Université Pierre et Marie Curie
Paris 6 – Jussieu

Manuscrit

présenté pour l'obtention d'une

Habilitation à Diriger des Recherches

Spécialité: Informatique

Interactive Multimedia Content

par

Jean-Claude Dufourd

Paris, Octobre 2010

Table des matières:

1. Résumé

J'ai travaillé sur de nombreux projets situés tout au long de la chaine de traitement du « rich media ». Certains projets étaient dans un contexte de financement européen ou national, d'autres dans des contextes de thèses : ceux-ci ont été publiés. D'autres projets encore ont eu lieu dans le contexte de la création d'une entreprise nommée Streamezzo dont j'ai été co-fondateur, et donc ont été peu publiés. D'autres enfin ont eu lieu dans des organisations de standardisation et la forme du résultat dépend de l'organisation :

- Formats :
    - MPEG-4 BIFS : conception de format de description de scène
    - XMT-A : format de description de scène basé XML pour BIFS
    - W3C SVG Tiny 1.2
    - LASeR: format de description de scène pour mobiles
    - OMA Rich Media Environment et 3GPP Dynamic Interactive Multimedia Scenes
    - Optimisation de format
    - Extensions pour faciliter les services
- Compression:
    - Encodage de scenes binaires: BIFS
    - Encodage LASeR
- Packaging:
    - Format de fichier MP4
    - Format de streaming SAF
- Création, outils auteur:
    - MPro: une interface graphique pour la création de contenu
    - Harmonia: des templates assistés par une interface graphique
    - B4: création de contenu à partir d'un format texte
    - RSP: création de contenu à partir d'un langage de programmation
- Adaptation/traduction:
    - Adaptation de contenu au contexte de rendu
    - Traduction vers un autre format: SVG vers BIFS, Flash vers « tout », PPT vers « tout »
- Rendu:
    - Qualité d'expérience
    - Multi-format: GPAC
- Elargir le spectre:
    - Compound Document Format
    - Widgets
    - HbbTV

Le rapport scientifique ci-dessous est un ensemble de sections décrivant mes contributions et celles de mon équipe sur chacun des sujets ci-dessus.

2. Summary

I have worked on many projects along the rich media chain. Some have been in the context of European or national projects, some in the context of PhD theses: these have been published. Some projects have been done in the context of the creation or running of a company called Streamezzo that I co-founded, and as such, have scarcely been published. Some have taken place in standards organization, and the form of the result depends on the organization:

- Formats:
  - MPEG-4 BIFS: scene description format design
  - XMT-A: XML-based scene description for BIFS
  - W3C SVG Tiny 1.2
  - LASeR: scene description for mobiles
  - OMA Rich Media Environment and 3GPP Dynamic Interactive Multimedia Scenes
  - Format optimization
  - Service-enabling extensions
- Compression:
  - Scene binary encoding: BIFS
  - LASeR encoding
- Packaging:
  - Multimedia packaging: MP4 file format
  - Multimedia streaming: SAF
- Authoring:
  - MPro: GUI to content
  - Harmonia: GUI-assisted templates
  - B4: simple text to content
  - RSP: program to content
- Adaptation/translation:
  - of content to rendering context
  - to another format: SVG2BIFS, Flash2any, PPT2any
- Rendering:
  - Quality of Experience
  - Multi-format: GPAC
- Widening the scope:
  - Compound Document Format
  - Widgets
  - HbbTV

The scientific report below is a set of sections on my contributions and my team's contributions to each of the above subjects.

3. Rapport Scientifique

3.1. Introduction

My work has been focused on interactive content since 1995. I chose to restrict this document to this subject, and not to speak about signal analysis on micro-computers from my PhD, speech synthesis with diphonems or VLSI computer aided design through symbolic design (analysis with the Stickizer or generation with Préforme).

I chose the term interactive content rather than interactive documents. There are two rather separate communities, one on content and one on documents. Content is closer to a video or audio stream: it may be subject to streamed modifications. Content evolves with time, whereas a document stays "the same", even if its internal state may change. Recently, using AJAX allows streaming modifications within documents, which reduces the difference in results, but not in design.

Along the work on interactive content, came a new buzzword, Rich Media, which we used and overused.

I have always been interested in graphics and interaction. VLSI computer aided design is all about user interfaces for browsing and editing large quantities of graphics. Entering the MPEG world, I soon focused on the mixing of media and interactive graphics, motivated for example by the creation of interfaces for the access to media (DVD). The state of the art of graphics formats in '95 was quite small. We had to start with description formats, compression, rendering and conformance. Then trying to promote the adoption of the format, the availability of authoring tools became an issue. Then the mobile world and its myriad of devices made adaptation, translation and quality of experience into major issues. Of course, progress was not linear, and different opportunities dictated the timing of going deeper into this or that aspect: available funding, meeting people… I tried to group projects into coherent aspects below, rather than keep a strict time line.

3.2. Interactive Graphics Formats

I have been involved in the design of many interactive graphics formats from '95 to now: BIFS (contributor), XMT-A (leader), SVG Tiny 1.2 (participant), LASeR (leader), RME (participant), DIMS (participant). I have also started a more in-depth study of these formats, trying for a sort of unification. And I have worked on a companion service-enabling format, necessary to the success of any interactive graphics format.

3.2.1. MPEG-4 BIFS

In '95, there were few scene formats. Flash was just starting. Apple lovers were either swearing by HyperCard [73] or by Director. 3D people were working on VRML. CD-I was the interactive solution for CDs, coupled with the MPEG-1 standard. OpenTV was just starting. The work on SVG did not start until '98.

The MPEG-4 Systems group, influenced by a wave of 3D enthusiasm, chose to base what became BIFS (Binary Format for Scenes) on VRML [65]. Because VRML did not have any 2D capability, the main extensions of BIFS over VRML are the set of 2D capabilities and 2D-3D mixing capability, and of course the binary encoding, the updating mechanism

and improved audio/video support [25]. The bulk of our contribution was on the 2D capabilities and on interactivity.
Based on a study of existing scene formats at the time, we proposed various sets of 2D objects:

- what we thought then as PowerPoint objects and properties,
- Bitmap, an unlimited surface that stays parallel to the screen, and is insensitive to transformations (even 3D), on which images and videos can be mapped efficiently (without a need for rescaling),
- Layout and Form, i.e. flexible layout primitives, Layout to be able to lay text and paragraphs out, including images for symbols, and Form for relative positioning and form-like layouts. The main drive was to be able to lay objects of an unknown size out at authoring time.

On interactivity, an analysis of the need for non-scripted interactivity led to the design of 3 nodes:

- Conditional
- Valuator
- InputSensor

3.2.1.1. Automatic Layout of Objects

The success of HTML as a standard is now obvious, but in 96, the emphasis was still on absolute object placement and fixed layout by the author. Layout determinism was the dogma. Still, we championed and got the addition of features allowing automatic layout of objects in the BIFS 2D renderer: automatic layout of text in paragraph, possibly interspersed with graphics such as symbols or inline images; and automatic layout of forms, as a very frequent use of interactive graphics. This work is one of the contributions of the PhD of Frédéric Bouilhaguet, together with the implementation of those nodes in the MPEG-4 Systems reference software [1].

Layout

The Layout node provides a simple paragraph of flowing text. Its parameters thus are the size of the region in which to lay the paragraph out, the characteristics of the paragraph (line spacing, writing direction, line progression, justification), whether the text should be wrapped and text strings broken if too long. They also include parameters for scrolling, speed and direction, by line or smooth.
The Layout node clips text and other objects that lie outside its region. It is a relatively straightforward feature, and the scroll feature makes it simple to create ticker tapes.

Form

The Form node provides relative layout (alignment of edges or centers) and spreading of objects, inside a region. The only complex part of this node is the mapping of the constraints onto the BIFS string or integer arrays. Each Form node has a set of children, a size and a set of constraints. All objects start at the center of the Form container. Each constraint is applied consecutively in the order defined. Each constraint applies to a set of objects. The constraints are the same as the ones that can be found in any graphical development tool. With an authoring tool, using this node is a breeze; this node is hard to use without an authoring tool.

3.2.1.2. Interactivity without Scripts

Conditional

One of the most important assets of BIFS is its update mechanism. I designed, implemented and proposed an update system for BIFS that was very close to the accepted one, albeit not as complete. I also pioneered authoring of scenes using scene updates in very innovative ways, without recourse to scripting, thanks to the Conditional node, which is a container of updates that can be triggered interactively.
Usually, updates are grouped in an Access Unit, and stamped with a Composition Time. A Conditional basically contains an Access Unit with no time stamp, waiting around for the user to trigger it. A suitable event source, e.g. a TouchSensor (in the case of a button), is routed to the Conditional, which implements the changes in the scene required when the user touches the button.
In terms of equivalent script, a Conditional is equivalent to a script where you can only use element creation, element removal, element replacement and property change, excluding variables, loops, conditions, etc… It seems very restricted, but it is amazing what can be done with this, compared to the negligible cost of implementation (negligible when compared to the cost of implementation of a scripting language and the DOM API [53]). In terms of efficiency, a study at Streamezzo found that Conditional was between one and two orders of magnitude faster than scripting.

Here is a real-life example of Conditional use. In order to create a button, we need three images: the normal image of the button, the image of the button when the mouse is over it, and the image of the button when the mouse button is pressed, i.e. when the button is pushed. Since the XMT listing is quite verbose, the repetitive parts of the listing are given in a kind of shorthand:

```
 1  <Group>
 2    <children>
 3      <TouchSensor DEF="N6" enabled="true"/>
 4      <Switch DEF="N2" whichChoice="0">
 5        <choice>
 6          ... normal state image ...
 7          ... over image ...
 8          ... down image ...
 9        </choice>
10      </Switch>
11    </children>
12  </Group>
13  <Conditional DEF="N10">
14    <buffer>
15      <Replace atNode="N7" atField="whichChoice" value="0"/>
16    </buffer>
17  </Conditional>
18  ... Conditional N11 with value 1 ...
19  ... Conditional N12 with value 2 ...
20  <ROUTE fromNode="N6" fromField="isOver"
21         toNode="N11" toField="activate"/>
22  ... ROUTE N6.isOver to N10.reverseActivate ...
23  ... ROUTE N6.isActive to N12.activate ...
24  ... ROUTE N6.isActive to N10.reverseActivate ...
```

Lines 1 to 12 pertain to the visual part of the button, showing one of three images according to the value of the "whichChoice" field. A TouchSensor node tracks the location and state of the pointing device and detects when the user points at a geometry contained by the TouchSensor's parent group. The Conditional nodes capture three actions: make visible the normal image, the over image and the down image, respectively. The TouchSensor fields are routed (lines 20-24) to the Conditionals (lines 13-19), so that mouse movements and clicks trigger the relevant Conditional.

Valuator

Content designers using scripts in VRML will tell you: most scripts are just one expression long: either just one cast, or one simple expression and a cast. A Valuator node is just that: one operation and a cast. Again, the benefit of this node is large compared to its implementation cost, and its efficiency is unrivalled by scripts.
In this example, an image or a video is shown in the middle of the screen. An object can be dragged with the mouse across it, and behaves as a magnifying glass on top of the image or video.
The object to drag is a Circle with wide yellow border, placed in a Group together with a PlaneSensor2D to make it draggable. Then we need the magnifying effect: a Transform2D with a scale factor of 3 (line 3 below) will have as single child a USE of the main image (line 4), and in order for the magnified image to follow the area below the "glass", we need to translate the USE (i.e. change the translation of the scaling Transform2D) by the magnifying glass movement multiplied by -3. This can be easily achieved with a Valuator node (line 8): it allows type casting and simple operations on events. Here is the XMT fragment for that, followed by the scene on Figure 1:

```
1     <Transform2D DEF="N0"><children>
2       <Layer2D DEF="N8" size="200.0 200.0"><children>
3         <Transform2D DEF="N6" scale="3.0 3.0"><children>
4           <Transform2D USE="N9"/>
5         </children></Transform2D>
6       </children></Layer2D>
7     </children></Transform2D>
8     <Valuator DEF="N7" Factor1="-3.0" Factor2="-3.0"/>
9     ...
10    <ROUTE fromNode="N1" fromField="translation_changed"
             toNode="N7"   toField="inSFVec2f"/>
11    <ROUTE fromNode="N7" fromField="outSFVec2f"
             toNode="N6"   toField="translation"/>
12    <ROUTE fromNode="N1" fromField="translation_changed"
             toNode="N0"   toField="translation"/>
```

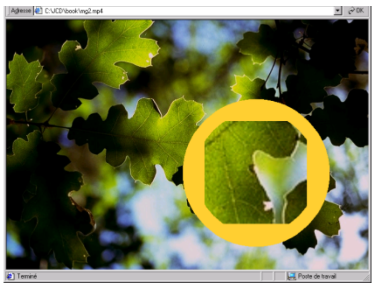


Figure 1: the Magnifying Glass scene

3.2.1.3. New Device Interfaces: InputSensor

VRML is poor in input device interfaces, beyond the pure 3D ones: you can interface scenes with joysticks, sliders, mice, trackballs and even some haptic devices, but the keyboard interface is desperately poor. Forget mouse wheels. Forget remote controls. Forget multi-touch or drag-and-drop. The MPEG-4 Systems group recognized this lack and tried to remedy it, but hit the following difficulty: for each new device, the device interface may change, and that means a new node, and that means a new BIFS amendment, with all the weight and inefficiency of that process. Adding a node means:

- changing the BIFS decoder;
- adding a new profile; even though this seems a non technical issue, the fact is that most standards are referred to through their profiles; as a result, every industry standard using BIFS will need to be updated before it can use a new device; it is not practical, specially if many new devices appear.

The design of InputSensor minimizes the size and complexity of BIFS amendments. In particular, the BIFS decoder does not need to be changed each time a new input device, with a new interface, is added. Old profiles are still usable. Hence, the problem of adding new input devices is separated, leaving decoder and profile stable.
For the purpose of InputSensor, an input device is modelled as:

1. a stream of information structured as device data frames (DDF) coming at random intervals;
2. each piece of data in a frame needs to be injected in the BIFS scene.

The content of a DDF is a set of any of the BIFS simple or multiple type. Here is an example of a DDF for a mouse with a wheel, which is something not supported by VRML97:

```
MouseDataFrame [[
 SFVec2f position
 SFBool leftButtonDown
 SFBool middleButtonDown
 SFBool rightButtonDown
 SFFloat wheel
]]
```

The first field contains the position, the next three contain the state of the three mouse buttons (true when pressed) and the last contains the wheel movement since the last frame.
The injection of data into the scene is specified with a command buffer similar to that of a Conditional. BIFS Commands of type FieldReplacement, IndexedFieldReplacement or NodeDeletion may be used. A FieldReplacement is used to replace the value of a whole field with the value from the DDF. An IndexedFieldReplacement is used to modify one value in a multiple field. A NodeDeletion is used to indicate that the corresponding DDF value shall be ignored.
Let us go back to the wheel mouse example, and assume that the author will use TouchSensor to deal with aspects in common with a standard mouse, and is only interested in managing the wheel through the InputSensor. The content of the command buffer will be:

```
SFCommandBuffer {
     NodeDeletion NULL; # ignore position
     NodeDeletion NULL; # ignore left button
     NodeDeletion NULL; # ignore middle button
     NodeDeletion NULL; # ignore right button
     Replace NodeID.FieldID with 0; # place wheel info in NodeID.FieldID
}
```

The value 0 in the Replace is a placeholder and will be replaced by data coming from the device when a DDF arrives.

In practice, the +1 or -1 of the wheel will be put in a Valuator, which is routed to a translation field of a Transform2D node, whose child is e.g. the button of a slider.

3.2.1.4. Analysis

An equivalent of Layout has been added to SVGT1.2 [51] in the form of the element textArea, proving the validity of the functionality in the context of a graphics format. Although the feature of flexible layout was repeatedly asked for, and almost made it into SVG, there is no equivalent of Form yet in SVG; it is possible to cover some of the Form functionality with the help of CSS. Our successful contribution of transformBehavior in SVGT1.2 is an equivalent of the Bitmap functionality. Conditional found its way into LASeR quite naturally. Valuator did not, because there are few trivial scripts in SVG and since the attribute types are much more complicated, simple type casting just does not work. Covered by our patent WO02056595, InputSensor was perfect for the specific context of BIFS, but would be useless in SVG, where user interaction is done with events; new types of events are relatively simple to add. Our work with cartoons started then [25].

