

W3C SVG WG Liaison to MPEG SC29/WG11

Dear SC29/WG11,

We, the W3C SVG WG, have received your liaison and the FDIS specification for Lightweight Application Scene Representation, or LAsER, and thank you for keeping us updated with the status of your work. We reviewed the specification during our latest face-to-face meeting held in Sydney, January 23-27, 2006, and this liaison is the result of this process.

While the Group generally appreciates the work that has been conducted on LAsER, which in many respects is of high quality, it appears that some technical issues around SVG harmonization remain preventing LAsER to reach some of its requirements related to SVG compatibility and the capacity to successfully reconstruct an SVG scene from a LAsER stream. A non-exhaustive list of issues follows.

1. High-Level Comments

It is the W3C SVG WG's belief that, ideally, the LAsER specification should not require specific language around SVG harmonization, but rather define a generic mechanism to provide a binary-encoded, streaming-optimized efficient transmission layer on top of SVG subsets as defined by the W3C in coordination with organizations such as OMA and 3GPP.

It is our belief that with such a compartmentalized architecture SVG interoperability will be preserved whatever the transmission mode, text-based XML, gzip, LAsER or others. Additional profiling on top of SVG Tiny 1.1 or SVG Tiny 1.2 harms SVG and its ecosystem, of which LAsER is a part.

2. Extension Conformance Requirements

Based on some of the issues that arose during our review of the LAsER FDIS specification, we have added new extension conformance requirements to the SVG Tiny 1.2 specification in order to clarify or further specify requirements that were previously not apparent enough. These new requirements are reproduced here:

1. An extension **MUST** support the normative chapter of the SVG specification that defines conformance to SVG.
2. An extension **MUST** support the normative chapter of the SVG specification that details extensibility.
3. An extension **MUST** support the normative chapter of the SVG

- specification that defines conditional processing.
4. If using features defined in SVG Full, an extension **MUST NOT** redefine the syntax of the syntax of those features.
 5. An extension **MUST NOT** redefine the semantics of any existing SVG element or attribute.
 6. SVG attribute and element names must not be reused in extensions, even in a separate namespace, in order to avoid author confusion.

3. Encoding Conformance Requirements

The SVG Tiny 1.2 specification has also been enhanced to specify a new conformance requirement for encoding mechanisms:

An encoding of SVG is compatible with SVG if and only if once a conforming SVG document has been encoded using that encoding, the specification that defines the encoding **MUST** guarantee that a user agent conforming to it processes the SVG document in a manner that is identical to the processing of the original SVG document by a conforming SVG user agent."

4. SVG Fonts

Section 6.2.2 about SVG Fonts in LAsER notes that "SVG elements related to font description are not supported by LAsER". While we understand that LAsER creates an OpenType font stream from the SVG description of an SVG font, there are cases in which the correct reconstruction of the SVG scene requires information found in the SVG font markup that cannot be accurately translated to OpenType.

4.1. Introduction

Glyphs in SVG fonts use the same syntax for their path data as the general SVG path command. In SVG Tiny, glyphs are limited to a single path and a single color. It may therefore seem that they could be readily converted to OpenType (compared to SVG Full, where this is clearly not the case). However, the conversion is still not straightforward.

4.2. Curve types

Type 1 fonts (and OpenType fonts with CFF data) use cubic splines. TrueType (and OpenType fonts with TrueType data) use quadratic splines. Thus, directly converting font data to SVG paths (and thus to SVG fonts) is straightforward. The reverse is not the case. An SVG font may contain glyphs that have quadratic data and others that have cubic; or each glyph may have a mixture of both types of data in the one path.

OpenType fonts should not mix the two types of glyphs, and must not mix both curve types in a single glyph [0]. Conversion is therefore required, which entails approximation; quadratic curves cannot represent all possible cubic curves [1], [2].

4.3. Point placement

In OpenType, points must be placed at the mathematical extrema of each shape (and not at the optical extrema, or at other places that might be convenient). [3]

SVG imposes no such restriction. Once again, this means that conversion from OpenType to SVG is straightforward, but from SVG to OpenType requires curve refactoring and thus approximation.

4.4. Winding rules

OpenType has more restrictive constraints of winding rules for shapes with holes; SVG has more flexibility. Thus OpenType to SVG is straightforward, but SVG to OpenType may require path reversal [3].

4.5. Overlapping shapes

TrueType and OpenType and Type 1 glyphs are constrained on overlapping shapes. Glyph outlines go around the union of all the shapes that seem to visually make up the design. SVG glyphs can overlap, especially when used to make 'stroke fonts'. Conversion from SVG to OpenType requires a mathematical union operation on the path segments.

