**SVG Glyphs in OpenType**

**Bringing Rich Graphic Expressivity to Fonts**

Unofficial Editor’s Draft · July 24, 2013

Sairus Patel, Adobe Systems · Cameron McCormack & Edwin Flores, Mozilla Corporation

**Document abstract and status**

This document proposes a way for Open Font Format/OpenType fonts (CFF or TrueType) to optionally include glyphs defined as SVG. This would allow for color, gradients, animation and other aspects of SVG’s rich graphics model.

This document is an unofficial editor’s draft that has arisen primarily out of design discussions in the W3C Community Group [SVG glyphs for OpenType](http://www.w3.org/community/svgopentype/), and out of an experimental implementation by Mozilla Corporation. It is formatted as a proposal to the Open Font Format and OpenType specifications.

Please send comments about this document to public-svgopentype@w3.org ([archived](http://lists.w3.org/Archives/Public/public-svgopentype/)).

**Use cases and requirements**

The primary use case for richly expressive glyphs is creative titling, drop-caps, and other display usage in digital publishing (e.g. HTML pages and e-magazines). Other use cases are e-learning materials, online advertisements, and Unicode emoticons for social media and text messaging.

A purely graphics-based solution for the above situations, e.g. a set of images that are to be manually arranged by the document author, is not acceptable since it would not allow for full semantic accessibility (e.g. searching, indexing), ease of user input, international text shaping, or typographic features.

The proposed glyph description technology is SVG, a part of HTML 5. Note that while the SVG 1.1 <font> element does allow for color and animation, its layout model is simple and lacks OpenType’s sophisticated international and typographic feature layout facilities. Nothing in the following proposal requires use of the SVG <font> element.

**Draft proposal**

*{ At OT specification* [*OT Tables*](http://www.microsoft.com/typography/otspec/otff.htm#otttables)*, insert the following section just before the “Tables Related to Bitmap Glyphs” section. The OFF specification may insert it at the end of sec. 5 to avoid renumbering existing sections. }*

**Tables Related to SVG Outlines**

|  |  |
| --- | --- |
| *Tag* | *Name* |
| SVG | SVG glyph descriptions |

TrueType or CFF OpenType fonts may contain an optional ‘SVG ’ table, which allows some or all glyphs in the font to be defined with color and/or animation. It is not a requirement that an OT engine support this table.

*{ Link the SVG tag above to a new table page, with the following contents: }*

**The SVG table**

‘SVG ’ – SVG glyph descriptions table

This table contains [SVG](http://www.w3.org/TR/SVG) descriptions for some or all of the glyphs in the font. For every SVG glyph description, there must also exist a corresponding CFF or TT glyph description in the font.

**Main header**

|  |  |  |
| --- | --- | --- |
| *Type* | *Name* | *Description* |
| USHORT | version | Table version (starting at 0). Set to 0. |
|  |  |  |
|  |  |  |
| ULONG | offsetToSVGDocIndex | Offset (relative to the start of the SVG table) to the SVG Documents Index. Must be non-zero. |
| ULONG | offsetToColorPalettes | Offset (relative to the start of the SVG table) to the Color Palettes subtable. May be 0. |
|  |  |  |

**SVG Documents Index**

The SVG Document Index is a set of SVG documents, each of which defines one or more glyph descriptions.

|  |  |  |
| --- | --- | --- |
| USHORT | numEntries | Number of SVG Document Index Entries. Must be non-zero. |
| SVG Document Index Entry | entries[numEntries] | Array of SVG Document Index Entries. |

*SVG Document Index Entry*

Each SVG document index entry specifies a range [startGlyphID, endGlyphID], inclusive, of glyph IDs and the location of its associated SVG document in the SVG table.

|  |  |  |
| --- | --- | --- |
| USHORT | startGlyphID | The first glyph ID in the range described by this index entry. |
| USHORT | endGlyphID | The last glyph ID in the range described by this index entry. Must be >= startGlyphID. |
|  | svgD | Document Index n  |
|  | svg |  |

Index entries must be arranged in order of increasing startGlyphID. The startGlyphID of an index entry must be greater than the endGlyphID of the previous index entry, if any.

For further details about the content of the SVG documents, see “Glyph identifiers” and the following sections below.

**Color Palettes subtable**

This optional subtable allows for the SVG glyph descriptions to vary their colors based on parameter values, a sort of “color by numbers” effect. It also specifies any predefined sets of color values, or color palettes, for these parameters.

*Color Palettes subtable header*

|  |  |  |
| --- | --- | --- |
| *Type* | *Name* | *Description* |
| USHORT | numColorParams | Number of color parameters for this font. Must be non-zero. See below. |
| USHORT | colorParamUINameIDs[numColorParams] | Array of name table IDs that describe user interface strings associated with each color parameter, e.g. “Outline”, “Fill”. This may be used by document authoring applications. Use 0xFFFF if no name ID is provided. The name IDs will usually be in the font-specific name ID range, but this is not a requirement. |
| USHORT | numColorPalettes | Number of predefined color palettes |
| Color Palette | colorPalettes[numColorPalettes] | Array of color palettes. |

If a Color Palettes subtable is present, then the SVG glyph descriptions contain variables whose values are specified in one of the various color palettes. The first color palette shall be the default one. It is strongly recommended that the default values for the color variables in the SVG documents be set to the same values as in the first color palette table, for implementations that may not support color palettes. The variable names in SVG must be of the form “color<num>” where <num> is the parameter index in the range [0, numColorParmaters–1], inclusive, expressed as a non–zero-padded decimal number. See the “Glyph rendering” section below for how the values are to be passed to the SVG renderer.