3.2.2. MPEG-4 XMT-A

I led the creation of an XML-based format for BIFS, initially designed entirely (and naïvely) in binary. Everyone working on BIFS ended up using a format called BIFS text, which was hacked VRML. An official text format was deemed necessary.
Since XML was trendy and has very nice properties [54], an XML version of BIFS was created. XMT was created in collaboration with IBM, who was interested in importing SMIL features in BIFS, through the design of XMT-O. I designed XMT-A, i.e. the part with one-to-one equivalence to the existing BIFS nodes.
Among the various motivations for the design of an XML version of BIFS, ours were:

- capability to intermix metadata / authoring information within the content, not possible directly in BIFS since there is no way to intermix private data within the binary encoding;
- use of XSL to generate XMT
- use of standard/open source software to ease parsing
- use of XML schema to help checking the syntax
- easy extensibility of the format

The work itself was straightforward, using XML Schema [57] as the format formal specification. Nodes were implemented as elements. Fields were implemented as attributes. Even MFFields were implemented as attributes, with lists of SF types separated by spaces. The only text content is used for the scripts. Children of a node are wrapped in a <children> element, and so are most sub-elements (geometry, appearance, etc). This makes for a format that is easy on programs, but very verbose and thus not so sympathetic with hand authoring. Yet, checking your production with the schema will catch many construction errors, which would not be the case with a terser but looser format. For example, any sub-element type restriction error will be caught by schema validation.
I designed the schema together with a parser and generator within our mp4tool implementation (Java), constantly testing the applicability of design choices and the amount/complexity of coding generated by syntax choices.

This work was done in parallel with the big push for BIFS conformance, and the use of XML technologies and related open source software helped both XMT and BIFS conformance [22].

3.2.3. W3C SVG Tiny 1.2

Scalable Vector Graphics, or SVG, is a W3C standard started in the mid 90s. It had a development parallel to that of BIFS and finalized its first version in 2001. When we researched the state of the art in 2003, looking forward to the LASeR call for proposals, SVG 1.1 was finished, a mobile profile SVGT1.1 was defined, and a version 1.2 was being developed [51]. Our analysis was:

- SVGT1.1 was too large and complex for current and near-future mobile phones (in 2003!), and hopeless for Java phones, which were the bulk of the medium class phones. With hindsight, we were so very right.
- SVGT1.1 did not have video and audio, but SVGT1.2 would have.
- SVGT1.2, even bigger than SVGT1.1, was too large.

I still participated in the SVG working group for a few years, first trying to add features we deemed necessary, such as scene updates, then trying to convince the group of the necessity to have a smaller profile, and finally to just liaise with MPEG.

3.2.3.1. Updates

Coming from MPEG, the notion of defining a scene progressively was so natural, but to the SVG WG document-oriented vision, it was just a bizarre contraption at first, and rejected. After two years, the pressure and use cases from LASeR led them to try to define a way to express updates of an XML tree: Remote Events for XML, or REX [55]. The use cases included a travel ticket reservation service, where more elements are added progressively to the form as is it filled.
When an XML tree is modified, events are fired, such as DOMAttrModified when an attribute is modified. The principle of REX was: in order to specify a scene update, you specified the event that the intended update shall generate. Upon reception of a REX event, the target scene tree is modified so that a similar event is generated. On top of this, a way to specify timing was added.
REX is now dormant for licensing reasons: the W3C statutes do not allow to publish a standard that is not "royalty-free" (as defined in its process document), and REX would not be RF.

3.2.3.2. Image rendering

In SVG 1.1, there is no way to render an image exactly at its media size. An image is always resampled, even shown at a scale of 1. This is all very well on a PC, with graphical acceleration, but was impossible on a mobile phone (specially in Java). This was relatively easy to convince the SVG group of. As a result, SVG T 1.2 includes an MPEG-inspired attribute called transformBehavior on the video element, which defines whether the video is resampled or not, and includes rotations of 90, 180 and 270 degrees as a bonus.

3.2.3.3. Memory Footprint

We have noticed while implementing SVG that there is a combination of features that mostly forces a tripling of the memory footprint and slows down the processing. Here is a simple description:

- upon loading of the scene, the value that is parsed from XML is put in the scene tree. This value is called the "DOM value". It needs to be kept around to restore it at the end of animations, or for scripting, or for serialization.
- because of CSS inheritance of properties [52], a property may "trickle down" the scene tree onto nodes that do not specify this property but on which the property has a meaning. This is called the "computed value" and its access is needed because of scripting access to the tree.
- finally, the "animated value", constructed from the DOM value and the computed value and evolving with time (as defined in SMIL animation [50]), needs also to be stored because of the script functions giving access to traits.

In other languages, only one value needs to be stored. In SVG, there are many properties and with scripting, no assumption can be made at load time about which node will be animated or not: as a result, the memory footprint of the scene tree is up to three times larger. Note: It is possible theoretically possible to avoid the x3 increase in storage, at the cost of extra processing. I have no evidence of a successful and compliant implementation of that possibility: for example, the GPAC implementation is not 100% compliant in that area.

3.2.3.4. Processing Overheads

CSS inheritance forces a pass on the scene tree to propagate property values each time properties change. Properties may be animated, and the animated value may be inherited. As a result, there is a need for a complex pass, mixing CSS inheritance and animation.
Node reuse with the use element creates a "shadow tree" (i.e. a live copy) of the reused node; the shadow tree inherits from its ancestors, not the ancestors of the used node, but the shadow tree is animated by animations similar to those applying to the used node. As a result, animations need to be duplicated and applied on properties inherited in the shadow tree context.

3.2.3.5. Analysis

Rectangular clipping failed to make it into the SVG T 1.2 spec, even though this is, in practice, very useful for authoring user interfaces. But the SVG conformance text was very significantly developed as a result of the LASeR liaisons to SVG.
SVG Tiny 1.2 is quite heavy in both footprint and processing: the resistance to profile these problems away comes from an attempt to keep the number of profiles as low as possible, and the perception that performance of terminals is going to keep increasing at a fast pace. The resistance may be similar to that of Macromedia/Adobe, which failed to appropriately profile Flash into FlashLite for mobiles.
SVG Tiny 1.2 has failed to get significant traction in the industry. There are very few mobile services in SVG, despite reported hundreds of millions of handsets supporting SVG. I believe this is the result of a failure to size the SVG mobile profile correctly. But SVG 1.1 Full is very successful on PCs. All major browsers are implementing it together with HTML5, in a move probably intended to replace Flash.

3.2.4. MPEG-4 LASeR

3.2.4.1. Introduction

When in 2001, we wanted to focus on mobile rich media, there was one obvious candidate: MPEG-4 BIFS [43] [44]. But BIFS has been designed for PCs in mind, and, for example, a generic processor implementation of BIFS binary encoding requires at least 200Kb just for the decoding tables. Those 200 Kb were definitely a killer for mobile phones of that time. The hope then was to be able to implement the whole decoder + compositor + renderer in 200 Kb or less, preferably 50 to 100 Kb. Then started a lengthy process of trying to tweak BIFS into something suitable for mobile implementations.
BIFS Core2D is an extremely restrictive profile, but it does not fit the bill. The limitations on drawing only triangle or rectangles are inadequate. Yet all the useful BIFS objects are in later versions, with longer (and less efficient) tags, requiring bigger decoding tables.
For some time, predefined profiles were investigated as a means to:

- create new objects without the long and inefficient decoding tags of later versions, and without needing newer and bigger decoding tables;
- reuse the new, more powerful objects of newer BIFS versions, in restricted situations, framed by the use within predefined profiles;
- have a standard immediately usable in existing, more powerful BIFS renderers on PCs;
- allowing ad-hoc implementations of the predefined profiles, not as profiles, but as if they were specific objects. These implementations could be optimized and small on mobile devices.

The work on predefined profiles made clear that BIFS was really not suitable for implementation in Java 2 Mobile Edition (J2ME), which represented the largest addressable market of phones capable of downloading apps. As a result, the LASeR project was started, with the usual MPEG process: searching for companies to declare an interest in defining a new, simple and efficient scene description format for mobiles; writing and sending out a call for proposals; choosing among proposed solutions; and improving on that solution [11] [14] [15] [47].

Objectives

The main LASeR requirements were to:

- Support an efficient, compressed, streamable, extensible and easy-to-decode representation of rich media scene data, compatible with SVG Tiny to leverage existing market traction and content development tools;
- Allow small profiles definition, including one profile for J2ME implementations; “small” here means leading to compact implementations, compatible with handheld devices with limited resources;
- Allow the representation of differential scenes, i.e. scenes meant to build on top of another scene;
- Be designed in such a way that implementations can be as compact and fast as possible, require as little as possible runtime memory and be implementable, at least partially, in hardware.

Target applications

Six main target applications were considered in the development of LASeR: (a) rich media portal, (b) interactive mobile TV, (c) interactive screen saver, (d) podcasting, (e) home convergence nexus and (f) device user interface.

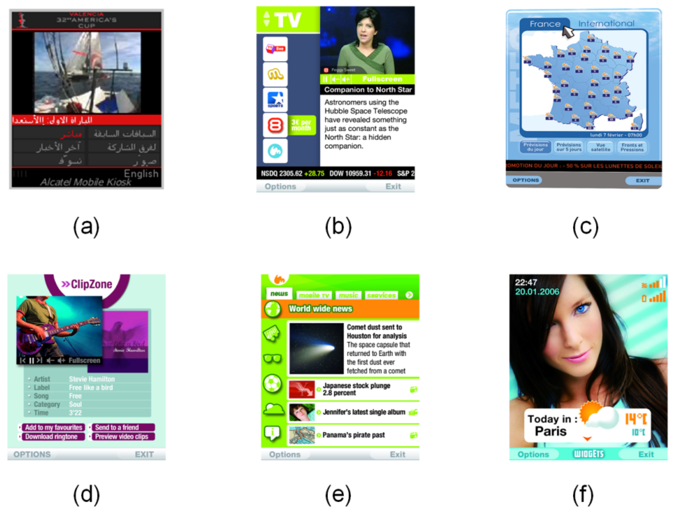


Figure 2: LASeR-based rich media services.

3.2.4.2. Technology

Architecture

Figure 3 describes the architecture of a LASeR implementation. Its modules are described in the next section. An application first shows to a user an initial scene, which is an *SVG scene (1)* with *LASeR scene extensions (2)*. The application then changes what is shown to the user by way of *dynamic updates (3)* to the scene tree. A *binary encoding for compression (4)* is used to improve transmission delays. Audio, video, image and font decoders contribute to the LASeR engine rendering. Applications can be built on top of a LASeR engine. The LASeR engine relies on the services of a transport layer, possibly complemented by *SAF (5)* to provide missing packaging and synchronization features.

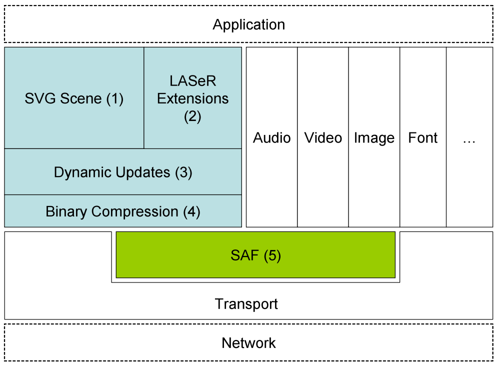


Figure 3 – Modular architecture of a LASeR implementation.

Tools

This section describes the most important tools of LASeR:

- SVG Scene Tree Representation – LASeR reuses relevant SVG elements [51].
- LASeR Scene Tree Extensions - We have identified areas where extensions are needed to allow the development of efficient services with LASeR:

- simple pixel-aligned rectangular clipping.
- restricted, non-resampling 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 – LASeR has scene updates similar to BIFS as well as 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 is described in a later section.

• Audiovisual support - LASeR includes SMIL2/SVGT1.2 audio and video elements as well as the additional SMIL2 MediaClipping module for VCR-like media control. Audio and video streams are carried beside the LASeR stream and referred to by binary identifiers.

• Usage of Font Information - Both SVG and LASeR allow content creators to embed font information within the scene. However, the SVG fonts solution was deemed too limited by MPEG. MPEG recommends the carriage of the font information beside the scene information as a media stream. The exact format is optional. One option is 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 rich media services rely on a key feature of LASeR: incremental scenes, made possible by the scene segments. A scene segment is a LASeR stream containing not an independent scene, but an addition to another existing scene. From a server-side point of view, the interactive services can be considered as a 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, it is the same scene/service that 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. SVG, Cascading Style Sheet (CSS) and Synchronized Multimedia Integration Language (SMIL) were designed for the PC platform, and although SVG Tiny defines a profile that is implementable on mobile devices, some concerns have been raised by implementors in 3GPP (Third Generation Partnership Project), OMA (Open Mobile Alliance) and MPEG, that it is not possible to make implementations small and efficient.
We created a profile of the LASeR specification named 'LASeR Core'. In order to allow efficient implementation and high rendering performances, LASeR Core restricts the

specification in the following areas: no bitmap resampling, no dash capabilities, no gradient animation, no inheritance, and animation restricted to a simple mode which ensures a smaller memory usage, 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. It contains most of the features of the third version of LASeR.

Comparison with other standards

LASeR 2nd Edition is a superset of SVGT1.2, adding scene extensions, binary encoding and LASeR commands. LASeR is also a superset of 3GPP DIMS (Dynamic Interactive Multimedia Scenes), which has adopted SVGT1.2, some of LASeR scene extensions and most LASeR commands.
Just like 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 2D, 3D and mixed 2D/3D scenes, has a very large object set and a complex encoding. Comparatively, LASeR is focused on 2D scenes only, has a much simpler object set, its decoding is much simpler (no floating point operation) and its implementations can be used satisfactorily on devices at least twice smaller in code, memory footprint and performance.

3.2.4.3. Performance

The main functional improvement of LASeR over SVGT1.2 is the dynamic updates. Implementing dynamic scenes in SVGT1.2 involves using scripting (mostly with interpreted ECMA-Script [64]), the DOM (Document Object Model) interface [53], a dedicated server, TCP connections and lots of XML manipulation. LASeR Commands allow authors to create dynamic scenes with declarative text rather than programming, and are multiple orders of magnitude faster and smaller to implement.
The main performance improvement in LASeR over SVGT1.2 and DIMS is the streamable binary encoding. The LASeR binary encoding achieves on average more than a factor of 2 compression improvement over gzipped SVG/DIMS. The difference is even bigger on small scenes or fragments. On the client, the author's consistent experience is that decoding a binary stream is a lot faster than parsing an XML document, thus adding the advantage of better client reaction time to the shorter transmission delay brought by better compression.
The LASeR Core profile is also a key complexity advantage. 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. The LASeR Core profile, by being implementable on J2ME, gives access to a wide range of lower-end existing and future phones, which despite predictions, still constitute the majority of the market.
Discussions of Moore's Law and claims that in a few years, performance/code size problems with SVGT1.2 implementations are going to disappear have been made irrelevant by the current evolution of devices. Moore's Law remains valid, yet the perceived performance of new phones is not increasing, but decreasing. Because of limitations of the batteries, the processors are not growing as fast as they could be, while display screens are growing very rapidly. The amount of processing power per pixel is going down: screens have grown by a factor of 4 while the processor speed has increased by a factor of 1.5 to 2. As a result, the load on processors to drive the larger

screens becomes heavier. The only way to compensate this added load is the adjunction of a graphics/multimedia coprocessor, but this hits the battery too. Since the fluidity of rendering is a major contribution to the feasibility of graphic animations, an achievable frame rate dropping by a factor of 2 will render a particular application running well on one phone, irrelevant for another phone with the same processor but a twice-larger screen. The same discussion applies to the evolution of set-top-boxes, whether for IPTV or for broadcast.

3.2.5. 3GPP DIMS and OMA RME

3GPP DIMS (Dynamic Interactive Multimedia Scenes) [67] and OMA RME (Rich Media Environment) [69] are two attempts to push MPEG LASeR in an industry standard, to improve its potential. Indeed, MPEG standards need to be picked up by industry forums and set into larger contexts. 3GPP defines handset environments from radio to media decoders, and OMA defines application environments for handsets. OMA goes higher than 3GPP in the OSI reference model, but there is some overlap. After some needed clarification, it was agreed that 3GPP DIMS dealt with scene and RTP (streaming) transport, and OMA RME dealt with everything “higher” and non-3GPP, for example the use in broadcast environments.
Figure 4 shows the respective positioning and main elements of SVGT1.2, LASeR, DIMS and RME. SVGT1.2 is included in LASeR, and the scene subset of DIMS and RME are included in LASeR, but both define elements outside of the scene sphere, in Transport. DIMS defines an RTP payload. OMA defines additional elements, such as UAProf and application to broadcast.