4.6. Language-sensitive glyphs

SVG allows a list of one or more languages to be placed in an attribute on each glyph. This allows glyphs for harmonized characters, such as the han 'bone' character, to have different glyphs for different languages. It might be possible to represent this information using OpenType tables, but this is unlikely to successfully round-trip.

5. Cursor and Virtual Pointer

Section 6.8.10 of the LAsER FDIS specification explains how, in LAsER, key events can trigger scene updates that then can update the location of a piece of SVG graphics referred to by the <cursor> element's xlink:href attribute. As the referenced graphics location gets updated, mouse events, such as mouseover, mouseout, mousemove and click, are triggered, the whole chain

of event processing in effect catering for the functionality of a “virtual pointer”.

The W3C SVG WG wishes to indicate that the use of the <cursor> element as described above entails a redefinition of the semantics of the <cursor> element as defined by the SVG 1.1 recommendation. Indeed, the SVG <cursor> element allows content authors to point to a raster graphics file to substitute the default pointer graphics offered by the platform. Currently, there is no functionality related to “virtual pointer” defined in the SVG 1.1 recommendation, or any other work-in-progress from the W3C SVG WG, neither is there any normative text that defines how the location of a given graphics group triggers mouse events on the SVG tree.

Furthermore, the SVG <cursor> element is not available in either of the SVG Tiny 1.1 and SVG Tiny 1.2 subsets, inherently introducing a discrepancy in feature set between the SVG features offered in LAsER and those offered by any of the W3C-approved mobile SVG profiles.

6. Multimedia Elements

Sections 6.8.8 and 6.8.36 of the LAsER FDIS specification define that LAsER supports the <audio> and <video> elements from the W3C SMIL 2.0 recommendation. While SVG Tiny 1.2 has a strong SMIL foundation, including SMIL’s timing and animation model, the <audio> and <video> elements defined in the SVG Tiny 1.2 specification are not the same as those described by SMIL 2.0. The reason for this adaptation is that SMIL delegates positioning to a layout system that is redundant with existing SVG, and more generic, features (in the case of the <video> element), and a resource referencing scheme based on the XLink specification.

While we applaud the intent of leveraging the <audio> and <video> elements as defined by the SMIL 2.0 recommendation for LAsER, we must note that supporting these elements as opposed to those defined in SVG Tiny 1.2 results in an incompatibility between LAsER and SVG in that a valid SVG Tiny 1.2 file using the <audio> and <video> elements will not be appropriately encoded in LAsER. For instance, with the current specification of LAsER, the x, y, width, height and xlink:href attributes of the SVG Tiny 1.2 <video> element shall not be understood by a LAsER encoder, thus resulting in content that will not be able to locate the video resource or be able to render the resource at the right size or location.

The SVG WG is aware that LAsER can only reference specification which have reached the status of a Recommendation. Therefore, we suggest that as soon as the SVGT1.2 becomes a REC, the LAsER group replaces the SMIL2 references by SVGT1.2 references by corrigendum

The <video> element referred to in LAsER has an overlay attribute with possible values being those of the SVG overlay attribute as well as an additional fullscreen value. The first issue here is that the SMIL 2.0 <video> element that LAsER references does not allow for the SVG overlay attribute to be used. Were the LAsER specification modified to reference the SVG Tiny 1.2 <video> element, the additional value fullscreen would be problematic as it would encourage content authors to deploy content that claims to be SVG but yet would not run in a consistent manner across LAsER and pure SVG clients.

7. Clipping

Section 6.8.15 of the LAsER FDIS specification defines extra size and choice attributes for the SVG <g> element that LAsER references. Using a combination of the two attributes' values, content authors are able to use the SVG <g> element as a clipping container. While clipping features are not available in either SVG Tiny 1.1 or SVG Tiny 1.2, the SVG 1.1 recommendation defines the <clipPath> element and the clip-path property that, used together, define regions subject to clipping. Neither of these two SVG language features are leveraged by the LAsER specification, thus failing to fully harmonize with the SVG language for a feature that could have been expressed using SVG language, potentially with restrictions such as alignment to device pixels or the type of shapes that can be used to define a clip path. The most appropriate way for this feature to be expressed using existing SVG features is currently being investigated by the SVG Working Group.

8. Display Switch

Section 6.8.15 of the LAsER FDIS specification defines extra size and choice attributes for the SVG <g> element that LAsER references. Using a combination of the two attributes' values, content authors are able to choose which, if any, of the child elements of a <g> element is to be displayed on screen, turning off the rendering of all the other siblings. The W3C SVG WG feels that extending the SVG <g> element to achieve such a feature redefines the semantics of the element, which is a serious concern to the Group. Were LAsER to retain this feature, the Group would recommend LAsER to define a separate container element of which role would specifically be to act as a display switch.