*Color Palette record*

|  |  |  |
| --- | --- | --- |
| *Type* | *Name* | *Description* |
| USHORT | uiNameID | name table ID that describes user interface strings associated with this color palette. This may be used by document authoring applications. Use 0xFFFF if no name ID is provided. The name ID will usually be in the font-specific name ID range, but this is not a requirement. |
| Color | paletteColors[numColorParameters] | Array of Color records. |

*Color record*

This specifies red, green, and blue values in the sRGB color space, and an alpha value.

|  |  |
| --- | --- |
| *Type* | *Name* |
| BYTE | red |
| BYTE | green |
| BYTE | blue |
| BYTE | alpha |

**Glyph identifiers**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

For each glyph ID in an SVG Document Index Entry’s [startGlyphID, endGlyphID] range, inclusive, the associated SVG document must contain an element with id “glyph<glyphID>”, where <glyphID> is the glyph ID expressed as a non–zero-padded decimal value. This element functions as the SVG glyph description for the glyph ID.

For example, say a font with maxp.numGlyphs=100 has SVG glyph definitions only for its last 5 glyphs. The last SVG glyph definition has its own SVG document, but the rest share an SVG document (say, to take advantage of shared graphical elements). There will be two index entries, the first with glyph ID range [95, 98] and the second with glyph ID range [99, 99]. The SVG document referenced by the first index entry will contain elements with id “glyph95”, “glyph96”, “glyph97”, and “glyph98”. The SVG document referenced by the second index entry will contain an element with id “glyph99”.

**Glyph semantics and metrics**

The glyph descriptions in the SVG documents are considered to be the SVG versions of the glyphs with the corresponding IDs in the CFF or glyf table. They are designed on an em specified in the head table’s unitsPerEm field, as with CFF and TrueType glyphs. SVG glyph definitions will be in SVG’s y-down coordinate system, with the default baseline at y=0. For example, the top of a capital letter may be at y=-800, and the bottom at y=0. This coordinate system will need to be translated appropriately to the coordinate system of the rest of the OT tables and the coordinate system of the graphics environment.

Glyph semantics are expressed in the usual OT way (cmap table followed by GSUB). Glyph metrics such as horizontal and vertical advances are specified in the usual OT tables (hmtx and vmtx), and glyph positioning adjustments by the GPOS or kern table.

As with CFF glyphs, no explicit glyph bounding boxes are recorded. The “ink” bounding box of the rendered SVG glyph should be used if a bounding box is desired; this box may be different for animated vs static renderings of the glyph.

The font bounding box in the head table must be suitable for static renderings of the glyphs. No explicit font bounding box for dynamic renderings of the glyphs is recorded.

Note that the glyph advances are static and not able to be made variable or animated.

**Glyph rendering**

The SVG glyph descriptions may be rendered statically or with animation enabled. Note that static rendering is done by not running any animations in the SVG document; this is different from running the document with animations running but at a snapshot time of zero seconds. Some clients may choose not to support – or may not be able to support – animation. Clients that support animation may still request, in certain cases, that the glyph be rendered statically, e.g. for printing to paper.

The following user agent style sheet must be applied to SVG documents processed from the SVG table:

:root {

 fill: context-fill;

 fill-opacity: context-fill-opacity;

 stroke: context-stroke;

 stroke-opacity: context-stroke-opacity;

 stroke-width: context-value;

 stroke-dasharray: context-value;

 stroke-dashoffset: context-value;

}

In addition, if a Color Palettes subtable is present and supported by the font engine, the user agent style sheet must include CSS Custom Property declar teations for the color variables. This is done by including ‘numColorParams’ declarations in the :root rule of the form:

 var-color<num>: <color>;

where <num> is each of the values from zero to numColorParameters–1, inclusive, expressed as a non-zero-padded decimal number; and <color> is the <num> index within the desired Color Palette, expressed in SVG’s [<color>](https://svgwg.org/svg2-draft/color.html) format. An example entry in the style sheet is:

 var-color0: rgba(255,0,0, 0.5);

and the corresponding usage in an SVG glyph description could be something like:

 <path fill=var(color0) d="..."/>

Note that SVG’s context-fill value may be used in the glyph descriptions to denote the current text color.

*Security*

It is required that all rendering of SVG glyphs be done in the “secure animated mode” or “secure static mode” specified in the W3C document [SVG Integration](https://svgwg.org/specs/integration/). These modes permit no script execution, external references, interactivity, or link traversal.

**Text layout process**

An implementation that supports the SVG table first does layout in the usual OT manner, using the cmap, GSUB, hmtx, and other OT layout tables. This results in a list of glyph IDs arranged at particular x,y positions on the surface (along with the appropriate scale/rotation matrices). At this point, for each such glyph ID, if an SVG glyph description is available for it, it is rendered (in static or animated mode, as appropriate and if supported by the engine); otherwise, the CFF or TT glyph description must be rendered. Since the glyph advances are the same in either case, and not allowed to be animated, switching between SVG and CFF/TT rendering, or between animated and static SVG, should not require re-layout of lines (unless line layout depends on ink bounding boxes).