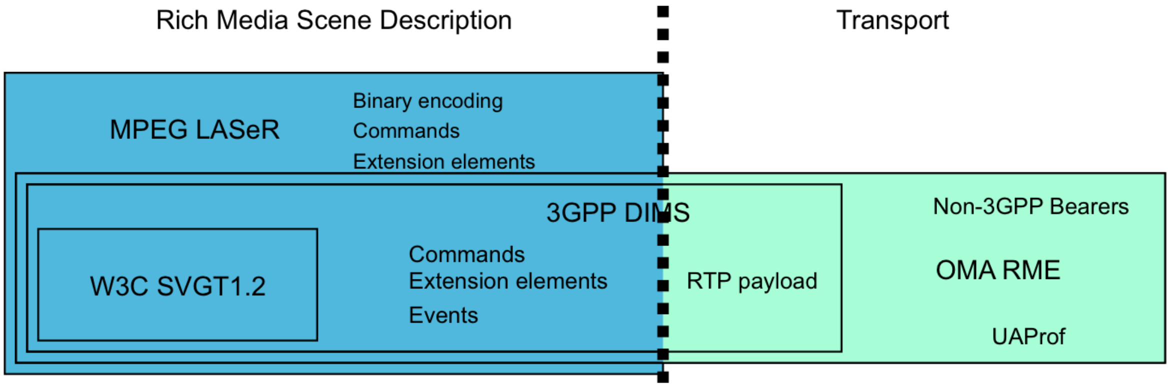


Figure 4 - Respective positioning of SVGT1.2, LASeR, DIMS and RME

To get a meaningful idea about the respective sizes of these four standards, the specification text for SVG T 1.2 is more that 800 pages long, DIMS is 40 pages, RME is 31 pages and LASeR barely over 100 (plus the specification of the binary format, which is very verbose).

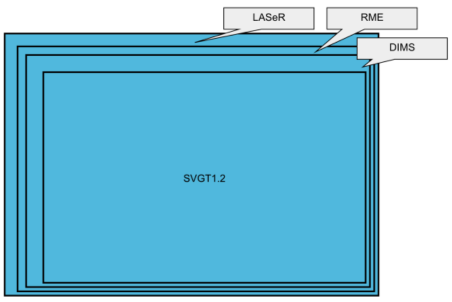


Figure 5 - Respective sizes of SVGT1.2, LASeR, DIMS and RME

DIMS cherry-picks features from LASeR, including updates, screen orientation management and preferences saving, but not the binary encoding nor the Core profile. OMA respects those choices. As a result, DIMS and RME have the already reported footprint and processing problems of SVG T 1.2.
Figure 6 and Figure 7 give an indication about how LASeR Core tried to solve the footprint and processing problems: LASeR Core profiles away a relatively small number of features, hence its "spec size" (i.e. the number of included features) is close to that of the others. LASeR Core forbids the definition of properties but on the nodes that will use them, which in effect removes CSS inheritance, because the nodes that use CSS properties are leaf nodes and CSS stylesheets are not permitted. LASeR Core also removes the need for keeping DOM values. See previous sections for more detail.

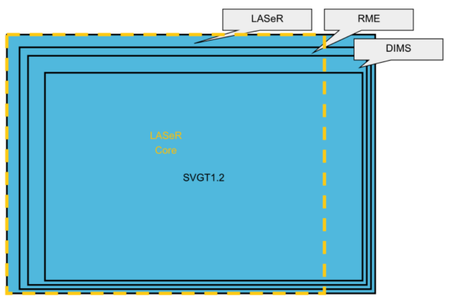


Figure 6 - Spec sizes including LASeR Core profile

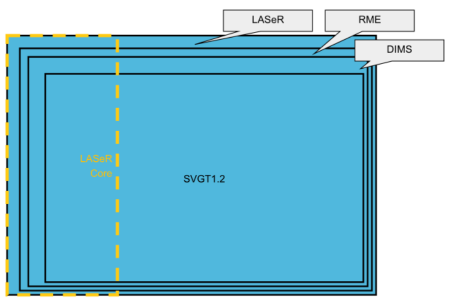


Figure 7 – Respective footprints

3.2.6. Formats Review and Optimization

This is Cyril Concolato's PhD [5]. His document consisted in two parts: a review, analysis and classification of existing formats, and a set of contributions.
In his first chapter, he introduced HTML, Flash, VRML, BIFS, SMIL 2.1, SVG Tiny 1.2 and LASeR. In his second chapter, he explored animation: by interpolation, frame-based and by program. In the third chapter, he explored compression techniques: use, prototypes, templates, number encoding and quantization, and structures such as planar maps. In the fourth chapter, he explored interactivity: event-based, declarative or program, client-side or server-side. In the fifth chapter, he explored delivery of scenes: file or stream.
In his first contribution chapter, he explored all the above in the context of (2D) cartoons [23], describes experiments in many of these formats [24] and possible optimisations in compression, delivery, playback footprint and adaptation [20] [21]. In the next chapter, he described a joint, optimized implementation of BIFS, SVG, LASeR and some of Flash, and his multiple attempts to obtain a solution that compares favourably with industry standards.
In MPEG, he spearheaded the work on Advanced Text and Graphics, a BIFS amendment where many missing features found in SVG and Flash were fitted into BIFS.
Cyril proposed some improvements to scene description formats in his conclusion:
1) use scene updates for streaming or progressive download, as opposed to progressive parsing of documents: I still agree technically, but I now doubt the need for streaming scenes: incremental scenes in response to interaction are needed, but streamed scenes, i.e. time-driven regularly updated scenes are not.
2) use compression with quantization: I doubt that compression is realistic (politically, commercially, etc) for scenes, when it only brings a factor of 2 over gzip, which is

free and already everywhere. Generic binarisation of XML, such as W3C EXI, may be used.

3) use signalling of branches of the scene tree that will never be used (e.g. not accessed by scripting) after initial rendering, keeping the composition graph will be enough: very relevant.
4) use signalling of static vector graphics: these can be rendered once and then replaced by their image, thus reducing the size of the scene tree and of the composition graph: it exists in SVG as a hint, it should be stronger than a hint.
5) use a scene structure with 4 branches: dictionary, interaction, timing, display. Each branch is processed only during part of the rendering, and the number of "walked" nodes during rendering is greatly reduced: very relevant.
6) use styles rather than property inheritance, again, to reduce tree processing: relevant.
7) use explicit geometric transforms mostly and automatic layout only when needed: because of the growing importance of content adaptation, I feel automatic layout is needed quite often.
8) use a single time base in the scene, and make time dependency explicit by using listener elements rather than activation lists: the complexity of the so-called SMIL timing is not so useful in practice, to say the least.
9) replace implicit propagation of properties and animations with explicit links (such as ROUTEs) which do not need additional tree walking.
10) remove the capture and bubble phases of event propagation, again for simplicity and because they are used very infrequently and are not interoperable.
11) reduce script by using Conditional whenever possible, for performance reasons.

3.2.6.1. Analysis

Formulating concrete recommendations was a very important conclusion to this review and optimization work. The recommendation to use ROUTEs (9) and Conditionals (11) is converging very much with the links in NCL [72], which integrate explicit propagation of events with resulting actions. However, there is such a non-technical pressure for HTML5 / CSS in the W3C sphere of influence, for Flash in more commercial spheres, and for various other formats pushed by individual companies for various reasons (all very good in their own context), that such proposed improvements have no chance at all to ever find their way into a well-used standard.

3.2.7. Service-Enabling Extensions

Take LASeR in a mobile context: the standard was designed for use in a mobile world, so designing a mobile service should be easy. Yet, if you want to add to your content something typical of mobile usage, such as "click to call", there is no standard solution.
There is a notion that is largely overlooked in academic work on rich media and interactive graphics: service-enabling extensions.

3.2.7.1. State of the Art

Within industrial fora such as Joint Innovation Lab (JIL), Wholesale Applications Community (WAC), Open Mobile Alliance (OMA), Open Mobile Terminal Platform (OMTP), Consumer Electronics Association (CEA) or Open IPTV Forum (OIPF), there are multiple standardization efforts on ECMA-Script APIs for domain specific services [36] [37] [38] [39]. Each defines a way to access the contacts, the agenda, the camera and

other mobile-specific functions such as battery level and network reception, some with an emphasis on widgets or the interactive TV domain (see later section on HbbTV). Proprietary solutions also exist, such as mobile extensions to ActionScript in Adobe FlashLite. More recently, the W3C started work in this area: access to system information, to contacts, calendar, messaging, etc also as ECMA-Script APIs. This is a work in progress.
All the above are instances of scripting APIs. Other possible solutions include:

- declarative: creating a set of XML elements describing the various actions, à la LASeR PMSI[76]; this means extending the host format, adding the new objects.
- protocol: using the existing interface of the scene with the outside world that is the hyperlink, by designing a new protocol; this does not require a change in the host language.

3.2.7.2. The cmd: Lightweight Solution

In the Streamezzo tool chain, the need for service-related extensions was fulfilled by a set of commands not relying on ECMA-Script obviously, because the Streamezzo player did not include an ECMA-Script interpreter. We used URLs with a private protocol cmd:, initially inspired by the tel: protocol (described in IETF RFC 2806). Since server-side actions were all implemented as triggering an HTTP GET on a server URL, it felt natural to implement local actions by triggering special URLs which would be filtered and interpreted by the player.
The following functions were thus implemented:

- device actions: volume up/down/mute/unmute, backlight lock/unlock, …
- phone actions: click to call, send SMS/MMS…
- personal information manager (PIM) actions: contacts and agenda actions.
- EPG requests/actions for TV services

The typical structure of these commands is: `cmd:<command> <parameters>`
The <command> part is usually constant. The <parameters> are often dependent on user data. The Streamezzo rich media format, which is an extension of the LASeR CD format, contains instructions to concatenate strings with constant strings, with strings found in texts in the scene, and with numbers found in some fields in the scene. These instructions make the construction of the <parameters> of commands rather straightforward. LASeR content would use the update command Add, with operandId and operandAttributeName to retrieve a value in the scene, for the construction of <parameters>.
All commands have a return value, if only an error code. When an error code can be expected, the command includes the ID of a Conditional in the scene, which will display an error. When data needs to be provided to the scene, the command parameters include the IDs of the scene elements, which will receive the returning data.

3.2.7.3. Analysis

At Streamezzo, because of a large Java mobile base, and the constraint to allow the downloading of the application, we could not adopt ECMA-Script. But everyone else does. So our declarative solution was a singleton, efficient but difficult to reuse elsewhere. Functionality available in 2005 from Streamezzo is only now made available interoperably in higher-end phones.
In terms of authoring and compression, there are problems:

- most complex services have a large number of protocol-related strings, which are not compressed because there is no entropy coding of strings in the LASeR binary format;
- the IDs of the elements that are used in the commands, are only known after binary encoding, or need to be forced to a constant.

However, these problems are small compared to the benefits in terms of development:

- the commands are developed, processed and tested separately from the scene, and the scene description language does not need any modification.
- the constant evolution/addition of commands does not impact the scene engine.

Security implications are the same as that of using scripting. But static checking can be done on the scene, by reading the plain text strings from the binary access units, to detect if a particular scene uses a "dangerous" or "illicit" command in a context.

3.3. Compression

MPEG is very keen on compression. While the endless search for more compression is obviously useful in audio and video, the need for scene compression is not so obvious.
The first argument is one of relative sizes: are scenes as big as audio or video? The answer is quite different for 3D and 2D. 3D scenes are large and their compression is fully justified. 2D scenes are mostly small, very small relative to any of the media that 2D scenes compose: most 2D scenes are smaller than the images or the audio clips they use, and almost always much smaller than even short videos. (2D) Maps are an exception, which is closer to 3D content in terms of characteristics.
In the mobile domain, specially 5 years ago, compression was seen as a must to improve service response times, hence the push for a LASeR encoding.

3.3.1. BIFS

The compression effort of BIFS was led by people interested in 3D [34]. Many of the BIFS compression features are much too complex, or even useless in 2D. Even though I was mostly interested in 2D, I became an early implementer and tester of BIFS with the development of mp4tool, and later led the conformance work. I have organized arguments on BIFS encoding in pros or cons below.

Pros	Cons
Few basic types	Extensions are inefficient
Contextual node encoding is optimal for compression of initial BIFS version	Contextual node encoding causes large decoders and is not efficiently extendable.
Quantization	Quantization configuration is inefficient
Good on large content	Sensitivity to bit errors

Pros:

- There are very few basic data types, which are reused everywhere. This is very good to reduce the complexity of decoders, and forces some order where other formats like SVG have little logic.
- Contextual node encoding: everywhere a sub-element is expected, the minimal number of bits is used. This is optimal in compression efficiency.
- There is a complex, potentially efficient quantization mechanism.

- The encoding is quite good on larger content (3D).

  Cons:

- Extensions (adding new nodes) are difficult to design, counter-intuitive and inefficient to use; no time was spent on globally optimizing the initial BIFS version (despite my warnings) and later extensions, thus yielding an optimal initial version which is never used.
- The contextual node encoding means large decoding tables, documenting which code to use for which node in every possible context.
- The contextual node encoding requires large, error-prone documentation which is impossible to write manually, and very difficult to check after automatic generation.
- The contextual node encoding is not efficient to extend, thus the efficiency of the version 1 encoding is lost when using many nodes defined in later extensions.
- Even though quantization is efficient on e.g. 3D scene, the configuration of quantization is done through a node (QuantizationNode) whose encoding is very inefficient, thus loosing much of the quantization improvement.
- Quantization is configured by inserting a special node in the flow of other nodes, which is very error-sensitive and counter-intuitive;
- The optimality of the encoding makes it very sensitive to bit errors. Roughly, a BIFS packet is composed of structure (20-30%) and attribute values (70-80%). Any bit error in the values means the scene is wrong, but decoding can proceed. Any bit error in the structure means the decoder will soon be forced to stop. Error recovery is not possible without protection, i.e. adding more bits.

Last but not least, many of these problems would be relatively easy to solve or at least reduce, by applying simpler techniques, which may not be local compression optima, but are more globally efficient.

3.3.2. LASeR

LASeR was designed for mobiles and constrained devices. Its binary encoding had the requirement of a small, simple decoder together with coding efficiency. BIFS, with the contextual node encoding and the large decoding tables, did not fit the bill.
My initial ad-hoc encoding was very limited and minimal. It had little extensibility, the various bit fields were sized for the mobile screens of 2004 (and would be completely outdated by now). But every attribute was smartly encoded: as simple as possible, yet efficient. For example, there were no divisions required in the decoder.
The BiM [48] encoding had the following properties:

- an encoding is derived automatically from a schema; this is very handy in standard design when things get added or changed up to the last minute;
- it was optimized for many devices, including set-top-boxes, which have usually even less power and resources than mobile phones; indeed, the complexity is on encoders;
- it used the type vluimsbf5 for indices; vluimsbf5 stands for variable-length, unsigned int, most significant bit first, with one bit to signal whether the next nibble is the last or not; vluimsbf5 is both efficient and adaptable in every situation;
- initially, its attribute encoding was quite poor, limited to the data types defined in XML schema: strings, integers of various sizes, floats, enumerations.

In SVG (and thus in SVG-compatible LASeR), there are lots of attributes for each element: up to 60 possible attributes, some being CSS properties that have nothing to do with the element, but are important for its children. And there is no logic to the design of attributes, no standard set of attribute types: each attribute has its own logic and possibly, a specific grammar for the representation of the value, e.g. for the d attribute of the path element. As a result, BiM, with its poor attribute encoding and not being designed with so many possible attributes per element, had no chance against our ad-hoc encoding.

In order to be able to compete, BiM had to be extended to include what was dubbed “type codecs”, i.e. the ability to include in the XML schema certain attribute types which detailed encoding will be dealt with by specific encoders. It is a sort of escape mechanism to go into ad-hoc encoding wherever the generic BiM encoding is insufficient. In a sense, this feature breaks the generality of BiM, since not every BiM implementation will be able to decode any BiM schema: a BiM decoder will not be able to decode a bitstream which uses a type codec that is not implemented in the decoder.

So the ad-hoc encoding was studied to define all the type codecs and port them into the BiM encoding. This was not enough. The large number of attributes per element forced another departure from BiM generality: the occurrence frequency of attributes (per element) was studied, and attributes were split into two categories: frequent and rare. Frequent attributes were left in the encoding schema and encoded with BiM. Rare attributes were removed from the encoding schema and encoded with a pseudo-type codec.

The final comparison results are that my ad-hoc encoding was still 6-8% smaller than the improved BiM encoding, over the agreed test set of many hundreds of SVG and LASeR scenes.

In the end, it was agreed to switch to a “BiM-compatible encoding for LASeR”. This meant you can use a BiM decoder to decode a LASeR stream, but the encoding is also fully defined without reference to BiM so that a specific LASeR decoder can be designed.

Note: there are XML-specific aspects that are ignored by this encoding, such as processing instructions, entities and the difference between specified value, default value and lacuna value. As the focus of LASeR was on the binary version, these aspects were deemed secondary and the lack of their support acceptable.

3.3.2.1. Analysis

Looking back, BiM has another problem that was revealed with LASeR extensions. BiM deals very gracefully and simply with evolving XML schemas, by resending a complete new schema with extensions, not by sending skippable extensions within a backward compatible bitstream. Skippable extensions to existing schemas are possible, but they are a pain to design and manage. LASeR thus inherits that problem.

The LASeR encoding ended up a lot better than our initial proposal: the LASeR encoding is extensible in every way, but for some of the choices embedded in the type codecs; a lot of extensibility was added because of the competition with BiM, and I now regret that I did not propagate this into all type codecs; for example, some numbers are still 24 bits fixed point when it should be possible now to have 32 bits fixed point.