9. Layout Container

Section 6.8.15 of the LAsER FDIS specification defines extra size and choice attributes for the SVG <g> element that LAsER references. Using a

combination of the two attributes' values, content authors are able to use the SVG <g> element as an automatic layout container. Currently and intentionally, none of the SVG specifications offer layout features. As such, the W3C SVG WG feels that extending the SVG <g> element to achieve such a feature redefines the semantics of the element, which is a serious concern to the Group. Were LAsER to retain this feature, the Group would recommend LAsER to define a separate container element of which role would specifically be to act as a vertical or horizontal layout container.

10. Font Style

Section 6.8.32 of the LAsER FDIS specification defines additional values for the SVG font-style property such that content authors can use this property to specify information that is usually defined via the use of two other properties, font-weight and text-decoration, in the SVG language. This extension to the SVG language is conducted in a way that valid SVG content encoded in LAsER would not be reproduced accurately on the client since the information provided by the font-weight and text-decoration properties would not be understood by the LAsER encoder. The W3C SVG WG feels that adding new values to an existing SVG property will promote the spread of contents incompatible with SVG.

11. Timed Script Execution

Sections 6.8.19.2 and 6.8.27.2 of the LAsER FDIS specification define several extra attributes (timeAttribute, delay and begin) allowing to control the time at which commands are executed for either the <listener> or the <script> elements. The features defined in such elements are already expressible using existing SVG language features, where commands would be executed when the beginEvent event is triggered by a <set> element which would be used to accurately control the execution time with the full extent of timing features expressible via its begin attribute. The W3C SVG WG believes that such duplication of features via extensions to the SVG language is not necessary and promotes the usage of LAsER-specific features whereas content authors could use strictly SVG language features.

12. PNG and JPEG Image Formats

All of the SVG specifications and profiles have a conformance requirement that mandates support for both the PNG and JPEG image formats in conforming user agents, allowing content authors to rely on both formats for content delivery. The LAsER FDIS specification fails to mandate support for image formats, thus introducing an incompatibility with SVG where SVG contents using raster images in either of the PNG or JPEG formats may not

be reproduced accurately by a LAsER client.

13. LAsER Mini

Section 8.2.2.5 of the LAsER FDIS specification restricts the list of possible values for the fill attribute on the family of animation elements to the single freeze value for the LAsER Mini profile. The default value for this fill attribute in all SVG specifications is remove, an unsupported value in LAsER Mini. As such, LAsER Mini introduces an incompatibility in that an animation element failing to explicitly specify the value of its fill attribute, thus relying on its default value, will have different behaviors in a LAsER Mini user agent, a vanilla SVG user agent or even a LAsER Full user agent where the default value for the fill attribute is unaltered.

Section 8.2.2.1 of the LAsER FDIS specification restricts the list of possible values for the transformBehavior and overlay attributes for the <video> element. The restrictions for both attributes remove the default value for each attribute as defined in the SVG Tiny 1.2 specification, which creates the same compatibility issue described above for the fill attribute of animation elements.

The SVG WG suggests 2 ways forward :

1. LASER-Mini uses the same default values as the SVGT spec. (for instance, the "fill" attribute would have 2 possible values : 'freeze' and 'remove', with 'remove' as the default value)
2. LASER-Mini spec. clearly states that the content without the attributes will be rejected (at the encoding step) as non-LAsER mini-compliant

14. Conclusion and Proposed Plan of Action

All of the issues listed so far are of deep concern to the W3C SVG Working Group, especially given how far along the ISO standardization process the LAsER specification is. In the spirit of seeing LAsER fit in the SVG ecosystem without disrupting interoperability, we would like to urge the LAsER Group to delay the full standardization of the LAsER specification so that all harmonization issues listed in this liaison are addressed. In order to address these issues, we offer the LAsER Group to join us in any number of joint teleconferences required to achieve this goal, as well as email exchanges.

As a general question, the W3C SVG Working Group would like to understand the planned process to make LAsER v1 and v2 compatible with SVG Tiny 1.2. Our suggestions may require the LAsER specification to deprecate or add elements. We would like to understand how this is applied

to the current specification, and how implementors of the specification can create conforming implementation.

As a concluding note, the W3C SVG WG would like to invite the LAsER Group to align completely with the two mobile profiles defined by the W3C, namely SVG Tiny 1.1 and SVG Tiny 1.2, in order to continue the spread of a consistent language feature set across the whole SVG ecosystem.

References

- [0] <http://www.microsoft.com/typography/otspec/recom.htm>
- [1] <http://www.true-type-typography.com/ttandt1.htm>
- [2] <http://developer.apple.com/fonts/TTRefMan/RM08/appendixE.html>
- [3] <http://developer.apple.com/fonts/TTQualSpec/QS02/FQS2.html>