SVG in OpenType
Bringing Color and Animation to Fonts

Working Draft · August 7, 2012
Sairus Patel · Adobe Systems

Background

This document proposes a way for OpenType fonts to optionally include colored and/or animated glyphs.

The primary use case 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 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.

Nothing in the following proposal requires an extension to the SVG specification.

Custom fill or a "color-by-numbers" facility is a compelling aspect of SVG-in-OT. While not explicitly mentioned in the draft proposal, it is being discussed in the W3C Community Group SVG glyphs for OpenType (all are welcome to participate). A future version of this working draft (and perhaps the SVG specification itself) may be extended to allow for custom fill.

Draft Proposal

{ At OT specification OT Tables, 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 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.

SVG glyph definitions table format

Type	*Name*	*Description*
USHORT	majorVersion	Major version (starting at 1). Set to 1.
USHORT	minorVersion	Minor version (starting at 0). Set to 0.
ULONG	svgGlyphClassDef	Offset from the beginning of the SVG table to a Class Definition Table that specifies which glyphs are defined in the SVG document, and, of these, which involve animation. See "Glyph identifiers" below. Must be non-zero.
ULONG	compressedDocOffset	Offset from the beginning of the SVG table to a compressed SVG document. See "The SVG document" below. Must be non-zero.
ULONG	compressedDocLength	Length of the compressed SVG document. Must be non-zero.
ULONG	uncompressedDocLength	Length of the uncompressed SVG document. This is provided as an aid to securely and efficiently decompressing the SVG document. Must be non-zero.

The SVG document

The SVG glyph descriptions are contained in a single, zlib-compressed SVG document specified by the compressedDocOffset and compressedDocLength fields. The compress2() function of zlib (or an equivalent, compatible algorithm) must have been used for compression. The uncompress() function of zlib (or an equivalent, compatible algorithm) is to be used for decompression.

Glyph identifiers

The svgGlyphClassDef field provides an offset to a Class Definition table that partitions the set of glyphs in the font as follows:

svgGlyphClassDef enumeration list

Class	*Description*
0	The SVG document does not contain a description for this glyph
1	The SVG document contains a description for this glyph that does not involve animation.
2	The SVG document contains a description for this glyph that involves animation.

For each glyph ID with class value 1 or 2, the 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. The svgGlyphClassDef is essentially a cache and allows an implementation to delay or avoid decompressing and parsing the SVG document. An implementation may also choose to ignore the svgClassDef and interface directly with the SVG document to determine if an element with a "glyph<glyphID>" pattern is present and whether that element involves animation.

For example, say a font with maxp.numGlyphs=100 has SVG glyph definitions only for its last 4 glyphs, and all of these are animated. The ClassDef will assign glyph IDs 96–99 to class value 2; the rest of the glyph IDs will be assigned to class value 0 (implicitly, for maximal compactness of the ClassDef). The SVG document will contain elements with id “glyph96”, “glyph97”, “glyph98”, and “glyph99”. It must not contain elements with id “glyph0”, “glyph1” and so on up to “glyph95”.

Glyph semantics and metrics

The glyph descriptions in the SVG document 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. 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.

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 Working Draft SVG 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).