In terms of implementation, our first J2ME LASeR CD decoder was less than 10Kb in a complete player of less than 60 Kb. The LASeR IS J2ME player is about the double, more than 100Kb, and the decoder is less than 20Kb. It is difficult to tell what is the influence of the increase of number of elements, and what is the influence of switching to BiM-compatible encoding.

To finish on encoding, as was already mentioned, it is not obvious that a specific encoding is necessary for scenes. The ubiquitous gzip could be enough, even though the LASeR encoding is approximately twice better than gzip over the test set. It has to be said that if there are large quantities of text (which happens, but less often than with (x)HTML content), gzip will perform significantly better than the LASeR encoding, because of the lack of entropy coding of character data in LASeR.
From the terminal implementer perspective, the decision is more complex:

- XML parsers are rather more complex, have a larger code and memory footprint than binary decoders.
- the SVG syntax for attributes has many options, which are usually dealt with in encoders, and binary decoders do not have that burden.

So the terminal implementers may prefer a LASeR-style binary format even if it is not so much more compact than XML + gzip.
Since then, W3C worked on Efficient XML Interchange [61], a binary encoding for XML, which may be an option.

3.4. Multimedia packaging

As soon as there are multiple media, each media being coded separately, there is a need for grouping encoded media together:

- for storage;
- for transport/streaming;
- for synchronisation.

These goals are not all fulfilled by all solutions. I have worked on two solutions, the MP4 file format and the SAF aggregation format.

3.4.1. MP4 file format

3.4.1.1. History and Contributions

At the end of 1997, in Fribourg, MPEG issued a call for proposal on an MPEG-4 Intermedia Format. At the next meeting in San Jose, there were two answers, one from Microsoft with ASF, and one from Apple with the QuickTime File Format, supported by IBM, Oracle, Sun and others. The second was chosen and work started [26]. I soon started an implementation named mp4tool funded by the ESPRIT project MPEG-4PC [34]. This implementation became the first freely available tool to generate MP4 files, released to the MPEG community at the Vancouver meeting in July 1999. This tool was also used for the generation of most of the BIFS test sequences, a work that ran in parallel with the MP4 file format development.
The MP4 file format was later split into ISO base media file format [45] and MP4 file format [46], in order to allow other standards, like 3GPP, to build on top of the generic part.
As part of his PhD [1], Frédéric Bouilhaguet also made a study of MP4 file format for use in download-and-play or quasi-streaming.

3.4.1.2. Analysis

The MP4 File Format has the following capabilities:

- store MPEG-4 streams with all data and metadata (synchronisation and other);

- store whole presentations;
- easy to edit one stream without changing the rest of the presentation;
- easy to extend, easy to skip extensions thanks to the atom/box model;
- relatively easy to play a presentation;
- has lots of options about how to arrange media, inside or outside the main file, possibly interleaved;
- allows random access to any part of the presentation, including the signalling of random access points;
- supports hinting formats, easing the conversion to streaming;
- supports XML-based scene formats.

It also has limitations:

- it is not streamable in itself, and a significant amount of work is needed to make it streamable, as is proven by the large amount of effort invested in MPEG Dynamic Adaptive Streaming over HTTP (DASH);
- it was never designed to transport images efficiently: the metadata for an empty track is at least 500 bytes, which is not negligible for some mobile scenarios; using "items" reduces the problem;
- it is not possible to add a stream after the presentation has been started, and in general it is difficult to do live streams (without DASH extensions);

Possible improvements:

- there is a lot of unused functionality carried over from QuickTime and kept for backward compatibility; as a consequence, the playback algorithm could be quite a lot simpler;

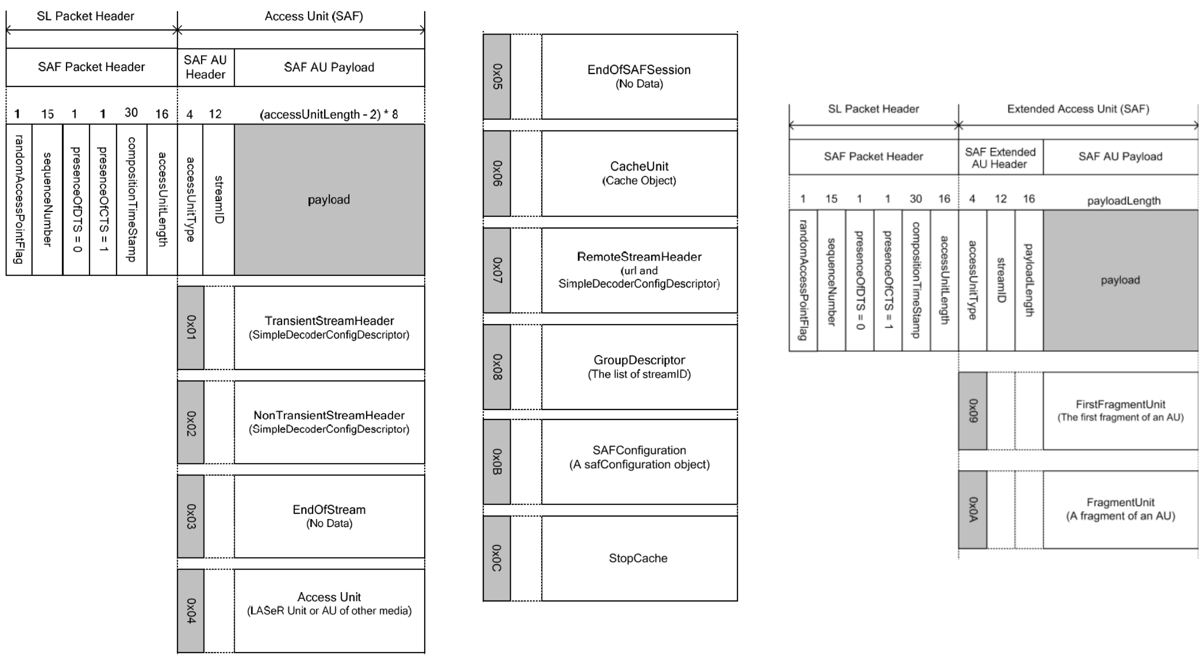
Further analysis is provided in the next section as part of the comparison with SAF.

3.4.2. Multimedia Streaming with SAF

The Simple Aggregation Format (SAF) defines the binary representation of a compound data stream composed of different data elementary streams (ES) such as LASeR scene description, video, audio, image, font, and metadata streams. Data from these various data elementary streams result in one SAF stream by multiplexing them for simple, efficient and synchronous delivery [47].

SAF is a specific configuration of the MPEG-4 Sync Layer [43]. The main reason was the ability to use the RTP payload for MPEG-4 ES defined in RFC3640. While initially thought a good idea, it became apparent later that this was not such a good idea to mix different streams inside what was defined, in MPEG-4 SL, as one elementary stream, with all the assumptions this carries.

While promoting SAF, we had to compare it with 3GP/MP4 file format, which 3GPP people wanted to use for streaming.



SAF Packet Architecture

3.4.2.1. SAF compared to 3GP/MP4

Three use cases are considered to study the benefits of SAF vs. 3GP/MP4 [68], and all results are based on byte-exact computations of relative sizes:

1. Simple download: the file of a multimedia presentation is downloaded with no need to play the content during the download of the file.

   ⇒ SAF is better than 3GP in this use case but the gain in header overhead is small.

2. Progressive download: the content may be played during its download. Two delivery scenarios are possible depending on the scene description:

   a. Without dynamic aggregation: The multimedia presentation to download is predictable and does not depend on interactions of the user. So downloading it consists in downloading its file with a Progressive-download profile. The structure of the file can use :

      i. either a basic movie (i.e. 'moov' atom) with interleaved media data,

   ⇒ SAF is relevant for avoiding significant initial latency.
   ⇒ 3GP files using 'moov' boxes are not relevant because of the large initial latency.

      ii. or a set of movie fragments (i.e. 'moof' atoms).

   ⇒ SAF is better than 3GP files using 'moof' boxes, but the difference is not significant.

   b. With dynamic aggregation: downloading the multimedia presentation by aggregating dynamically the right set of elementary streams at the right time is required because of interactive rich media browsing features. These features make the downloaded presentation unpredictable.

   ⇒ To avoid significant latency (i.e. ½ sec compared to the network round trip of ~3 secs) after unpredictable user requests, SAF is relevant for 64 kbps networks (e.g.

EDGE generation) and significantly relevant for 32 kps networks (e.g. GRPS generation).

   c. Of live streams: the processing of live streams when the size and number of future frames is unknown is of similar difficulty as dynamic aggregation.

⇒ 3GP files with 'moov' boxes cannot be used. Using 3GP files with 'moof' boxes will be subject to a trade-off between latency and 'moof' overhead: the shorter the 'moof' box period, the shorter the latency, but the heavier the overhead. SAF is definitely optimal in this case, and is better than any choice with 3GP files.

3. The limitations of the bytecode footprint: in the particular (but strategic in mobile phones industry) case of Java-based client terminals, we study if SAF can help to decrease the Java bytecode footprint of the client application.

   ⇒ The footprint of Java SAF parser is ~5 kbytes when the footprint of a Java 3GP File Format parser is ~35 kbytes (optimized implementations measured in similar conditions: compressed, obfuscated and stripped of symbols).

3.4.2.2. Analysis

This study concluded that SAF is technically better than 3GP file format on a set of use cases and for different measures. In particular, SAF allows the dynamic addition of a stream at any time, which is not possible with 3GP files. Yet, SAF can still not be used in 3GPP context, as non-technical arguments prevailed.
Going back to initial requirements, MP4/3GP was designed for authoring and storage, and SAF for streaming and transmission. So the result of the study is no surprise: both formats were designed this way. What is a surprise is that MP4/3GP is used in a context it was not designed for, and a push to use SAF, a format designed for that context was not successful.

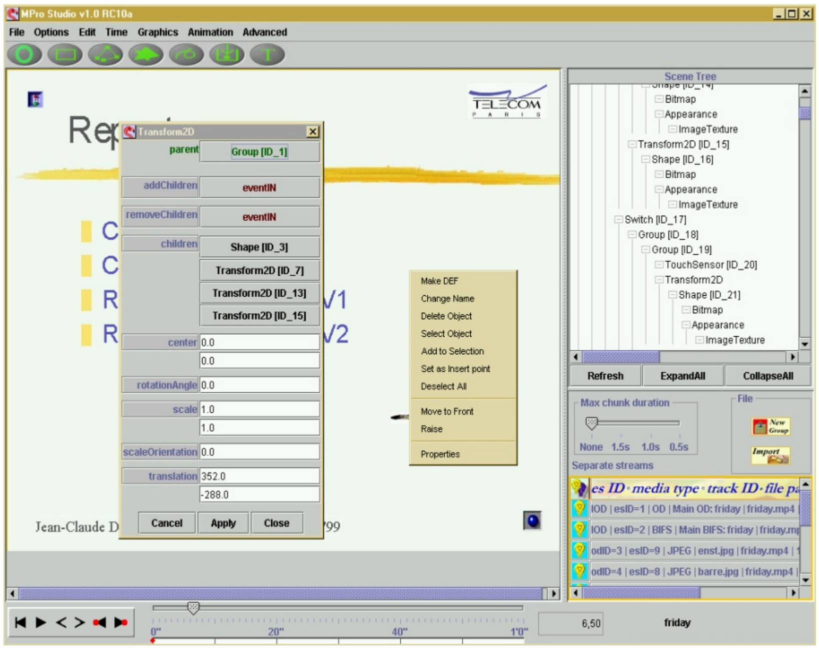
3.5. Authoring

Very early in the design of a standard, authoring becomes an issue: there is a need for conformance sequences and for demo content. The availability of authoring tools helps a lot with the dissemination of the standard, the ease of content production being a quality criterion for any standard. We started with a rather straight-forward tool/framework named MPRO for experts, then worked on a tool for the general public named Harmonia, based on templates, within a commercial project. We later tried another form of templates with B4 with much less resources. And to co-design service logic and content, we designed a variant of JSPs with a twist, but again for experts.

3.5.1. GUI to content

This work was part of the PhDs of Frédéric Bouilhaguet [1] and Souhila Boughoufalah [2], within a collaborative project called MPEGPro. The collaboration partners were CSELT (now called TI Labs), CNET (now called Orange Labs) and ENST (now called Telecom ParisTech) [28] [30] [32] [31].
Souhila started her thesis with a taxonomy of scene formats, then a taxonomy of authoring tools, with different possible paradigms and/or views on content: WYSIWYG

spatial, timeline-based, frame-based, structure, script-based, contraint-based and template-based. She then presented MPEG-4 Systems, the native format of the proposed tool.



The tool itself is called MPRO and is a detailed-level authoring tool for BIFS experts. It has a structure view, a spatial WYSIWYG view, a timeline as well as various helper views (media, property sheets). She described the architecture and design of MPRO. She finished with an analysis of the next most needed feature: changing the default synchronization of media. MPRO was implemented in Java, using the Swing graphics library.
Frédéric contributed the media management and the MP4 file management, including the media interleaving.

3.5.1.1. Analysis

Like the tools of the time, MPRO is multi-view and includes a player. Compared to Macromedia Director, one big difference is the lack of scripting support: on one hand, this could have been added later, and on the other hand, we were encouraging the use of Conditional to achieve more efficient scenes. Compared to constraint-based systems, such as Madeus [78], the timing approach in MPRO is explicit, and relies on the MPEG timing model, which was favoured in professional media environments (TV, movies). Compared to systems editing a different language than the one produced, such as Director or [79], we needed to be able to import content produced elsewhere and edit it natively, and also had a mandate to be able to use all of the standard, which forced us to adopt the BIFS model inside the tool.
As a first project on authoring GUIs, it also forced us to realize that its scope was too large for the team, and we did not have the resources to keep it alive.
The project was very successful with respect to its initial goal: a 2D BIFS authoring tool for experts including a WYSIWYG view. What may be questioned is the validity of the initial goal: is it enough to just wrap a GUI around such a vast and low-level format as BIFS?
The next project is the result of these two realizations:

- all of BIFS2D is too big, target a subset;
- simplify complex BIFS functionality to make it accessible to a wider audience.

3.5.2. GUI-assisted templates

This work was also part of the PhD of Souhila Boughoufalah [2], and funded by an industrial project called Concerto with TDK.
The intent was to offer a set of templates to users. The templates needed to be extremely simple to use. The templates should be compiled to MPEG-4 BIFS. The templates should be editable by a professional. The templates library should start small and get richer and richer.
The simple use mandated an authoring tool for the manipulation/customization of templates. The context of the project imposed other technology choices:
- PC and Windows as a platform to reach the Japanese market;
- MPEG-4 as a multimedia technology, in particular for the use of non-rectangular video (obsoleted since then);
- Java for the reuse of MPro assets, and Microsoft J++ to allow the cooperation with other partners.

Harmonia was a simple graphical, “drag & drop” editor [29]. An author can choose a predefined template and insert objects like movie, slideshow... and replace an object in this template. Objects can be configured by modifying their attributes (size, color, ...). Complex objects like slideshow need to be configured in separate editors to set all the slides with the transitions related to each slide. The author can play his/her content at any time with an integrated player or publish the MP4 file with the default scenario for http context.

3.5.2.1. Elements Composing a Harmonia Scene

To understand the elements composing a Harmonia scene, we give a few definitions.
A BIFS object is an MPEG-4 scene element.
A Harmonia object is an object specific to our authoring environment including “java” objects for customization purposes. These objects have a meaning only in the Harmonia editing environment.
A Building Block (BB) is the association of BIFS object(s) and Harmonia object(s). The building blocks can be of different types : (still) image, video, audio, text, slideshow, button. BBs have generic properties which can be on or off depending on the template context: movable, resizable, deletable and replaceable, which comes with a list of allowable replacement types.
BBs have variants. When a user creates an object by drag and drop, a default BB for the selected type is added. For example, the “default” image BB resizes the image to the BB’s size: there are variants of the image BB that do not resize the media, or clip it, or show scrollbars… This is specially relevant for buttons. The tool comes with a small set of variants, and new variants can be added in a plug-in fashion.
Harmonia provides the author with a notion of behavior which can be triggered by interaction or time. An action is defined by four parameters: source object, target object, type of event e.g. button click, time … and requested behavior: e.g. start media, show object...
A Template is an entity including several building blocks. Some templates are empty and the user can insert any building block at any location and set the needed behavior. Other

ones are more specific because more complex. For example, the mosaic template is a program grid comparable to what most satellite operators offer to their customers. In this case, the author has only to customize the A/V content of the grid elements.

