Jean-Claude Dufourd

LASeR: The Lightweight Rich Media Representation Standard

Rich media stands for any dynamic combination of video, audio, images, graphics, text; and featuring interactivity, animation, and dynamic behavior over time according to the user's requests. Using rich media allows for better user interfaces, replacing unwieldy catalogs by fluid services and eye-pleasing presentations. The term rich media is different from "multimedia," the latter being sometimes identified with "audiovisual." The Lightweight Application ScenE Representation (LASeR) is the rich media representation standard for mobile devices based on reusing simple MPEG-4 Systems tools and the MPEG-4 decoder model. LASeR addresses rich media services on a wide range of devices and delivery situations.

INTRODUCTION

MOTIVATION

Five years ago a quest for an open rich media representation standard for mobile devices started, with an initial focus on existing standards. There existed a powerful part of the Moving Pictures Expert Group (MPEG) MPEG-4 standard, the so-called Binary Format for Scenes (BIFS). Although it included most of the rich media features, its mobile implementations were not sufficiently efficient. There existed also Flash, a dominant proprietary standard on PCs, with little traction on mobile devices. There existed a promising open standard, Scalable Vector Graphics Tiny 1.1 (SVGT1.1). However, it was not a rich media standard since SVGT1.1 had no audio and video support. Although its successor SVGT1.2 contained audiovisual support, the standard was too complex for most mobiles, was not dynamic, and had a very slow development process.

LASeR filled a gap that none of the existing standards at the time were able to and became the rich media standard for small mobile devices, composing and synchronizing media, graphics, and text dynamically and interactively. LASer allows the representation of scenes, where a scene is presented to a viewer visually and aurally, possibly with interaction and changes over time. If there are changes over time, the starting state is referred to as the initial scene, and the changes to the scene are referred to as scene updates.

OBJECTIVES

The main LASeR requirements were the following:

- support an efficient, compressed, streamable, extensible, and easy-to-decode representation of rich media scene data, compatible with SVG Tiny to leverage existing market trends and content development tools
- allow the definition of profiles leading to compact implementations, compatible with handheld devices with limited resources; an example consists of a profile for Java 2 Mobile Edition (J2ME) implementations
- allow the representation of differential scenes, i.e., scenes meant to build on top of another scene
- allow for compact and fast implementations, with low runtime memory requirements and suitability for (at least in part) hardware implementation.

ISSUING BODY AND SCHEDULE

LASeR was developed within the Systems subgroup of MPEG starting mid-2004. A first version including SVG Tiny 1.1 features with audio, video, gradients and transparency, scene updates, and a binary compression format was published in June 2006. A second version with broadcast optimizations, automated scrolling, and interfaces to device features was completed in mid-2007. A third version including SVG Tiny 1.2 compatibility was completed in the end of 2007. A fourth version with new requirements for broadcast and Internet Protocol TeleVision (IPTV) services has started to be defined in early 2008.

TARGET APPLICATIONS

The LASeR target applications include: rich media portal, interactive mobile TV, interactive screen saver, podcasting, home convergence nexus, and device user interface.

- **Rich media portal**—A portal is the starting point for all customers of a service provider, such as a mobile operator. A rich media portal uses rich media to provide better content presentation and easier interactivity. The rich media portal application shows how a LASeR engine can enhance an existing service with rich media and is illustrated in Figure 1(a).
- **Interactive mobile TV**—This application provides access to TV channels and related information. It is the aggregation of multiple different rich media use cases including interactive mosaic, electronic program guide, voting, and personalized newscast. An example is illustrated in Figure 1(b) and shows a small TV screen, additional interactive information, a channel subscription list and a ticker.
- **Interactive screen saver**—The application in Figure 1(c) showing weather information is one instance of a larger class of interactive screen saver applications that get content

Digital Object Identifier 10.1109/MSP.2008.929813

1053-5888/08/$25.00©2008IEEE

[FIG1] LASeR-Based rich media services: (a) rich media portal, (b) interactive mobile TV, (c) interactive screen saver, (d) podcast, (e) home convergence nexus, and (f) device user interface.

updates in background (e.g., fix/mobile convergent services).

- **Podcast**—This now well-known application combines on-demand media with syndication feeds. It can be used in a variety of cases and its radio or video instances are popular. The example in Figure 1(d) shows a promotional podcast with commercial links.
- **Home convergence nexus**—These applications illustrated in Figure 1(e) stem from a converging trend in electronics services, phone, Internet, TV (broadcast or IPTV), media on demand, wireless, and wired. These tend to converge in the home and require a layer of integration.
- **Device user interface**—This application uses the services offered by the LASeR engine present on the device to modify and extend the user interface of the device seamlessly, providing continuity with mobile services. The device resources and the connected resources are blended into a coherent interface. An example is shown in Figure 1(f) with a "phone top" with a weather popup.

