point and place an off-curve point.

Using any of x, y, r, g, b does not cause them to be reset.

Mapping of a glyph can be carried out using either decimal or hexadecimal data. This is for the convenience of fontmakers, the mapping in the Java program is to a number, regardless of whether it was input as decimal or hexadecimal.

The definition of a font in eutofont format

The definition of a font is intended to start by setting a value for the top of the font in font units. The default value is 2048 font units. However, if the value used is 2048, explicit setting to 2048 can be used if desired for aesthetic reasons. The lower limit of a font may be set. The default value is 0, and only 0 and negative values are valid. However, if the value used is 0, explicit setting to 0 can be used if desired for aesthetic reasons.

Next a glyph is started. Glyphs are numbered. Glyph 0 is the default glyph for an undefined character, glyph 1 is not used at present, glyph 2 is for the space character and font characters generally start at glyph 3. Glyphs should be numbered sequentially from 3 upwards. Applications should be tolerant over all glyphs not being numbered and should treat a glyph numbered as 0 as the next available glyph with an index number of at least 3 unless it is the first glyph received as that will be the glyph to use for an undefined glyph. This seems to imply the need for a flag in the application program to say whether an undefined glyph has already been received.

Within a glyph the point size of the glyph may be specified. This is an option and may perhaps only rarely be used. In most fonts it will not be used.

Within a glyph the width of the glyph needs to be specified.

Within a glyph a collection of contours may be specified, each with its own individual colour. The eutofont format font does not use clockwise and counter clockwise glyphs as such, designating instead the colour of the contour. However, as fonts might be exported to other font formats, fontmakers might like to draw contours for background colour contours in a counter clockwise direction for maximum compatibility.

Contours are not individually numbered within the definition of a eutofont format font.

A contour must be closed and that closure is carried out when the end of contour character is used. There is no need to repeat the data for the start point of the contour in order to complete the contour.

Within a glyph a glyph may be mapped to a code point. Mapping to a code point may be carried out anywhere within a glyph. It is a matter of style as to whether it is done at the start of the glyph definition or at the end. It could indeed be done anywhere within a glyph definition, even within a contour definition, yet fontmakers are asked to only map a glyph at just after the start or just before the end of a glyph definition and not within the definition of a contour.

The colour information of contours

Each contour needs to have colouring information. It is recommended that colouring information for a contour should specifically be included in a eutofont font file. However, applications should assume a contour to be in foreground colour unless otherwise stated. This is regarded as being helpful to a forgetful fontmaker, not as a reason to omit defining the colour of a contour specifically. Colouring of a contour may be anywhere within the defining of a contour. However it is

recommended that colouring either takes place just after a contour is started or just before it is ended.

Colouring may be in terms of background colour, foreground colour, first decoration colour, second decoration colour and so on with the colours actually used in display determined by the application which is using the font, or may be in explicit colours supplied by the font.

When explicit colouring is being used the colour used is set using red, green, blue values. Thus the colour of each contour may be defined within the contour if so desired. However, where only one explicit colour is being used, that colour need not be defined within every contour as the colour will be retained within the r, g and b variables. However, where two or more explicit colours are being used the colours would need to be redefined every time a colour different from the last used colour is used.

This produced a dilemma in the design process of the eutofont specification. Suppose that a font is being produced where the letters are each in foreground colour and each letter has decoration of holly and berries and it is desired to make the holly always explicitly green and the berries always explicitly red. As the system has been thus far explained, with each glyph using green and red the colours green and red would need to be redefined within each glyph. The glyph could define one or more green contours and one or more red contours so where more than one contour of any one explicit colour were being used within a glyph there would be an efficiency gained by the fact that explicit colour information is retained until it is explicitly changed. So a mechanism to add an internal palette of explicit colours would be helpful. However, it adds extra complexity into implementing the eutofont system in an application, bearing in mind that for some fonts the use of multiple coloured glyphs might not use explicit colours at all.

On balance it was decided to add the feature of an internal palette of explicit colours. This is done using two code points.