3.5.2.2. Structure of a Building Block

The persistent part of a BB is a set of a BIFS scene, a Harmonia script and a set of Java classes. Table 1 shows a sample with a BIFS tree, the script and a short description of what functionality each node helps creating. The Java code has two functions: (1) parsing the script and establishing the link with the in-memory representation of the BIFS ; (2) implement any BB-specific customization.
The BIFS structure is a generic canvas and depending on the Harmonia script, not all the options are open to the user. For example, on the 4th line of the script, the "true" means that the BB is movable.
The last part of the Harmonia script, about EventIn and EventOut, defines the behavior of the BB. The EventIn spec concerns the actions that this BB can execute, in this case Show or Hide: the first line points to the Conditional implementing the Show, the second line points to the Conditional implementing the Hide and the last line points to the Switch node in order to set the initial visibility state. The EventOut spec concerns the events that this BB can generate: boolean events isActive and isOver from the TouchSensor.

BIFS Component of the BB	Description	Harmonia Script
		Image "Image" {
Anchor [ID_55]	Creates the hyperlink capability	top, url "ID_55"
PlaneSensor2D [ID_56]	Creates the draggable capability	enabled "ID_56"
Transform2D [ID_1]	Position of the BB (incl. dragging)	translation "ID_1" true
TouchSensor [ID_57]	Allows actions and internal links	enabled "ID_57"
Switch [ID_2]	Creates the Show/Hide capability	
Transform2D		
Shape		
Rectangle [ID_3]	Background of the BB	size "ID_3" false
Appearance		
Material2D [ID_4]	Color and transparency of bg	emissiveColor ,transparency "ID_4"
LineProperties [ID_5]	Border of the bg	lineColor, width, lineStyle "ID_5"
Shape		
Bitmap		
Appearance		
ImageTexture [ID_6]	Image	url "ID_6"
Conditional [ID_7]	Implements the Show action	EventIn {
Conditional [ID_8]	Implements the Hide action	activate 223 "ID_7"
		reverseActivate 224 "ID_8"
		whichChoice 225 "ID_2" }
		EventOut {
		isActive 226 "ID_57"
		isOver 222 "ID_57" } }

Thanks to the generic architecture of Harmonia, building blocks of existing types or of new types can be easily added, allowing customization of Harmonia for specific application areas and catering for particular user group needs. For the specific BBs Slide-show and Movie, the customization is done in different helper tools that have their own internal formats. Specific APIs enable the communication between the different environments and Harmonia performs the translation from these formats into MPEG-4 format.

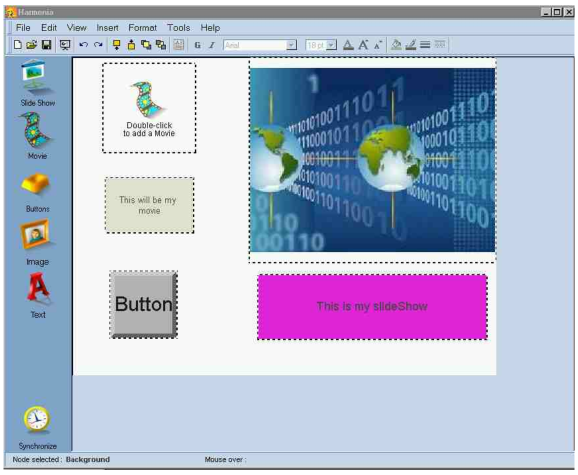
3.5.2.3. Harmonia Architecture

Harmonia is a layer on top of MPRO providing what was referred to as the Template Manager in [31] [32].

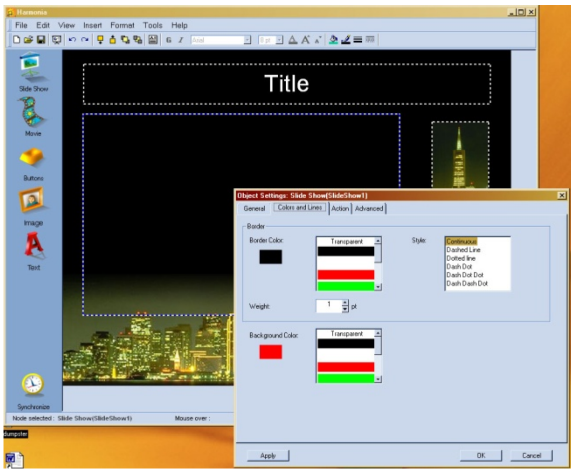
The MPRO part is responsible for the management of an MPEG-4 scene from encoding/decoding – composition - rendering – to MP4 management. Harmonia is responsible for the management of a Harmonia scene and Harmonia objects with the customization information and the conversion into BIFS of some Harmonia objects once customized.

Harmonia uses MP4 files to store templates, building blocks and partially customized scenes. These MP4 files contain a standard atom —'skip', designed to encapsulate opaque information— to store all non-standard information. Customization information is stored in this atom, as well as building block information. In final scenes, all customization information has been removed, so final scenes are not editable any more within Harmonia.

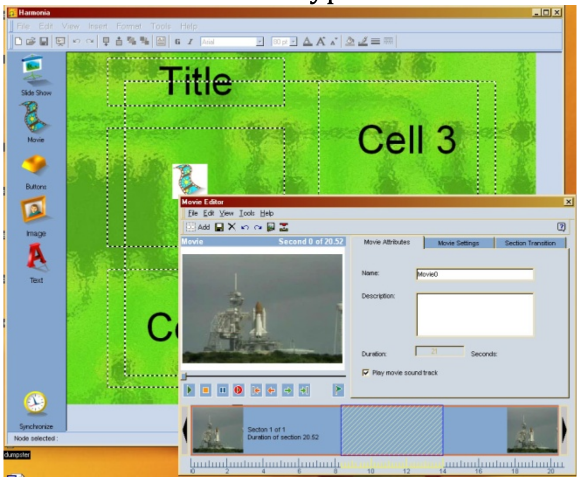
The usage scenario for final scenes is "http pseudo-streaming". This consists in organizing the MP4 file in such a manner that the information needed to start playing is at the beginning of the file, and the media samples are interleaved. Thus, the terminal can start playing the scene before the end of the downloading. There is no guarantee about quality of service.



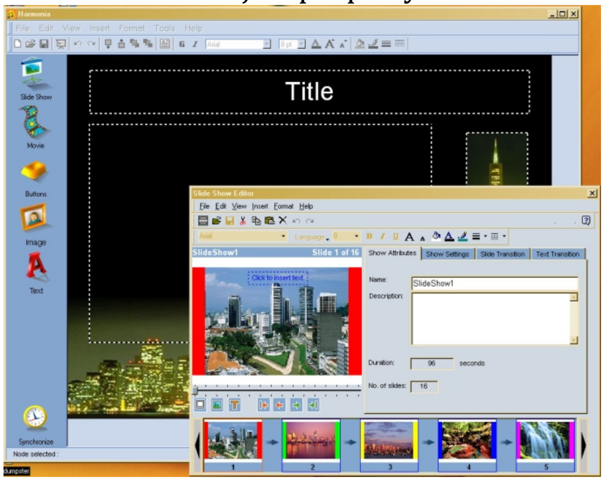
the different types of BB



with an object property sheet



with Movie Editor



with Slide Show Editor

3.5.2.4. Analysis

Today, such a tool would be very useful, but support for some of the technologies does not exist anymore, and in particular, MPEG-4 video with non-rectangular shape: the corresponding profile was never successful, this feature is not implemented in any of the available decoders, and later video standards do not have this feature.
The following adaptations would make Harmonia very useful today:

- simplification of the technology set, e.g. dropping the non-rectangular video.
- rewriting of the BB/template technology in a less fragile environment, e.g. not having 2 scripts that need to be reconnected each time by running some code. Using an XML-based format would make it possible to add template metadata in a different namespace interspersed with the scene description.
- extension of the interactivity to scripting, using object-oriented design to define a library of extensible behaviours, à la Flash.

3.5.3. Simple text to content

The initial problem is that most formats are created too low-level for the author, as a compromise with the capacities of terminal rendering (specially mobile terminals). In particular, in BIFS, there are many intermediate objects such as Shape, Geometry, Appearance, Interpolators, Routes, which are much lower level than the usual objects manipulated. The initial B4 project was developed within the ISIS European project, and applied to BIFS/XMT-A: the target was to allow the easy creation of complex interactive content.
The principle is to create a set of higher-level objects in a language close to the presentation language. The presentation language is XML-based, and can be XMT, SVG or LASeR. The author edits a scene with e.g. a complex menu system described with menus, submenus and menu items decorated with the usual objects of the host language, and connected with actions described in the host language. However, all the mechanics of the higher level objects, here the menu, are hidden from the author.
A translation process transforms the authored scene tree into a much more complex, and usually difficult to edit, scene tree constituted only from objects of the host language.
[72] describes another approach to extend other languages, with an XSL translation, but with quite a different context: instead of working incrementally, NCL2 is based on an earlier framework, and intends to port NCM features to e.g. SMIL, such as the NCM links, which are explicitly propagated triggers and associated actions.
B4 defines high-level objects called building blocks. Building blocks implement a set of features in BIFS, hiding the complexity of BIFS from the author. Building blocks can be nested, thus combining features from different building blocks.
For example, there is a building block for an image, a building block for bevel borders, a building block for hyperlinks and a building block for an object you can show or hide.

Here are the objects and their combination:

```
<image stream=”toto.jpg”/>
```

is the image alone.

```
<link url=”myUrl.html”>
    <image stream=”toto.jpg”/>
</link>
```

is the image with the hyperlink.

```
<bevel bevel="8 8" color="0 0 0" dark="1 0 0" light="0 1 0">
      <link url=”myUrl.html”>
         <image stream=”toto.jpg”/>
      </link>
</bevel>
```

is the image with a link and bevel and

```
<showhide name=”beveledimage”>
      <bevel bevel="8 8" color="0 0 0" dark="1 0 0" light="0 1 0">
         <link url=”myUrl.html”>
               <image stream=”toto.jpg”/>
         </link>
      </bevel>
</showhide>
```

is the final object with show/hide capability.

The inner image automatically translates to the Shape / Geometry / Bitmap / Appearance / ImageTexture hierarchy, and defines the Object Descriptor and ES Descriptor needed to point to the image stream. The size of the expanded tree is often 10 to 20 times bigger than the initial tree edited by the author.
The link element is much simpler as it expands to a simple anchor element.
The bevel element adds decoration, here a set of polygons, which automatically adapt to the size of the content.
The showhide element defines interaction capabilities, implemented as two Conditionals and a Switch around the content: one Conditional contains code to turn off the rendering of the Switch content, and the other contains code to turn on the rendering of the Switch content.

The underlying technology in B4 is XSL Transformations [58]. Three passes of XSL are applied to the initial scene tree. One pass defines the Object Descriptors. Another pass defines the initial scene tree. The last pass defines scene updates, if needed.

There are groups of coherent building blocks, which interact: some building blocks define actions and other building blocks call actions. For example, the showhide block defines the capability of switching some elements off and on, and any of the buttons building blocks may trigger an existing action. In BIFS/XMT-A, I have chosen to implement the connection between the two by ID: showhide defines two Conditionals called “show_nameOfBlock” and “hide_nameOfBlock”. A button may have action=”show” and target=”nameOfBlock”.
This naming strategy will work in any scene description language with textual IDs, and an object similar to Conditional, which can be triggered by its ID. In the absence of a Conditional equivalent, script functions could be used.
Otherwise, building blocks are completely unrelated. Building block sets designed by different people can be used concurrently, and even on the same elements: one block provides an action, another provides a decoration, yet another provides layout, …

The advantage of B4 lies in this flexibility. Whereas XMT-O constructs are fixed and add a rather complex layer on top of BIFS, B4 blocks use plain BIFS, but simplify the work of the designer.

The latest version of B4 includes:

- <video>, <image>, <background> create the element together with the object descriptor,
- <pages> creates a set of overlaid pages (à la HyperCard, or Java CardLayout),
- <inline> invokes another scene as a sub-element of this scene,
- <bevel> creates a bevel-like decoration around a rectangular zone, typically to help create Motif-like raised or sunk rectangular buttons,
- <group> is a general container, allowing the insertion of multiple objects where one is expected,
- <link> is the equivalent of HTML href,
- <hotspot> creates a mouse-over-sensitive region,
- <button> creates a simple button and <mbutton> creates a button with multiple actions,
- <showhide> is a wrapper which can show or hide its content,
- <activeButton> is a button which has multiple visual states (up-down or up-over-down),
- <radio> creates a set of radio buttons,
- <hss> creates a slide show, with slides, animated slide sub-elements, looping, navigation arrows.
- <hsummary> creates a slide navigator for an <hss>
- <slider> creates a scrollbar or slider
- <animate> maps slider output onto various possible fields
- <tooltip> creates a tooltip
- <grid> creates a grid-like replication of a single object; a similar construct could be designed to generate a new object for each node of the grid;
- <maxim> creates a object with zoomin-zoomout behaviour, <maxim1> is the self-zoom version
- <popper> manages the drawing order of its components
- <nervous> is a container that moves its content around in a “nervous” manner.

3.5.3.1. Reuse of B4 in other contexts

B4 served as a basis for a similar set of macros in LASeR. Of particular interest were the more complex macros to:

- define a tree with expandable nodes, each node composed of an optional icon, a text and an action upon clicking on the text.
- define a column of items which focused item always stays in the middle of the screen, and item validation goes to an item-specific page.

The principle behind the design of these macros was:

- per item, there is a number of Conditionals, one per action: for the column of items, actions were “next-up”, “next-down”, “go-to-item-page”, “return-to-item-column”;
- the state information is “stored” in the actions connected to each used keystroke: state-changing conditionals change the keystroke actions.

In BIFS, the connexion is done by a ROUTE, which can be deleted and recreated for a different connexion.

In SVG, with no Conditional, the whole mechanism would be implemented with script functions and event listeners.

In LASeR, the connexion can be done by setting all Conditionals with “begin=’accessKey(key)’”, only one of the Conditionals being enabled at a time.

3.5.3.2. Analysis

The pros of this technique are:

- this technique is applicable to many situations, and enriches the existing set of objects proposed by the base language.
- this system is incremental, and new macros can be added at any time; ideally, anyone should be able to add his own macros.
- it should be possible to wrap a GUI around these macros.
- it is very easy to provide this tool as a web service, or as a command line tool, to insert it into an automated chain.

The cons of this technique are:

- designing new macros requires a high level of a rare competence (XSL).
- mixing macros with native objects of the language requires a relatively good knowledge of the native language.
- at this time, there is no GUI wrapper around these macros.

My template approach is one to increase the level of abstraction of objects proposed to the author in the scene format he has to use; it solves quite a different problem from other approaches such as LimSee3 [80] where the templates are used to propose a simpler view on complex features of SMIL.

3.5.4. Program to content

In Streamezzo, we developed a technique derived from Java Server Pages, with a twist. The name of the technique is RSP for Rich media Server Pages, but it applies to any XML-based strongly typed description format.

3.5.4.1. Richmedia Server Pages

The whole system is based upon equivalence between declarative XML for a scene tree, and a program whose execution would generate the same declarative XML. Any piece of XML can be easily translated into a program which:

- creates a program object for every XML element found;
- adds a property to the program object for every XML attribute added to the XML element;
- adds a text property to the program object for every text content in an XML element;
- adds to the list of children of the parent program object any program object created for an XML element which is a child of the parent XML element.

In order to be able to do this translation, we assume the availability of a library of functions to manipulate a set of program objects implementing the set of XML elements of the scene description. The host language needs to have strong typing, in order to detect as many errors as possible upon compilation. The library should have one type of object per XML element type, and implement as many methods as necessary to ensure "correct by construction" content. The library typically implements methods to serialize the content back to XML, to serialize the content into a compressed form if any, and any sort of checking relevant to the description format.

The above process is very powerful, as it may use programming language type checking in order to check that:

- a property exists for an certain type of object;
- the value of the property is of the right type and/or in the right range;

- the type of the child object is compatible with the type of the parent object.

For a program derived from a declarative XML, this is overkill. There are technologies, such as XML schema, allowing the validation of a piece of declarative XML without resorting to translating it to a program and then compiling it.

However, the above technique is extremely useful when designing complex interactive content, some of which is designed as declarative XML, and the rest is designed as a set of program pieces, usually interspersed with the XML, à la Java Server Pages.

JSPs compile to a program manipulating strings. There is no intelligent structure left in the compiled program. In a RSP, the XML structure is conserved in the program. This translates into differences at compile time and differences during execution.

At compile time in a JSP, the declarative parts compile to a string constant. Program pieces are sets of string computations and then string concatenations. There is no checking of the integrity of the generated text as XML.

At compile time in a RSP, each XML element is translated into a typed object of the library. If there is an instruction to set a property that does not exist, an error with precise documentation of the problem is raised.

During execution of a JSP, the program concatenates strings, which are sent to the terminal, parsed there, and errors are raised in the browser on the client terminal, with no trace of a context.

During execution of a RSP, typed objects are manipulated. If compilation did not catch an error, e.g. because the type of a variable was too generic, a type error is raised during execution if the object resolves to an incorrect type: the error is raised on the server, and adequate documentation of the problem can be logged.