STRUCTURE OF THE STANDARD

The MPEG-4 LASeR standard is packaged as document ISO/IEC 14496-20 and specifies a format for rich media scene description called LASeR and a companion simple packaging standard called Simple Aggregation Format (SAF). LASeR applications can benefit from SAF packaging in many scenarios, but they do not need to be used together.

TECHNOLOGY

FUNCTIONALITIES

A LASeR scene describes four aspects of a presentation: the spatial and temporal organization of scene elements (media or graphics), the interaction with these elements, and the changes in the scene.

Spatial organization may refer to the spatial layout of the visual elements. Temporal organization may refer to the synchronization of the scene elements. The interaction with the scene elements may refer, for instance, to a case when the user clicks on an image. The scene changes may refer to other modifications in any of the components of the scene.

A LASeR scene is described as a tree of elements, some of which describe how to layout, paint, animate, modify, synchronize or interact with other elements. Other elements describe what the viewer sees or hears (e.g., images, video, graphical shapes, text, audio).

A LASeR scene may change by means of animations or scene updates. Deterministic or parametric animations such as bouncing or blinking an element repeatedly can be described by elements that animate the properties of other elements, such as their position and visibility. Scene updates are either timed modifications of the scene tree that are sent by a server or modifications triggered by user interaction. The LASeR initial scene and scene updates can be compressed and streamed over various delivery mechanisms.

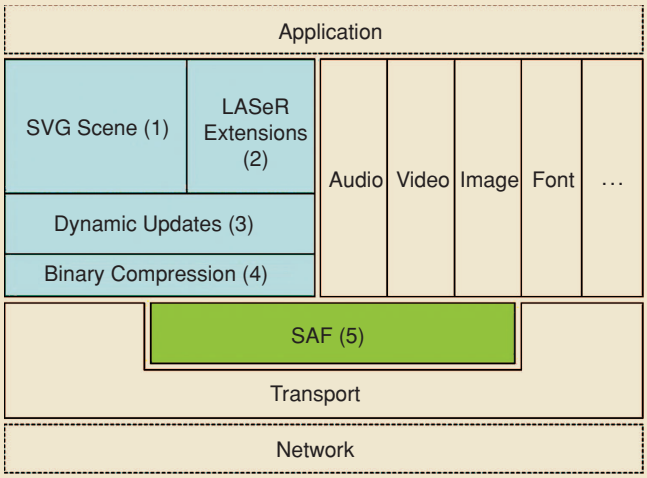
ARCHITECTURE

The architecture of a LASeR system is illustrated in Figure 2. An application first shows an initial SVG scene [labeled (1) in the figure] with LASeR scene extensions [labeled (2)] to a user. The application then changes what is shown to the user via dynamic updates [labeled (3)] to the scene tree. Next, a binary encoding for compression [labeled (4)] is used to improve transmission delays. Finally, the LASeR engine on top of which the application is built uses the services of a transport layer, possibly complemented by SAF [labeled (5)] to provide missing packaging and synchronization features.

TOOLS

The LASeR standard allows SVG scene tree representation, LASeR scene tree extensions, dynamic updates, binary encoding, audiovisual support, usage of font information, and services as incremental scenes.

- **SVG Scene Tree Representation**—MPEG organization reused relevant SVG elements as a basis for the development of LASeR scene tree representation. These consist of
  - — structuring elements, such as the root object <svg>, the group <g>, the dictionary <defs>, the hyperlink <a>, the selector <switch>, and finally the reference object <use>
  - — media elements inherited from Synchronized Multimedia Integration Language (SMIL) such as <audio>, <image>, <video> and <animation> (a reference to a complete scene)
  - — animation elements, providing parametric animations, such as <set>, <animate>, <animateColor>, <animateTransform> and <animateMotion>
  - — geometric shapes, such as <path> defining complex curves, <rectangle>, <circle>, <ellipse>, <line>, <polyline>, and <polygon>
  - — painting methods, such as <solidColor>, <linearGradient> and <radialGradient>
  - — metadata or documentation elements, such as <title>, <desc> and <metadata>.
- **LASeR Scene Tree Extensions**—MPEG has also identified areas where extensions were needed to allow the development of efficient services with LASeR:
  - — simple pixel-aligned rectangular clipping
  - — restricted, nonresampling rotation and a full screen mode for videos
  - — multiple synchronization references
  - — new events: longAccessKey, repeatKey, shortAccessKey, screen orientation events, pause and resume, the last two used to pause and resume video, audio and other timed elements
  - — automated generic scrolling
  - — incremental scenes, designed as sets of independent scene segments: a scene segment is either an initial scene, or a set of timed updates to be applied to another scene
  - — a mechanism of local ID and global ID for elements and streams: local IDs are specific to a scene segment, and cannot clash; global IDs are used on elements and streams that need to be used across scene segments.
- **Dynamic Updates**—Dynamic updates allow the application to change what is presented to the user and are implemented using the LASeR commands, which include
  - — newscene: installs a new scene in the LASeR engine; any LASeR application begins with this command
  - — insert: inserts new elements in a scene



[FIG2] **Modular architecture of a LASeR system.**

— replace: modifies properties of elements, or replaces an element by another element
— delete: removes an element from the scene tree
— commands to increment properties, to send events; an interface to persistent storage, a tune-in command and authoring optimizations.