U+EADC, decimal 60124, DECIMAL SET CURRENT CONTOUR TO BE IN THE EXPLICIT INTERNAL PALETTE INDEXED ITEM ABSOLUTELY

U+EADD, decimal 60125, DECIMAL SET EXPLICIT INTERNAL PALETTE INDEXED ITEM TO BE THE CURRENT COLOUR ABSOLUTELY

The Explicit Internal Palette should use indexed items from 11 upwards. This is to avoid confusion. In principle, confusion should not occur as using 2 to colour a contour in first decoration colour uses U+EAD7, decimal 60119, DECIMAL SET CURRENT CONTOUR TO BE IN INDEXED PALETTE COLOUR whereas setting a contour to be coloured using a colour from the Explicit Internal Palette would use U+EADC DECIMAL SET CURRENT CONTOUR TO BE IN THE EXPLICIT INTERNAL PALETTE INDEXED ITEM ABSOLUTELY and the two palettes are different palettes anyway. However, using indexes from 11 upwards for the Explicit Internal Palette will helpfully assist in avoiding mistakes in discussions and so on. A deliberate design decision was taken not to combine the two palettes into one run of index numbers from 0 upwards. Certainly 11 is a two-digit number and that will mean more characters being used, yet an application might have ten palette colours as there is at least one graphics package available which uses ten colours in its preset themed selections. Certainly, the two palettes are separate and so index numbers would not clash. Anyway it is a suggestion and

fontmakers need not follow it if file size is an issue as the Explicit Internal Palette should be implemented as starting at index 0.

The Explicit Internal Palette could be defined at the start of the font before any glyphs are defined, or colours could be defined the first time that they are used, as desired. However, it might be best to have a practice that where an explicit internal palette is being used that the colours are all defined before the first glyph is defined. This would, in effect, produce an explicit internal palette definition section near the start of the file.

The explicit internal palette also has some preset colouring indicator codes. These are used to indicate such effects as metallic ink to the application. It is for the application to decide how to deal with such colouring information, for example using a flat colour or some special effect. The list is as follows.

701 metallic ink gold
702 metallic ink silver
703 metallic ink copper
704 metallic ink rhodium
705 metallic ink turquoise

Using the Quest text font to prepare a eutofont font

In preparing a eutofont format font file as a Unicode plain text file, a fontmaker may enter the Private Use Area codes as he or she chooses. However, the Quest text font, (which is a TrueType font for use with ordinary text editors), from version 1.63, has glyphs for all of the code points mentioned below with the glyphs intended to be meaningful in the context of the eutofont specification. One point to note is that two stylized F and two stylized H augmentations are used in the glyphs. Where a stylized F or a stylized H is used, the choice of which version of each is used is just an artistic decision. Where the basic design of the glyph fills the area above the y axis of the glyph, the smaller version of the F or H is used: where the basic design of the glyph does not fill the area above the y axis of the glyph, the large version of the F or H is used. F is used to denote font and H to denote font hexadecimal. The H is only used on those glyphs related to hexadecimal numbers. For the avoidance of doubt, the glyphs used in the Quest text font are solely for fontmaking convenience, the symbols used are not used when the font is in use in an application. If someone is preparing a eutofont format font using a system other than a plain text editor in conjunction with the Quest text font, then the symbols used in the Quest text font are immaterial. The inclusion of those symbols in an illustration in this document is for the convenience of those readers who choose to use them.

Supplementary note of 8 May 2004

An error in the original published version of this document, namely stating “Using ad causes ad to be set to 0 and ad_sign to be set to ‘-’. “ has been corrected to “Using ad causes ad to be set to 0 and ad_sign to be set to ‘+’. “ in this published version of the document.

Here are the code point allocations.

U+EAC0, decimal 60096, DECIMAL TOP OF FONT (value in font units, default is 2048)
U+EAC1, decimal 60097, DECIMAL LOWER LEVEL OF FONT (value in font units, default is 0, only 0 and negative values valid)

U+EAC8, decimal 60104, DECIMAL START A GLYPH DEFINITION
U+EAC9, decimal 60105, DECIMAL POINT SIZE OF CURRENT GLYPH (optional, default is 0. 0 denotes all sizes)
U+EACA, decimal 60106, DECIMAL WIDTH OF CURRENT GLYPH