The program pieces interspersed with the declarative XML can also be very rich, much richer than anything possible in JSPs:

- there can be a keyword for the current object, and any use of the current object will be type-checked: any wrong assumption on the type of the current object will be caught at compile time;
- the same is true with the enclosing objects (or “father” and ascendants);
- and many properties of the current spot where program code is being inserted: current transformation, lists of existing actions or objects, etc…

There is one limitation to RSPs that JSPs do not have: because JSPs are fully unstructured, program pieces may construct anything, even invalid XML fragments, provided that after concatenation with constant parts of the page, the result is valid. RSPs program pieces can only yield attribute values, attributes, lists of attributes, elements or element lists, i.e. valid XML atoms, not half a tag name (it is possible to compute a complete tag name, though).

We believe such use of JSPs is more of a hack than a good procedure, and should be discouraged anyway.

3.5.4.2. Analysis

3.5.4.2.1. Advantages of RSPs

RSPs use a strongly typed language (Java) in order to catch as many errors as possible at compile time. RSPs document execution errors with a full scene-related context to enable intelligent debugging. The RSP environment is much richer than JSP environment: it offers many predefined objects, contextual properties and methods increasing the power of the code inserted in the RSPs. RSPs are executed on the server and minimize the performance requirement on terminals. Terminals without scripting engine can be used.

3.5.4.2.2. Comparison with AJAX

RSPs can be compared to the server-side code of AJAX services.
AJAX services typically require:

- a script executing on the terminal, sending a complex request to the server;
- a server, typically PHP code or servlet, computes an answer to the request in the form of a piece of XML;
- the terminal script needs to parse and process the XML reply, possibly creating dynamically on the terminal fragments of scenes.

RSP services typically require:

- a content stub on the client, whose role is to send a request to the server from the user context, thus providing the service with a full HTTP request context;
- a servlet engine with the RSP servlet installed; the servlet receives the request and constructs the content appropriate to the user;
- the content is received and rendered as is by the terminal; there is no processing on the terminal, thus making RSP the technology of choice on lower-end terminals.

The current trend of using the “cloud” for processing is also very compatible with the RSP technology.

3.6. Adaptation

Content reuse became a problem with the fragmentation of the market in many dimensions: fragmentation of the mobile device market, separation into Internet and mobile networks, multiplication of formats… We had two activities in the area of adaptation: one within EU projects related to MPEG-21 and an industrial contract with Orange Labs, and another of format translations for academic and commercial projects.

3.6.1. Adaptation of content to rendering context

This work was part of Mariam Kimiaei Asadi’s PhD [4]. The context was both MPEG-21 standardisation and two European projects called ISIS and DANAE [19]. The work is organized around two poles: resource conversion and scene adaptation. Resource conversion is a basic tool that is necessary for scene adaptation. It was also a tool that was quite incomplete in MPEG-21 Digital Item Adaptation. Scene adaptation is a quite wide field, but we chose semantic scene adaptation, in the sense of adaptation based on semantic information provided by the author as part of the MPEG-21 description of the scene.

3.6.1.1. Resource Conversion

After a state of the art in single media adaptation, Mariam introduced our main contribution to DIA: modality conversion, or transmoding [16]. The context of the project was to use on-demand, hinted conversion. In this type of adaptation, the media is considered alone, i.e. as a mono media, without any multimedia structured presentation or independently of its native multimedia composition. She defined description tools extending the MPEG-21 DIA schema, for description of hints on different media adaptations, also called resource conversions, and their corresponding parameters. She also implemented a media adaptation engine that, based on these direct hints as well as usage context constraints, applies the most appropriate form of media adaptation with optimal values of adaptation parameters, in order to provide the end-user with the best quality of experience. All this was done in parallels with contributions to MPEG-21 DIA.

In / Out		Audio			Image	Video	Graphics		Text
		Speech	Audio 2D	Audio 3D			Graphics2D	Graphics3D	
Audio	Speech	▨	?	?	▒	▒	▒	▒	?
	Audio 2D	?	▨	?	▒	▒	▒	▒	▒
	Audio 3D	?	?	▨	▒	▒	▒	▒	▒
Image		*	▒	▒	▨	√	√	√	√
Video		*	▒	▒	▒	▨	√	√	√
Graphics	Graphics 2D	*	▒	▒	▒	√	▨	√	√
	Graphics 3D	*	▒	▒	▒	▒	▒	▨	?
Text		?	▒	▒	▒	▒	▒	▒	▨

In the table above, hashed means no conversion, gray means the conversion is not doable in general, a check mark means it is part of our study, "*" means there are multiple ways/paths and "?" means the transmoding should be possible but was not studied in detail.

3.6.1.2. Semantic Scene Adaptation

In semantic adaptation of structured multimedia documents, she addressed the question of adaptation based on temporal, spatial and semantic relationships between media objects. When adapting a multimedia presentation, in order to preserve the consistency and meaningfulness of the adapted scene, the adaptation process needs to have access to the semantic information of the presentation. She defined a set of descriptors for the expression of semantic information within composed multimedia content. These descriptors contain information provided by the author of the multimedia scene, or any other entity of the multimedia delivery chain, that helps the adaptation engine decide on the optimal type and nature of the adaptations that are to be applied to the multimedia document. The information included in these descriptors cover the independent semantic information of each media object of the scene, the semantic dependencies between media objects of the scene, and the semantic preferences on scene fragmentation. This information is then used in the heuristic described in Figure 8.

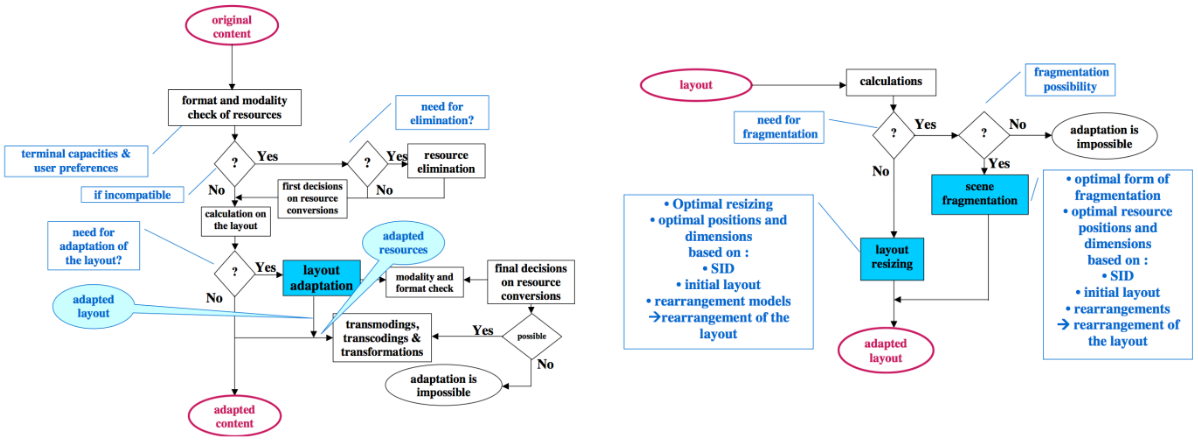

original content
format and modality check of resources
terminal capacities & user preferences
if incompatible
need for elimination?
?
Yes
No
resource elimination
first decisions on resource conversions
calculation on the layout
need for adaptation of the layout?
adapted resources
layout adaptation
modality and format check
final decisions on resource conversions
adapted layout
transmodings, transcodings & transformations
possible
adaptation is impossible
adapted content
layout
calculations
fragmentation possibility
need for fragmentation
adaptation is impossible
scene fragmentation
• Optimal resizing
• optimal positions and dimensions based on :
• SID
• initial layout
• rearrangement models
→rearrangement of the layout
layout resizing
• optimal form of fragmentation
• optimal resource positions and dimensions based on :
• SID
• initial layout
• rearrangements
→ rearrangement of the layout
adapted layout

Figure 8 : Scene adaptation heuristic (left) and layout adaptation heuristic (right)

She then went on to implement a prototype using a subset of SMIL2 as a scene language, for its simplicity and closeness to the semantic groupings of media objects. The descriptors capture useful information and the heuristics are efficient on the types of documents that were of interest [17]. The prototype addresses the following adaptations:

- adaptation of media to the terminal capabilities, with selection of alternatives, elimination, transcoding or transmoding as feasible;
- optional scene fragmentation, with and generation of navigation among the fragments;
- layout adaptation.

3.6.1.3. Analysis

The functional annotations in [81] are a more formalized version of the above semantic relations. We have used heuristics here, in the spirit of a rule-based expert system, but others, including evolutions of [78], have used constraint-based systems for similar purposes.
The main drawback of this proposal is the need for the author to capture the semantic relationships and information. The proposed heuristics may be further developed or be adapted to different situations or types of documents, and it seems obvious that adaptation that does not respect the semantic information will simply not work as related objects could be separated. When an integrated (GUI) authoring tool will be used, then careful addition, within the workflow, of the capture of semantic information should reduce the burden on the author substantially. In other situations ("manual" design in text, programming/automatic generation), a format extension might reduce that burden.

3.6.2. Format translation

3.6.2.1. SVG to BIFS

As part of the state of the art work before the BIFS extension named Advanced Text and Graphics, Cyril Concolato started to work on SVG to BIFS translation. The goal was to study the "compatibility" of SVG and BIFS, their relative power of expression, and allow the reuse of SVG content, in particular icons.

Since SVG and XMT are both XML-based, the tool was first implemented as a number of passes of XSLT:

- clean up of namespaces and removal of authoring tool metadata;
- removing CSS-like style compound attributes, replacing them with single attributes;
- clean up of polyline/polygon coordinate lists;
- converting units into pixels;
- generating Object Descriptors/Elementary Stream Descriptors for media;
- translating as much of animation as possible;
- translating graphics and text primitives.

The result was actually a very limited SVGT1.1 to XMT translator, dealing only with geometries and simple animations.
The first implementation was pure XSLT. Later versions were optimized with bits of Java:

- to speed up the process, a Java tokenizer was used to accelerate path translation;
- as features were added, bits of Java were used to do some computations (square root, degree to radian conversion, transform matrix manipulations…).

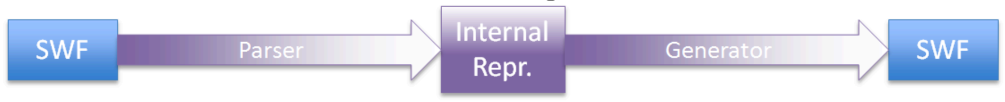
Some of the code of this project was consolidated and extended into the B4 tool.

3.6.2.2. Flash to BIFS/SVG/LASeR

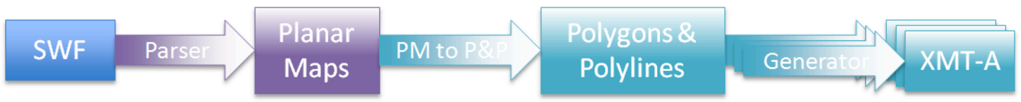
After Cyril's work on SVG, and a study of the structure of Flash and its encoding, we had the idea of trying to find out how much of Flash could actually be translated to other formats. If Flash was mostly translatable, then it proved that it is "just another scene description format". The big hurdle was the translation of the planar maps.
Another goal was to be able to reuse some of the wealth of free Flash content floating around the Internet. Yet another goal was to complement our suit of tools for resource conversion.

Software

Frédéric Blanc was asked to write the software, with help from Cyril Concolato and myself. The software was written in Java. We reused an open source package that parsed SWF and created an XML "equivalent" of the SWF. We added a module translating from planar maps to polygons and polylines, and then a series of different generators for our formats of interest: XMT-A, SVG and LASeR (CD version). The architecture of original software and translator software is shown in Figure 9 below.



Architecture of the original open source package



Architecture of our Translator

Figure 9 – Architecture of original software and Flash translator

Dealing with Planar Maps

The input of the "planar maps polygonizer" is a set of quadratic curves, specified by two extremas and one control point, and for each of these a left color and a right color are specified. It can be safely assumed that the curves are non-intersecting, and that extremities match. The (non optimal) process is:

1. for each curve extremity, find all matches among other curve extremities and create a graph whose nodes are the extremities and whose arcs are the curves;
2. then for each unaccounted curve, walk the graph to find its left shape and its right shape; during this process of creating shapes, all curves building a shape are marked as accounted for on the shape's side;
3. the end is when all curves are accounted for on both sides.

Sorting extremities in X and then Y allows the optimisation of phase 1 in terms of computing time. Looking for unaccounted curves sequentially in the list of curves, skipping curves already accounted for on both sides, optimizes the second phase.
There are edge cases to take care of. And there are Flash quirks to detect and eliminate.
Shapes are then sequences of quadratic curves that may need to be translated:

- into straight polygons for LASeR CD or some BIFS profiles;
- into cubic curves for other BIFS profiles.

Finally, the shapes need to be generated in the appropriate format.

Dealing with Flash Updates

Flash uses a simple update mechanism, and is frame-based. The main rendering-related commands are:

- add a reference to an object at depth D;
- remove a reference to an object at depth D;
- change a property.

It is quite simple to maintain in BIFS, SVG, or LASeR, a ordered list of empty groups represented the possible depths, and add/remove object references in these groups.
As a result, the mapping of these updates to BIFS/XMT-A and LASeR are relatively straightforward. Generating SVG requires:

- either use SMIL animation to set the display property of elements *on* upon "insertion", and *off* upon "removal";
- or use scripting to add or remove elements from the scene tree.

Dealing with MovieClips

The MovieClip mechanism of Flash defines a rendering loop, a series of frames containing insertions, deletions and property changes, which can be called as many times as needed. In early video games, this was called a "sprite". Translating MovieClips is possible assuming that the loop is finite, so that it can be unrolled. The unrolling penalty is usually small and acceptable:

- each MovieClip frame is translated into a Conditional adding and removing object references and changing properties;
- an unrolled frame just consists in triggering the appropriate Conditional.

If scripting is available, infinite loops are easy to generate, and unrolling is not necessary.

Dealing with Flash Scripts, Morphs, etc

This is the main limitation of the translator: there is no way we could think of to translate Flash scripts. Morphing and filter effects are also allowed in SWF content, and most of these cannot also be translated. Some filter effects could be translated to SVG 1.1 Full, but our target has always been SVG Tiny 1.2, which does not including filter effects. However, morphs and filters are used less often than scripting. Scripts are used in most Flash content, if only to deal with progressing loading. And there lies the most serious obstacle against reusing Flash content massively.

3.6.2.3. PowerPoint to BIFS/SVG/LASeR

In Microsoft Office, there is a "interactive multimedia authoring tool" used by many: Powerpoint (PPT). There is also a scripting language, Visual Basic, with a complete API to manipulate PowerPoint documents.
Even though this format is rather restrictive, it represents a meaningful item in the interactive multimedia constellation, even if only by the number of its users. As part of our work on format translation, it felt important to at least try to create a translator from PPT to BIFS and other more open formats.
As it soon appeared, it was relatively easy to export slides, text with properties, images and other slide components, to BIFS. So we had a partial translation quite soon. What stopped us were:

- the proliferation of different ways to do the same thing,
- the lack of APIs for some properties: part of the format was simply not accessible in any way but through the PPT GUI,
- and a lack of overall logic, which made the lack of documentation a killer.

3.6.2.4. Analysis

This work was very useful in many ways:

- it was the basis of the BIFS extension "Advanced Text and Graphics", an amendment which brought BIFS at the same level of descriptive power than SVG and Flash in 2D;
- content reuse: in Streamezzo, the ability to repurpose SVG icons or Flash short sequences into our format was a great asset. In the team here, availability of content translated from Flash made parts of other projects possible, e.g. the cartoons study in the QoS section.
- it also gave us a thorough understanding of each format, its pros and cons, and its quirks. It helped validate many of our assertions.
- many translation techniques were reused in other projects, such as B4.

The translators are also still usable for later studies.

3.7. Rendering

At first, rendering was just a tool, "development" and not research. As Gianluca and Jean demonstrated, study of rendering is needed: to solve the problems of rendering new types of content, such as streamed scenes (cartoons); to provide a framework for studying all formats and to serve as a base for projects based on interactive graphics.

3.7.1. Rendering for QoS

This work was part of Gianluca DiCagno's PhD [3]. It is our only work on quality of service. It is often called quality of experience rather than quality of service, as it focuses on the user experience [18].

During the implementation of an MPEG-4 BIFS player within a complete MPEG-4 Systems/Audio/Video terminal, Gianluca had to tackle multiple problems. As an engineer, he tried to find existing solutions, found none, and started his research.
He found that there is too much variability in too many dimensions, despite the widespread use of profiles in MPEG, which reduces the problem significantly by making single stream rendering more predictable. The variability is even worse in decoding time for scenes, specially with 3D. There is a need for predictors.

3.7.1.1. Quality Modelling

He started with setting up a quality model for multimedia rendering, and restricted his work to best effort systems: indeed, as soon as you can reserve resources, most of the problems disappear, and more and more devices today run on a best-effort, general-purpose OS. He also restricted his model to multiple audio and video streams first and aimed for a user-level QoS model. He designed descriptors for semantic importance of streams (not unlike in the previous section), and dealt with sensitivity to degradation, which is larger for audio. He implemented an extension of the MPEG-4 terminal architecture to include measure and control, both global and local, with very impressive improvements in QoS.

