NAME

Font::TTF::Glyph − Holds a information for a single glyph

DESCRIPTION

This is a single glyph description as held in a TT font. On creation only its header is read. Thus you can get the bounding box of each glyph without having to read all the other information.

INSTANCE VARIABLES

In addition to the named variables in a glyph header (`xMin` etc.), there are also all capital instance variables for holding working information, mostly from the location table.

Variables for all glyphs:

The standard attributes each glyph has are:

numberOfContours
: For simple glyphs this will be the count of contours. For compound glyphs this will be −1.

xMin
yMin
xMax
yMax
: These identify the bounding box of the glyph.

There are also other, derived, instance variables for each glyph which are read when the whole glyph is read (via `read_dat`):

instLen
: Number of bytes in the hinting instructions (Warning this variable is deprecated, use `length($g−{`'hints'})> instead).

hints
: The string containing the hinting code for the glyph

Variables for simple glyphs (numberOfContours >= 0):

endPoints
: An array of endpoints for each contour in the glyph. There are `numberOfContours` contours in a glyph. The number of points in a glyph is equal to the highest endpoint of a contour.

numPoints
: This is a generated value which contains the total number of points for this simple glyph.

There are also a number of arrays indexed by point number:

flags
: The flags associated with reading this point. The flags for a point are recalculated for a point when it is `updated`. Thus the flags are not very useful. The only important bit is bit 0 which indicates whether the point is an 'on' curve point, or an 'off' curve point.

x The absolute x co-ordinate of the point.

y The absolute y co-ordinate of the point

Variables for compound glyphs (numberOfContours == −1):

metric
: This holds the component number (not its glyph number) of the component from which the metrics for this glyph should be taken.

comps
: This is an array of hashes for each component. Each hash has a number of elements:

  glyph
  : The glyph number of the glyph which comprises this component of the composite. NOTE: In some badly generated fonts, `glyph` may contain a numerical value but that glyph might not actually exist in the font file. This could occur in any glyph, but is particularly likely for glyphs that have no strokes, such as SPACE, U+00A0 NO-BREAK SPACE, or U+200B ZERO WIDTH

SPACE.

args
: An array of two arguments which may be an x, y co-ordinate or two attachment points (one on the base glyph the other on the component). See flags for details.

flag The flag for this component

scale
: A 4 number array for component scaling. This allows stretching, rotating, etc. Note that scaling applies to placement co-ordinates (rather than attachment points) before locating rather than after.

numPoints
: This is a generated value which contains the number of components read in for this compound glyph.

Private instance variables:

INFILE (P)
: The input file form which to read any information

LOC (P)
: Location relative to the start of the glyf table in the read file

BASE (P)
: The location of the glyf table in the read file

LEN (P)
: This is the number of bytes required by the glyph. It should be kept up to date by calling the `update` method whenever any of the glyph content changes.

OUTLOC (P)
: Location relative to the start of the glyf table. This variable is only active whilst the output process is going on. It is used to inform the location table where the glyph is located, since the glyf table is output before the loca table due to alphabetical ordering.

OUTLEN (P)
: This indicates the length of the glyph data when it is output. This more accurately reflects the internal memory form than the `LEN` variable which only reflects the read file length. The `OUTLEN` variable is only set after calling `out` or `out_dat`.

Editing

If you want to edit a glyph in some way, then you should read_dat the glyph, then make your changes and then update the glyph or set the `$g->{' isDirty'}` variable. The application must ensure that the following instance variables are correct, from which update will calculate the rest, including the bounding box information.

```
numPoints
numberOfContours
endPoints
x, y, flags         (only flags bit 0)
instLen
hints
```

For components, the numPoints, x, y, endPoints & flags are not required but the following information is required for each component.

```
flag                (bits 2, 10, 11, 12)
glyph
args
scale
metric              (glyph instance variable)
```

METHODS

Font::TTF::Glyph−>new(%parms)

Creates a new glyph setting various instance variables

`$g`**−>read**

Reads the header component of the glyph (numberOfContours and bounding box) and also the glyph content, but into a data field rather than breaking it down into its constituent structures. Use read_dat for this.

`$g`**−>read_dat**

Reads the contents of the glyph (components and curves, etc.) from the memory store `DAT` into structures within the object.

`$g`**−>out($fh)**

Writes the glyph data to outfile

`$g`**−>out_xml($context,** `$depth`**)**

Outputs an XML description of the glyph

`$g`**−>dirty($val)**

This sets the dirty flag to the given value or 1 if no given value. It returns the value of the flag

`$g`**−>update**

Generates a `$self−{`'DAT'`}>` from the internal structures, if the data has been read into structures in the first place. If you are building a glyph from scratch you will need to set the instance variable `' isDirty'`.

`$g`**−>update_bbox**

Updates the bounding box for this glyph according to the points in the glyph

`$g`**−>maxInfo**

Returns lots of information about a glyph so that the `maxp` table can update itself. Returns array containing contributions of this glyph to maxPoints, maxContours, maxCompositePoints, maxCompositeContours, maxSizeOfInstructions, maxComponentElements, and maxComponentDepth.

`$g`**−>empty**

Empties the glyph of all information to the level of not having been read. Useful for saving memory in apps with many glyphs being read

`$g`**−>get_points**

This method creates point information for a compound glyph. The information is stored in the same place as if the glyph was not a compound, but since numberOfContours is negative, the glyph is still marked as being a compound

`$g`**−>get_refs**

Returns an array of all the glyph ids that are used to make up this glyph. That is all the compounds and their references and so on. If this glyph is not a compound, then returns an empty array.

Please note the warning about bad fonts that reference nonexistent glyphs under INSTANCE VARIABLES above. This function will not attempt to filter out nonexistent glyph numbers.

BUGS

- The instance variables used here are somewhat clunky and inconsistent with the other tables.
- `update` doesn't re-calculate the bounding box or `numberOfContours`.

AUTHOR

Martin Hosken <http://scripts.sil.org/FontUtils>.

LICENSING