U+EACC, decimal 60108, DECIMAL MAP CURRENT GLYPH
U+EACD, decimal 60109, HEXADECIMAL MAP CURRENT GLYPH

U+EACF, decimal 60111, END OF DEFINITION OF CURRENT GLYPH

U+EAD0, decimal 60112, DECIMAL FONT DATA X
U+EAD1, decimal 60113, DECIMAL FONT DATA Y

U+EAD4, decimal 60116, START A CONTOUR DEFINITION (uses current values of x and y)
U+EAD5, decimal 60117, POINT ON THE CONTOUR (uses current values of x and y)
U+EAD6, decimal 60118, POINT OFF THE CONTOUR (uses current values of x and y)
U+EAD7, decimal 60119, DECIMAL SET CURRENT CONTOUR TO BE IN INDEXED PALETTE COLOUR
0 is background, 1 is foreground, 2, 3, 4 are decoration colours.
The colours are set in the application. Suggestions are red, green, blue for decoration unless other colours are specifically selected within the application.
Use of -1 is recommended internally in an application if the colour is absolute, but is not signalled using this code.
U+EAD8, decimal 60120, SET CURRENT CONTOUR TO BE IN THE CURRENT COLOUR ABSOLUTELY
U+EAD9, decimal 60121, DECIMAL RED COMPONENT OF CURRENT COLOUR
U+EADA, decimal 60122, DECIMAL GREEN COMPONENT OF CURRENT COLOUR
U+EADB, decimal 60123, DECIMAL BLUE COMPONENT OF CURRENT COLOUR

U+EADC, decimal 60124, DECIMAL SET CURRENT CONTOUR TO BE IN THE EXPLICIT INTERNAL PALETTE INDEXED ITEM ABSOLUTELY

U+EADD, decimal 60125, DECIMAL SET EXPLICIT INTERNAL PALETTE INDEXED ITEM TO BE THE CURRENT COLOUR ABSOLUTELY

U+EADF, decimal 60127, END OF DEFINITION OF CURRENT CONTOUR

U+EAE0, decimal 60128, FONT DECIMAL 0
U+EAE1, decimal 60129, FONT DECIMAL 1
U+EAE2, decimal 60130, FONT DECIMAL 2
U+EAE3, decimal 60131, FONT DECIMAL 3
U+EAE4, decimal 60132, FONT DECIMAL 4
U+EAE5, decimal 60133, FONT DECIMAL 5
U+EAE6, decimal 60134, FONT DECIMAL 6
U+EAE7, decimal 60135, FONT DECIMAL 7
U+EAE8, decimal 60136, FONT DECIMAL 8
U+EAE9, decimal 60137, FONT DECIMAL 9

U+EAEF, decimal 60143, FONT DECIMAL MINUS

U+EAF0, decimal 60144, FONT HEXADECIMAL 0
U+EAF1, decimal 60145, FONT HEXADECIMAL 1
U+EAF2, decimal 60146, FONT HEXADECIMAL 2

U+EAF3, decimal 60147, FONT HEXADECIMAL 3
U+EAF4, decimal 60148, FONT HEXADECIMAL 4
U+EAF5, decimal 60149, FONT HEXADECIMAL 5
U+EAF6, decimal 60150, FONT HEXADECIMAL 6
U+EAF7, decimal 60151, FONT HEXADECIMAL 7
U+EAF8, decimal 60152, FONT HEXADECIMAL 8
U+EAF9, decimal 60153, FONT HEXADECIMAL 9
U+EAFA, decimal 60154, FONT HEXADECIMAL A
U+EAFB, decimal 60155, FONT HEXADECIMAL B
U+EAFC, decimal 60156, FONT HEXADECIMAL C
U+EAFD, decimal 60157, FONT HEXADECIMAL D
U+EAFE, decimal 60158, FONT HEXADECIMAL E
U+EAFF, decimal 60159, FONT HEXADECIMAL F

Here is a display of the authoring-time glyphs for the eutofont 2003 format which are included in the Quest text font from version 1.63 of the Quest text font. These symbols can be useful in preparing a eutofont format font within a plain text file.

Here is an attempt to encode a small test font in eutofont format.

The glyph designs are copied from the Quest text font.

They are the undefined glyph and the glyphs for U+0020 SPACE, U+25A0 BLACK SQUARE and U+25A1 WHITE SQUARE.