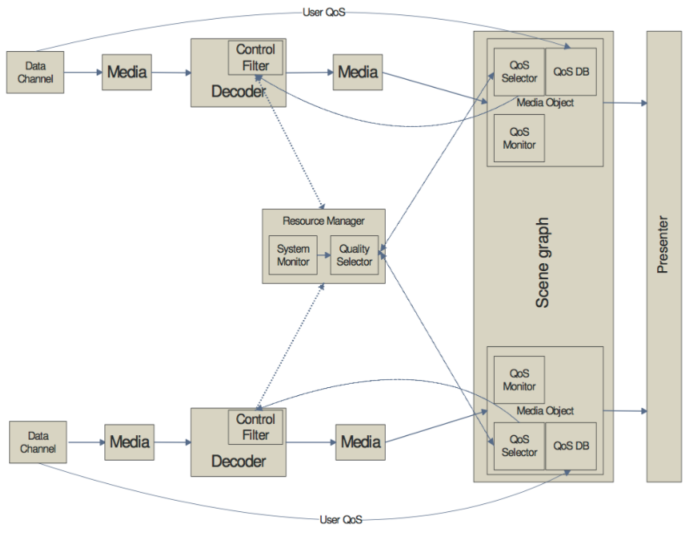


Figure 10: QoS extensions of MPEG-4 terminal

3.7.1.2. Addition of Scene Streams

There is considerable experience in the decoding and rendering of audio and video streams. It is not the case for scene streams, which are quite unpredictable, compared to even to video streams including different types of frames.
Gianluca goes at length into variability causes in the processing of scene streams: decoding and tree insertion, tree traversal, painting algorithms and their optimisations, anti-aliasing, interfacing strategies with the graphic hardware… For example, it is faster to write a video directly into video memory, but it is faster to draw graphics into main memory and then transfer the drawing into video memory. He implemented scenes into his previous model and proved the validity of the model on this new type of stream, but still found problems, e.g. a degradation of video quality in the presence of a scene stream of lesser frame rate.

3.7.1.3. Quasi-Continuous Scene Streams: Cartoons

Cartoons constitute an extreme example of variable scene stream. Investigation of cartoons rendering showed extreme quality challenges on normal PCs of that period, let alone slower devices. Gianluca designed and implemented predictors for decoding time to improve the quality of rendering for content including cartoons and in particular to react constructively to changes in CPU availability (when an application starts in parallel on the device). The best predictor is based on the relative size of next access unit compared to the size and decoding time of the previous four access units.

3.7.1.4. Analysis

This work just scratched the surface of multimedia QoS, even if he did propose a good model, and raised and solved difficult questions around scene rendering. His implementation work (MPEG-4 Systems and BIFS) is available as part of the MPEG-4 Systems reference software, but this standard is not used very much. Possible use of SVG as part of HbbTV/widgets on connected TVs, which are very resource-limited devices (even more than mobile phones), would make this study very useful. His detailed study of scene rendering should be emulated on SVG or LASeR or other scene format implementations, and results should be used to provide feedback for the design of the scene format.

3.7.2. Multi-format rendering

GPAC [13] is an open source multimedia framework for research and academic purposes in different aspects of multimedia, with a focus on presentation technologies (graphics, animation and interactivity). GPAC is cross-platform. It is written in C, for portability to embedded platforms and DSPs, and for low memory footprint. It is currently running under Windows, Linux, WindowsCE, Embedded Linux, SymbianOS, Mac OSX 10.4+ and iPhoneOS3.
The project started in 2003 with the initial goal of developing clean software compliant to the MPEG-4 Systems standard, when Jean Lefeuvre joined the project. He implemented the “Advanced Text and Graphics” BIFS amendment designed by Cyril Concolato on the basis of his format comparison work [5]. Since then, the project has evolved and features now three sets of tools: a multimedia player, a multimedia packager and several servers.
The GPAC multimedia player differs from traditional audiovisual players because, in addition to its capabilities to play any video or audio format and its support for most of the existing delivery protocols, it focuses on graphics, animations and interactivity technologies. GPAC offers a unique integrated player capable of playing back audiovisual content mixed with 2D and/or 3D content in the following formats: MPEG-4 BIFS and LASeR (partial), W3C SVG (Tiny 1.2), W3D VRML and X3D.
GPAC is now a key tool for the research in interactive graphics:

- it served as a base in research on how to best render SVG [74];
- it still serves as a base for research on streaming multimedia, from hinting for RTP, to usage in broadcast environments, to experiments with MPEG Dynamic Adaptive Streaming over HTTP (DASH);
- it serves as a design/development platform for W3C Widgets and MPEG-U [6] [7] [8] [9] [41] [42];
- it is used in rendering for new 3D screens requiring multi-views[75].

3.8. Widening the scope

Interactive services, on mobiles or Internet, usually include some text and some graphics. For a long time, you had to choose: either you chose "text", "document" and mostly "HTML", or you chose "graphics" and "scenes" and formats mentioned in this thesis. Text-based services were usually not so good with graphics rendering and manipulation. Graphics-based services were usually not so good with text rendering and manipulation. I have participated in multiple actions to reconcile the two worlds: CDF as an attempt to make the two work together, specifying the unspecified interstices; widgets as a tool which applies to both; and HbbTV as an attempt to include graphics in a text-based service world.

3.8.1. Compound Document Format

In the early days of Streamezzo, I participated in the creation of the W3C working group called Compound Document Format (CDF) [62], until the completion of WICD Core 1.0 [63]. The scope was to make multi-format content work well. The initial goal was to have HTML and SVG work well together as separate documents. Other formats were obviously in the picture too: CSS, DOM, ECMA-Script, XFORMS, MathML etc. The work was split in:

- multi-document content, with links between the various documents, with its first instance named WICD;
- single-document, multi-format content: this means using XML namespaces to be able to switch, within a document, from describing xHTML to describing SVG, then possibly X3D, etc, and then back to xHTML to close the document.

The work consisted in identifying the cracks in-between formats:

- how to consistently negotiate rendering space between a host format and an included format (e.g. an SVG drawing inside an xHTML page).
- how to pass events or the focus between engines managing different formats.
- how to handle fonts, synchronization, hyperlinking in a coherent manner across formats.
- how to allow a script in a part of a document to access objects in another part using another format.

3.8.1.1. Analysis

My participation in this working group was out of interest for the possible applications, and to make sure that LASeR could be used in CDF/WICD in the place of SVG, if needed. The group started with an ambition of having a first release in 6-8 months. Obviously, it took much longer. When the usage time-frame of the technology slipped beyond one or two years, my participation was not justified any more.

Still, this was my first contact with content as a mix between (x)HTML and SVG, between text-based content and rich media. We had the problem at Streamezzo that many of the applications we were designing would have been easier to design with some of the features of text-based formats, which rich media missed:

- longer texts, laid out in paragraphs, with styling
- tables with automatic layout à la CSS

We felt that including a subset of xHTML in rich media was important to increase the relevance of our rich media format and decrease the cost of service design. But it was a development cost and disruption we could not afford.
Today, the work on HTML5 is a continuation of the CDF single-document work, with a specification of how SVG can be rendered when specified inline in HTML5 content.

3.8.2. Widgets

Widgets are a more recent aspect of my work in the domain. Widgets are a mixture of content fragments and behaviour, expressed in web standards. They represent a break up of the HTML page limitation: users can assemble widgets in their browsers, and thus recompose information. They do not need to depend on portals to aggregate content, each person can be his/her own aggregator of fragments packaged as widgets.
The most frequent behaviour of a widget is to fetch data and to organize it for easy viewing. But with the increase in the availability of many types of APIs, there are fewer and fewer limits to what a widget can do.
Widgets are also not limited to computers, desktops, laptops or netbooks. Widgets are available on many smaller devices and platforms. And that is one of their big advantage: widgets are cross-platform and cross-device.
Our interest in widgets is focused on some concepts:

- widgets should be able to discover and communicate with other entities, services or other widgets, in order to create a cooperation ecosystem;
    - one variant is the creation of task-specific UIs for services;
    - another variant is the splitting of a task into multiple widgets, each of which may run on another device;
- widgets should not be tied to a device, e.g. to the particular device they first ran on: it should be possible to start a widget on a device, then move it to another without losing context.

We based our work on widgets on the W3C widgets initiative [41] [42], which defines:

- a packaging format, with an internationalization scheme and many default mechanisms;
- a manifest, which declares the components of the widget;
- a set of scripting APIs and events.

Our work on widgets and MPEG-U [77] has been partly funded by a contract with Samsung Electronics [6] [7] [8] [9].

Discovery

If a widget does not have discovery capabilities, it will need to be edited or configured specifically for a particular situation. This would exclude flexible situations, such as a home network when new devices can be added or old devices removed at any time. There is no preferred solution for discovery. We just assume the presence of a discovery mechanism, and we prefer those allowing the dynamic creation of service interfaces, such as UPnP (and unlike Bonjour). We have implemented widgets that discover UPnP services, widgets which announce themselves as UPnP services, etc…
The natural consequence of the discovery of a service that a widget knows about, is the connexion of the widget with that service. Such connexions are dynamic, for the duration when widget, service and network all keep working. If one of the three stops working, the connexion is cut, until the three elements are up again.

Communication

The communication contract is part of the definition of a widget. We have chosen to extend the W3C widget manifest with the exhaustive set of messages that a widget can send or receive from services or other widgets.
We use a communication abstraction centred on messages grouped into interfaces. Typically, an input message is composed of input parameters, followed by output parameters if a reply may be sent. An output message is composed of output messages, followed by input messages if a reply may be sent.
Messages can be sent and received (processed) by scripts. Messages can also be directly connected (without scripts) to the widget scene: widget scene events may trigger the sending of messages, and incoming messages may trigger modifications in the widget scene tree.

Mobility
During a session, where a user views information from a widget running on a particular device, there may appear another available device better suited for the viewing of information: for example, you have started searching for books on a commercial web site on your mobile phone, and you arrive home and want to switch to a larger screen for the end of your buying session. We have created mechanisms for the running widget to save an execution context and make the widget and its context available to another device in order to be able to pick up the session on this new device.

Standardization and Implementation
This work was standardized as part of MPEG-U, or ISO/IEC 23007.
We have implemented the system within the GPAC framework, in the following fashion:

- Support for the widget management has been implemented in C/C++ within GPAC, on top of the open source Platinum implementation of UPnP. Manifest parsing and W3C widget support is included.
- Widget managers, or widget user agents, are implemented in a mix of a scene description language and ECMA-Script:
    - the widget manager included in standard reference/utility software is based on BIFS and ECMA-Script, and has been tested with SVG and BIFS widgets; it is quite complete in terms of functionality, but complex to use and reserved for experts.
    - a simpler widget manager, emulating an iPhone UI, is based on SVG and ECMA-Script; it has been tested with SVG and BIFS widgets.
    - another SVG-based widget manager emulates a Yahoo TV connected widgets system.
- Many widgets have been implemented for demonstrations, engineering tests and conformance tests, in SVG and BIFS.

On-going work includes the implementation of all of the above on top of Webkit, to validate MPEG-U usage in the context of (x)HTML widgets, and later integration with HbbTV.

3.8.2.1. Analysis

With hindsight, as hinted in [6], this work has a wider potential scope than just widgets. Of course, widgets offer an easy application, e.g. because the interface declarations can be added to the widget manifest. But it should be possible to have discovery, communication and migration of content beyond widgets.

Scene/document formats are under constant pressure of extension to deal with e.g. new user interaction mechanisms. The newest is the multi-touch paradigm. The first implementers of multi-touch interfaces have extended the scene formats they use by adding a new set of events, which a non-interoperable extension: their scenes will not work elsewhere, and scenes designed for a mouse will not work on these devices either. To be able to deal in an interoperable manner with such extensions, it is better to use our way to declaring communication interfaces for the scene/document. In a sense, events are an ad-hoc interface to the scene/document. By creating a generic communication interface layer above/separate from the scene/document layer, we remove the need to constantly extend scene/document formats to deal with new interactions needs. This seems to us a major contribution/potential of MPEG-U.
We took a risk in mixing (again) W3C royalty-free standards with MPEG RAND standards, where such mixes do not have a good success record.
Another risk is the overwhelming trend to implement most things as ECMA-Script APIs: it would be relatively easy to redesign all the above as ES APIs [10] [71]; our advantage is that declarative interfaces are very easy to statically check, which is not the case of ES code.

3.8.3. HbbTV

HbbTV is a standard for interactive TV, resulting from the merge of a French project to add interactivity to TNT (French acronym for digital terrestrial TV, thus broadcast oriented), and a similar German project (more IPTV oriented). HbbTV is based on the W3C technologies xHTML and CSS, with somewhat hacked profiles, as well as the ubiquitous ECMA-Script. The xHTML set is very close to W3C profile: xHTML 1.1 transitional, there is just one extra element, which is a clean extension. The CSS set is much less clear, and contains some CSS3 that is not “standard” yet.
The largest specific part of the standard is a set of ECMA-Script APIs: some from CE-HTML [38], many from OIPF (Open IPTV Forum) [39], and a few specific to HbbTV around broadcast [40].
At first glance, HbbTV and W3C Widgets [41] [42] are very close, in the sense that HbbTV has all the basic elements of Widgets: xHTML, CSS and ECMA-Script. Even looking in detail, the additional load of implementing W3C widgets specific functionality is very small, almost negligible compared to the size of an HbbTV implementation. Many think there is a very good possible synergy between HbbTV and W3C Widgets: widgets develop fast on the general Internet as well as on the mobile Internet; as described in the previous section, widgets are very useful in the home network to deal with multiple small devices; a merge of all three worlds into the TV set has great potential, specially with a possible sharing of content and widgets across PCs, handsets and TVs.
We are working within a French multi-regional project called openHbb (initially, OpenWidget). Our most important task is to push for widgets and SVG in HbbTV version 2. Our technical tasks are:

- implement HbbTV in open source on PC on a WebKit base, adding widget and SVG support; this basically means to port our GPAC implementation of widgets to WebKit, and extend it to xHTML-based widgets.
- create open source authoring tools for HbbTV: a validator is in progress; next are bandwidth management and generators, i.e. from skinnable applications to full application generators.

3.9. Conclusion

I have tried to report the breadth and depth of my work in the domain of interactive content over the years, including the more academic work, the standards work as well as the commercial developments.

I have tried, in the analysis at the end of each project description, to describe my current view on the results. Let me try now to make a global analysis. If I were to work on a format now, I would:

- mix text-based and graphics features (xHTML and SVG),
- include an update mechanism, Conditional, etc… but also an efficient scripting interface, with XmlHttpRequest and many device APIs,
- not include specific compression, and rely on gzip, possibly EXI.
- work jointly on the format, rendering engines, authoring tools and a few key applications,
- try to stay with royalty-free standards and open source software, to ease the promotion,
- work at the interface between Internet, mobile and TV in the home environment.

For authoring, I would try to reuse the concepts, from Harmonia, of simple template-based authoring with more modern technologies, including a simple GUI and lots of ways for developers to incrementally improve the tool.
I would also separately work on a tool for authoring adaptable content, where the user can work on an unlimited number of alternate layouts, where the computer would create automatically the adaptation program by trying to interpolate linearly between alternate layouts, and when interpolation is impossible, it would ask the user to resolve the discontinuity. This tool would deal with spatial layout, temporal layout, content fragmentation, media alternates, etc. It would generate fully flexible layouts for powerful machines, as well as static versions for specific low-resource devices.
And for the application domain with highest potential, it seems clear that it is the fusion of applications on/in the constellation of devices to be found in a home: TV, PC, media center, mobile phones, tablets, game consoles, picture frames, etc… all cooperating/working together, communicating, sharing applications as well as content.

3.10. References

[1] Frédéric Bouilhaguet, “Architecture de systèmes MPEG-4”, Thèse ENST, 2001

[2] Souhila Boughoufalah, “Outils Auteurs pour MPEG-4”, Thèse ENST, 2003

[3] Gianluca Di Cagno, “ Systèmes multimedia et qualité d’expérience”, Thèse ENST, 2004

[4] Mariam Kimiaei Asadi, “Adaptation de contenu multimedia avec MPEG-21: conversion de resources et adaptation sémantique de scenes”, Thèse ENST, 2005

[5] Cyril Concolato, “Descriptions de scenes multimedia: representations et optimizations”, Thèse ENST, 2007

[6] C. Concolato, J.C. Dufourd, J. Le Feuvre, K.M. Park, J.Y. Song, “Communicating and Migratable Interactive Multimedia Documents », submitted to IEEE MTAP

[7] J.C. Dufourd, C. Concolato et J. Le Feuvre, “SVG Communicating Widgets”, 7th International Conference on Scalable Vector Graphics, October 2 to 4, 2009, Mountain View, California, USA

[8] C. Concolato, J. Le Feuvre et J.C. Dufourd, “Declarative Interfaces for Dynamic Widgets Communications”, 9th ACM Symposium on Document Engineering, Sept 15-18, 2009, Munich, Allemagne.