■ **Binary Encoding**—The LASeR binary format allows the encoding of the scene tree and the updates. It uses a compact representation for the structure of the elements and coding algorithms for the attribute values. Because the mobile platforms usually lack hardware float processing, the compression of these attribute values is simpler than that using BIFS. Special care was taken for the encoding of values for some attribute types, for instance, series of coordinates are encoded differentially; a different encoding is used for transforms and matrices, taking into account their different statistics. The LASeR binary syntax is extensible so that private extensions can be mixed with regular LASeR elements and attributes, to be ignored by unextended decoders.

■ **Audiovisual Support**—The LASeR specification supports not only the audio and visual objects within LASeR scenes but also the composition and control of such objects within the scenes. The LASeR specification includes SMIL2/SVGT1.2 audio and video elements as well, and the additional SMIL2 MediaClipping module for VCR-like media control. Audio and video streams are carried outside the LASeR stream and referred to using binary identifiers.

■ **Usage of Font Information**—Both LASeR and SVG allow content creators to embed font information within the scene. Because the SVG fonts solution was deemed to be too limited by MPEG, the carriage of the font information along with the scene information as a media stream was recommended. The exact format to be used is optional, a possible option being the usage of MPEG-4 Part 18, which provides the definition of a font data stream, able to carry OpenType fonts (possibly compressed).

■ **Services as Incremental Scenes**—Many LASeR incremental scenes are made possible by scene segments, which are LASeR streams, each containing an addition to an existing scene. From a server-side point of view, the interactive services can be considered as series of separate connections, as opposed to the continuous connection of the streamed services. Interactive services are typically implemented using separate HTTP connections, since each data burst results from a user request. However, from a LASeR viewer point of view, the same scene/service is modified, so each server response is a scene segment rather than a new scene.

PROFILES AND LEVELS

Throughout the process of defining LASeR, two objectives have always been difficult to accommodate: efficiency and SVG compatibility. Some concerns have been raised by implementers in the Third Generation Partnership Project (3GPP), Open Mobile Alliance (OMA), and MPEG about making the implementations small and efficient. These concerns were justified by the fact that SVG, Cascading Style Sheet (CSS), and Synchronized Multimedia Integration Language (SMIL) were designed for the PC platform, despite the existence of the SVG Tiny profile targeted at mobile devices.

In response to such concerns, MPEG created a profile of the LASeR specification referred to as LASeR Core. To allow efficient implementation and high rendering performance, LASeR Core restricts the specification to allow no bit map resampling, no dash capabilities, no gradient animation, no inheritance, and a simple mode for animation. These ensure a small memory footprint, a much faster scene tree management, and reasonable rendering performance even when using video and gradients.

The LASeR Main profile targets LASeR usage on more powerful devices, such as PCs and contains most of the features of the third version of the LASeR specification.

The LASeR Mini profile is a subset of LASeR Core containing only elements from the first version of the LASeR specification. LASeR Mini targeted J2ME implementations but is now superseded by LASeR Core. The LASeR Full profile contains all elements from the first version of the LASeR specification. LASeR Full targeted more powerful devices, such as PCs, but is now superseded by LASeR Main. Both LASeR Mini and LASeR Full maintain backward compatibility.

COMPARISON WITH OTHER STANDARDS

When comparing LASeR with other standards, SVG, DIMS, SMIL, and BIFS are likely candidates for comparison.

The LASeR standard (with Amendments 1 and 2) is a superset of SVGT1.2, adding scene extensions, binary encoding, and LASeR commands. LASeR is also a superset of 3GPP Dynamic Interactive Multimedia Scenes (DIMS), which has adopted SVGT1.2, some of LASeR scene extensions and most of the LASeR commands.

Similarly to SVG, LASeR does not have the complex layout and timing containers of SMIL and thus is much easier to implement. SMIL is missing dynamic updates and compression to be relevant for fluid mobile services.

BIFS addresses two-dimensional (2-D), three-dimensional (3-D), and mixed 2-D/3-D scenes and has a very large object set and a complex encoding.