A diagram showing the attempt set in glyphs from the Quest text font version 1.63 follows on the next page.

The attempt is displayed in nine blocks for convenience of explanation and so as to produce a clear diagram.

The top of the font is 2048 font units, the lower level is -768 font units.

U+EAE2 U+EAE0 U+EAE4 U+EAE8 U+EAC0 U+EAEF U+EAE7 U+EAE6 U+EAE8 U+EAC1

Glyph 0, the undefined glyph, is 2048 font units wide. It has one contour. The contour has a start point of (0,0), an off-curve point at (0,2048), an on-curve point at (2048,2048), an on-curve point at (2048, 1024), an on-curve point at (1024,1024) and an on-curve point at (1024,0). The contour is in foreground colour.

U+EAE0 U+EAC8 U+EAE2 U+EAE0 U+EAE4 U+EAE8 U+EACA U+EAE0 U+EAD0 U+EAE0 U+EAD1 U+EAD4 U+EAE2 U+EAE0 U+EAE4 U+EAE8 U+EAD1 U+EAD6 U+EAE2 U+EAE0 U+EAE4 U+EAE8 U+EAD0 U+EAD5 U+EAE1 U+EAE0 U+EAE2 U+EAE4 U+EAD1 U+EAD5

U+EAE1 U+EAE0 U+EAE2 U+EAE4 U+EAD0 U+EAD5 U+EAD1 U+EAD5 U+EAE1 U+EAD7 U+EADF U+EACF

Glyph 2, the space glyph, is 1024 font units wide. It has no contours. It is mapped to U+0020.

U+EAE2 U+EAC8 U+EAE1 U+EAE0 U+EAE2 U+EAE4 U+EACA U+EAF2 U+EAF0 U+EACD U+EACF

Glyph 3, a black square, is 2048 font units wide. It has one contour. The contour has a start point of (0,0), an on-curve point at (0,1792), an on-curve point at (1792,1792) and an on-curve point at (1792,0) and is coloured in foreground colour. The glyph is mapped to U+25A0.

U+EAE3 U+EAC8 U+EAE2 U+EAE0 U+EAE4 U+EAE8 U+EACA U+EAE0 U+EAD0 U+EAE0 U+EAD1 U+EAD4 U+EAE1 U+EAE7 U+EAE9 U+EAE2 U+EAD1 U+EAD5 U+EAE1 U+EAE7 U+EAE9 U+EAE2 U+EAD0 U+EAD5 U+EAD1 U+EAD5 U+EAE1 U+EAD7 U+EADF

U+EAF2 U+EAF5 U+EAFA U+EAF0 U+EACD U+EACF

Glyph 4, a white square, is 2048 font units wide. It has two contours. The first contour has a start point of (0,0), an on-curve point at (0,1792), an on-curve point at (1792,1792) and an on-curve point at (1792,0) and is coloured in foreground colour. The second contour has a start point of (256,256), an on-curve point at (1536,256), an on-curve point at (1536,1536) and an on-curve point at (256,1536) and is coloured in background colour. The glyph is mapped to U+25A1.

U+EAE4 U+EAC8 U+EAE2 U+EAE0 U+EAE4 U+EAE8 U+EACA U+EAE0 U+EAD0 U+EAE0 U+EAD1 U+EAD4 U+EAE1 U+EAE7 U+EAE9 U+EAE2 U+EAD1 U+EAD5 U+EAE1 U+EAE7 U+EAE9 U+EAE2 U+EAD0 U+EAD5 U+EAD1 U+EAD5 U+EAE1 U+EAD7 U+EADF

U+EAE2 U+EAE5 U+EAE6 U+EAD0 U+EAE2 U+EAE5 U+EAE6 U+EAD1 U+EAD4 U+EAE1 U+EAE5 U+EAE3 U+EAE6 U+EAD0 U+EAD5 U+EAE1 U+EAE5 U+EAE3 U+EAE6 U+EAD1 U+EAD5 U+EAE2 U+EAE5 U+EAE6 U+EAD0 U+EAD5 U+EAE0 U+EAD7 U+EADF

U+EAF2 U+EAF5 U+EAFA U+EAF1 U+EACD U+EACF