In comparison with BIFS, LASeR focuses on 2-D scenes only, has a much simpler object set, and a much simpler decoding (with no floating point operation). The LASeR implementations can be used satisfactorily on devices at least twice smaller in code, memory footprint and performance than those of BIFS.

LASeR RESOURCES

Standard

- LASeR: ISO/IEC 14496-20:2006 [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/c041650_ISO_IEC_14496-20_2006%28E%29.zip
- LASeR reference software: ISO/IEC 14496-5:2001/Amd.17:2008 [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/c046597_ISO_IEC%2014496-5_2001_Amd_17_Reference_Software.zip
- LASeR test sequences: ISO/IEC 14496-4:2004/Amd.25:2008 [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/c046295_ISO_IEC_14496-4_2004_Amd_25_2008_Conformance_Testing.zip and ISO/IEC 14496-4:2004/Amd.27:2008 [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/c046297_ISO_IEC_14496-4_2004_Amd_27_2008_Conformance_Bitstreams.zip
- SVG Tiny 1.1 [Online]. Available: http://www.w3.org/TR/SVGMobile/, SVG Tiny 1.2 [Online]. Available: http://www.w3.org/TR/SVGMobile12/ SVG 1.1 [Online]. Available: http://www.w3.org/TR/SVG11/
- SMIL [Online]. Available: http://www.w3.org/TR/SMIL2/
- 3GPP DIMS [Online]. Available: http://www.3gpp.org/ftp/Specs/archive/26_series/26.142/26142-710.zip

Tutorials and White Papers

- LASeR and SAF for Dummies, MPEG Public Document w6969[Online]. Available: http://www.chiariglione.org/mpeg/technologies/mp04-lsr/laser-presentation.zip
- MPEG-4 LASeR White Paper, MPEG Public Document w7507 [Online]. Available: http://www.chiariglione.org/mpeg/technologies/mp04-lsr/index.htm

Overviews

- J.C. Dufourd, O. Avaro, and C. Concolato, "An MPEG standard for rich media services," *IEEE Multimedia*, pp, 60–66, Oct.-Dec. 2005.

Resources for Further Development

- LASeR Interest Group [Online]. Available: www.mpeg-laser.org
- Y. Lim, Y.S. Joung, W. Cheong, J. Cha, and K. Kim, "The simple aggregation format for lightweight applications scene representation (LASeR)," *IEEE Trans. Consumer Electron.,* vol. 52, issue 1, pp. 287–291, Feb. 2006.
- Y.S. Joung, J.H. Cha, W.S. Cheong, Y.K. Lim, and K.H. Kim, "An efficient type codec for point data in Lightweight Application Scene Representation (LASeR)," *ETRI J.*, vol. 27, no. 6, pp. 818–821, Dec. 2005.
- GPAC open source project, including LASeR tools and renderer [Online]. Available: http://gpac.sourceforge.net

LASeR PRODUCTS

Software products

- Streamezzo: see www.streamezzo.com
- PineOne: see www.pineone.com

Both product chains are end to end, including LASeR players on a variety of mobile devices and platforms, LASeR content servers and streamers, LASeR authoring software, and white-label applications.

PERFORMANCE

FUNCTIONAL PERFORMANCE

The main functional improvement of LASeR over SVGT1.2 consists of the dynamic updates. Implementing dynamic scenes in SVGT1.2 involves scripting (mostly with interpreted ECMA-Script), the Document Object Model (DOM) interface, a dedicated server, TCP connections and lots of XML manipulation. By contrast, LASeR Commands allow authors to create dynamic scenes with declarative text rather than programming, and are multiple orders of magnitude faster and smaller in code size.

SPEED/COMPLEXITY PERFORMANCE

The main speed/complexity performance improvement in LASeR over SVGT1.2 and DIMS is due to the streamable binary encoding. The LASeR binary encoding achieves more than a factor of two compression improvement over gzipped SVG/DIMS. The difference is even bigger on small scenes or fragments. On the client side, the author's consistent experience is that decoding a binary stream is much faster than parsing an XML document, thus adding the advantage of better client reaction time to the shorter transmission delay brought by better compression.

A key complexity advantage is also provided by the LASeR Core profile. SVGT1.2 and DIMS cannot be implemented realistically on a mobile Java (J2ME) platform: a pure Java implementation would be too big, and if it runs at all, would be too slow to be usable. By contrast, the LASeR Core profile is implementable on J2ME, and therefore gives access to a wide range of lower-end existing and future phones.

RESOURCES

Several LASeR resources are listed in "LASeR Resources."

PRODUCTS

A few examples of LASeR products are included in "LASeR Products."

AUTHOR

Jean-Claude Dufourd (jean-claude.dufourd@streamezzo.com) is chief scientist and cofounder of Streamezzo. He is co-editor of the LASeR standard. SP