[9] J. Le Feuvre, C. Concolato et J.C. Dufourd, “Widget Mobility”, International Conference on Mobile Technology, Applications and Systems, Mobility 2009, 2 - 4 september 2009, Nice, France

[10] Sire, S., Paquier, M., Vagner, A., and Bogaerts, J. 2009. A messaging API for inter-widgets communication. In Proceedings of the 18th international Conference on World Wide Web (Madrid, Spain, April 20 - 24, 2009). WWW '09. ACM, New York, NY, 1115-1116. DOI= http://doi.acm.org/10.1145/1526709.1526884

[11] Jean-Claude Dufourd, « LASeR : The Lightweight Rich Media Representation Standard », Signal Processing Magazine, IEEE Volume 25, 2008, Pages 164-168, Issue : 6, DOI : http://dx.doi.org/10.1109/MSP.2008.929813.

[12] J.C. Dufourd, “La TMP n’est pas la TV sur mobile: le rôle de l’interactivité”, Revue Telecom, numéro TV Mobile, juin 08

[13] Le Feuvre, J., Concolato, C., and Moissinac, J. 2007. GPAC: open source multimedia framework. In *Proceedings of the 15th international Conference on Multimedia* (Augsburg, Germany, September 25 - 29, 2007). MULTIMEDIA '07. ACM, New York, NY, 1009-1012. DOI= http://doi.acm.org/10.1145/1291233.1291452

[14] J.C. Dufourd, “Services « Rich Media » mobiles”, Revue Telecom, septembre 06

[15] J. C. Dufourd, O. Avaro, et C. Concolato, “An MPEG standard for rich media services”, IEEE Multimedia, 12(4):60–68, Decembre 2005.

[16] M. Kimiaei-Asadi et J. C. Dufourd, “Support de transmodage de contenus multimédia dans MPEG-21, étude et validation d’un outil de description”, Revue des sciences et technologies de l’information, 24(7):815–835, July 2005.

[17] J. C. Dufourd et M. Kimiaei-Asadi, “Context-aware semantic adaptation of multimedia presentations”, In IEEE International Conference on Multimedia and Expo, pages 362–365, Amsterdam, The Netherlands, July 2005.

[18] Gianluca Di Cagno, Cyril Concolato, J. C. Dufourd, “Multimedia adaptation in end-user terminals”, Signal Processing: Image Communication vol. 21, n° 3, pp. 200-216, 2004.

[19] P. Gioia, K. Kamyab, I. Wolf, G. Panis, A. Difino, M. Kimiaei, T. Digiacomo, A. Cotarmanac'H, P. Goulev, A. Graffunder, A. Hutter, B. Negro, C. Concolato, C. Joslin, E. Mamdani, J. C. Dufourd et N. Thalmann, “ISIS: intelligent scalability for interoperable services”, (2004), 1st european Conference on Visual Media Production, Londres, Angleterre, pp. 295-304.

[20] C. Concolato et J. C. Dufourd, (2004), « Adaptation de contenu MPEG-4 BIFS suivant la norme MPEG-21 », Conférence nationale Mcube MultiMédia Mobile, Montbéliard, France.

[21] Cyril Concolato, J. C. Dufourd, J. C. Moissinac, "Creating and Encoding of Cartoons Using MPEG-4 BIFS: Methods and Results", IEEE Transactions on Circuits and Systems for Video Technology, (T-CSVT), special issue on Image-based Modeling, Rendering and Animation, Volume: 13 Issue: 11 Nov 2003 pp. 1129-1135.

[22] J. C. Dufourd, «MPEG-4 XMT versus SMIL/SVG», article invité, The Synchronised Multimedia Integration Language European Conference, Paris, February 12, 13, 14, 2003

[23] Cyril Concolato, J. C. Dufourd, J. C. Moissinac, « Representing 2D cartoons using SVG », The Synchronised Multimedia Integration Language European Conference, Paris, February 12, 13, 14, 2003

[24] C. Concolato, J.-C. Dufourd, “Comparison Of MPEG-4 BIFS And Some Other Multimedia Description Languages”, Workshop and Exhibitions on MPEG-4, San Jose, California, June 2002

[25] J. C. Dufourd, « BIFS: Scene Description », Chapitre 4 de “The MPEG-4 Book”, édité par Fernando Pereira et Touradj Ebrahimi, pp 103-148, Prentice Hall, 2002

[26] David Singer, « The MPEG-4 file format », Chapitre 7.3 de “The MPEG-4 Book”, édité par Fernando Pereira et Touradj Ebrahimi, pp 103-148, Prentice Hall, 2002

[27] J.C. Moissinac, C. Concolato, J.C. Dufourd, "Codage MPEG-4 de dessins animés", Journées Coresa, 12-13 novembre 2001.

[28] F. Bouilhaguet, C. Concolato, S. Boughoufalah, J. C. Dufourd, "Adding Delivery Support to MPEG-PRO, an Authoring System for MPEG-4", Workshop and Exhibitions on MPEG-4, 2001, June 18-20 San Jose.

[29] S. Boughoufalah, M. Brelot, F. Bouilhaguet, J.C. Dufourd, “A Template-Guided Authoring Environment To Produce Mpeg-4 Content For The Web”, International Conference on Media Futures, Firenze, May 2001.

[30] F. Bouilhaguet, C. Concolato, S. Boughoufalah, J.C. Dufourd, “Adding Delivery support to MPEG-Pro, an Authoring System for MPEG-4”, Proceedings PacketVideo '01, Kyongju, Corée, 1er mai 2001.

[31] S. Boughoufalah, J.C. Dufourd & F. Bouilhaguet - "MPEG- Pro, an Authoring System for MPEG- 4 with Temporal Constraints and Template Guided Editing", ICME 2000, New York, July 30 - August, 2000

[32] S. Boughoufalah, J.C. Dufourd & F. Bouilhaguet - "MPEG-PRO an MPEG-4 Authoring System", ISCAS 2000, Geneva, May 28 - 31, 2000

[33] F. Bouilhaguet, S. Boughoufalah, J.C. Dufourd & C. Havet, "Interactive Broadcast Digital Television The OpenTV Platform versus the MPEG-4 Standard Framework", ISCAS 2000, Geneva, May 28-31, 2000

[34] J. Signès, Y. Fisher, A. Eleftheriadis, “MPEG-4’s Binary Format for Scene Description”, Tutorial Issue On The MPEG-4 Standard, Image Communication Journal, Elsevier, August 1999

[35] P. Gerken, S. Schultz, G. Knabe, F. Casalino, G. Di Cagno, M. Quaglia, J.C. Dufourd, S. Boughoufalah, F. Bouilhaguet, M. Stepping, T. Bonse, U. Mayer, J. Deicke & M. Glesner - "MPEG-4 PC : Authoring and Playing of MPEG-4 Content", EMMESEC 99, Stockholm, June 21-26, 1999. Proceeding : Business and Work in the Information Society : New Technologies and Applications, Edited by Jean-Yves Roger, Brian Stanford-Smith & Paul T. Kidd

[36] Joint Innovation Lab (JIL), JIL Widget System API Specification-Handset API 1.2.2, http://www.jil.org/c/document_library/get_file?uuid=03eb4771-e9a2-42fc-9874-0bb402e0244c&groupId=10158

[37] OMTP BONDI, http://bondi.omtp.org

[38] CEA-2014-A, http://www.ce.org/Standards/browseByCommittee_2757.asp

[39] Open IPTV Forum, http://www.openiptvforum.org/specifications.html

[40] Hybrid Broadband Broadcast Tele-Vision, ETSI TS 102 796 v1.1.1, http://pda.etsi.org/pda/home.asp?wki_id=hsZ9g8-p%27v475895z4LHX

[41] W3C, Editor: Marcos Caceres, “Widgets Packaging and Configuration”, available at http://www.w3.org/TR/widgets/

[42] W3C, Editor: Marcos Caceres, “The Widget Interface”, available at http://www.w3.org/TR/widgets-apis/

[43] Information technology – Coding of audio-visual objects – Part 1: Systems, ISO/IEC 14496-1:2004

[44] Information technology – Coding of audio-visual objects – Part 11: Scene description (BIFS) and application engine (MPEG-J), ISO/IEC 14496-11:2005

[45] Information technology – Coding of audio-visual objects – Part 12: ISO base media file format, ISO/IEC 14496-12:2005

[46] Information technology – Coding of audio-visual objects – Part 14: MPEG-4 File Format (MP4), ISO/IEC 14496-14:2003

[47] Information technology – Coding of audio-visual objects – Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF), ISO/IEC 14496-20:2005

[48] Information technology – MPEG systems technologies – Part 1: Binary MPEG format for XML (BiM), ISO/IEC 23001-1:2006

[49] XHTML 1.0, The Extensible HyperText Markup Language (Second Edition), W3C Recommendation, 26 January 2000, revised 1 August 2002, http://www.w3.org/TR/html/

[50] Synchronized Multimedia Integration Language Specification (SMIL 2.1), W3C Recommendation, 13 December 2005, http://www.w3.org/TR/SMIL/

[51] Scalable Vector Graphics (SVG) Tiny 1.2 Specification, W3C Recommendation, 22 December 2008, http://www.w3.org/TR/SVGMobile12/

[52] Cascading Style Sheets, level 2 (CSS2) Specification, W3C Recommendation, 12 May 1998, http://www.w3.org/TR/REC-CSS2/

[53] Document Object Model (DOM) Level 3 Core Specification, W3C Recommendation, 07 April 2004, http://www.w3.org/TR/DOM-Level-3-Core/

[54] Extensible Markup Language (XML) 1.0 (Fourth Edition), W3C Recommendation, 16 August 2006, http://www.w3.org/TR/xml/

[55] Remote Events for XML (REX) 1.0, Working Draft 02 February 2006, http://www.w3.org/TR/rex/

[56] Efficient XML Interchange (EXI) Format 1.0, W3C Candidate Recommendation 08 December 2009, http://www.w3.org/XML/exi/

[57] XML Schema Part 0: Primer Second Edition, W3C Recommendation, 28 October 2004, http://www.w3.org/TR/xmlschema-0/

[58] XSL Transformations (XSLT), Version 1.0, W3C Recommendation, 16 November 1999, http://www.w3.org/TR/xslt

[59] XML Events, An Events Syntax for XML, W3C Recommendation, 14 October 2003, http://www.w3.org/TR/xml-events/

[60] The XMLHttpRequest Object, W3C Candidate Recommendation, 3 august 2010, http://www.w3.org/TR/XMLHttpRequest/

[61] Efficient XML Interchange (EXI) Format 1.0, W3C Candidate Recommendation 08 December 2009, http://www.w3.org/TR/exi/

[62] Compound Document Format, W3C working group, http://www.w3.org/2004/CDF/

[63] WICD Core 1.0, W3C Candidate Recommendation 18 July 2007, http://www.w3.org/TR/WICD/

[64] ECMAScript Language Specification, 3rd edition (December 1999), www.ecma-international.org/publications/standards/Ecma-262.htm

[65] The Virtual Reality Modeling Language (VRML) specification, http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/

[66] The Extensible 3D (X3D) specification, http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/

[67] 3GPP TS 26 142, Dynamic Interactive Multimedia Scenes (DIMS), http://pda.etsi.org/pda/home.asp?wkr=RTS/TSGS-0426142v730

[68] 3GPP TS 26 244, Transparent end-to-end packet switched streaming service (PSS); 3GPP file format(3GP), http://www.3gpp.org/ftp/Specs/html-info/26244.htm

[69] OMA Rich Media Environment (RME) v1.0, http://www.openmobilealliance.org/technical/release_program/rme_v1_0.aspx

[70] Douglas Crockford, “JavaScript: The World's Most Misunderstood Programming Language”, http://javascript.crockford.com/javascript.html

[71] Sire, S., Paquier, M., Vagner, A., and Bogaerts, J. “A messaging API for inter-widgets communication”. In Proceedings of the 18th international Conference on World Wide Web (Madrid, Spain, April 20 - 24, 2009). WWW '09. ACM, New York, NY, 1115-1116. DOI= http://doi.acm.org/10.1145/1526709.1526884

[72] H. V. O. Silva, R. F. Rodrigues, L. F. G. Soares, and D. C. M. Saade. “Ncl 2.0: integrating new concepts to xml modular languages”. In ACM DOCENG’04, pages 188–197, 2004.

[73] Apple Computer, Inc. (1988). “HyperCard Script Language Guide: The HyperTalk Language”. Addison-Wesley Publishing Company. ISBN 0-201-17632-7.

[74] C. Concolato, J. Le Feuvre, J. C. Moissinac, “Design of an Efficient Scalable Vector Graphics Player for Constrained Devices”, IEEE Transactions on Consumer Electronics, Mai 2008, vol. 54, n° 2, pp. 895-903, DOI: 10.1109/TCE.2008.4560176

[75] J. Le Feuvre, “SVG Extensions for 3D displays”, 8th International Conference on Scalable Vector Graphics, August 30 to September 1, 2010, Paris, France

[76] ISO/IEC 14496-20:2008/Amd.3:2010, Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF) AMENDMENT 3: Presentation and Modification of Structured Information (PMSI)

[77] Information technology – MPEG systems technologies – Part 1: Widgets, ISO/IEC 23007-1:2011 (also known as MPEG-U)

[78] Nabil Layaida, “Madeus: système d’édition et de présentation de documents structurés multimedia”, Thèse, Université Joseph Fourier Grenoble1, 1997

[79] Marc Brelot, “Représentation orientées objet de scenes visuelles pour la composition flexible”, Thèse, INP Grenoble, 1999

[80] Romain Deltour, Cécile Roisin, “The LimSee3 Multimedia Authoring Model”, DocEng’06, October 10–13, 2006, Amsterdam, The Netherlands

[81] SébastienLaborie, Jérome Euzenat, Nabil Layaïda, "Semantic Multimedia Document Adaptation with Functional Annotations" In Proceedings of the 4th

International Workshop on Semantic Media Adaptation and Personalization, SMAP'09, IEEE, December 2009

4. Curriculum Vitae

de Jean-Claude Dufourd
Adresse : 1 bis, Impasse Courteix, 94270 Le Kremlin Bicêtre, France
Mobile : +33 677 843 843
Email: jc@dufourd.org ou jean-claude.dufourd@telecom-paristech.fr
Né le 8 novembre 1960, à Metz (Moselle)
Nationalité française

4.1. Postes occupés

avril 2009-présent Télécom ParisTech, Directeur d'Etudes

- OpenWidget : préparation et rédaction du projet FUI8 OpenWidget, avec une dizaine de partenaires, projet accepté en cours de négociation, sur la TV interactive.
- HBBTV : nouveau standard ETSI qui servira de support à OpenWidget, contribution au standard / implémentation / authoring / interopérabilité.
- Widget : travail sur une extension du standard W3C Widget pour la communication entre widgets, la mobilité des widgets, et leur usage dans un contexte iTV et réseau domestique, en plus de mobile et Internet.

nov 2004-dec 2008 Co-fondateur et Directeur Scientifique, Streamezzo, Paris

- Gestion du processus brevets dans Streamezzo : incubation des idées, rédaction des demandes, dépôt et gestion de la vie des brevets, insertion des brevets dans des standards, dépôt de 29 brevets dont 23 en tant qu'inventeur ou co-inventeur.
- Gestion des aspects techniques de la standardisation autour du Rich Media, MPEG LASeR (éditeur et principal contributeur), 3GPP DIMS (contributeur majeur) and OMA RME (contributeur majeur), plus récemment avec l'accent sur le broadcast ; participation active dans le passé à W3C SVG autour de SVGT1.2 et à CDF autour de CDR / WICD 1.0
- Validation scientifique des propositions de projets collaboratifs et contribution à leur développement (projets techniques à financement européen, national ou régional).

1990-oct 2004 Directeur d'Etudes : Ecole Nationale Supérieure des Télécommunications, Paris

- Pour 2004, construction du projet d'essaimage Streamezzo.
- Gestion de l'équipe MPEG-4, une des équipes les plus performantes de l'ENST (sur la période '98 – '04) en termes de nombre de contrats et de chiffre d'affaire.

- Directeur de thèse de 8 docteurs :
    - Saïd Boumaraf, «S-intervalles : un nouveau modèle pour la gestion des contraintes - Application à la migration de VLSI sous Préforme», 1995
    - Aïssam Mezhoud, «Application de la programmation par contraintes aux problèmes de CAO des circuits intégrés : allocations d'interconnexions et floor planning orienté communications», 1996
    - Mahieddine Bouzidi, «Estimation de la consommation au niveau architectural», 09/1999
    - Frédéric Bouilhaguet, «Architecture de systèmes MPEG-4», 10/2001
    - Souhila Boughoufalah, «Outils auteur pour MPEG-4», 06/2002
    - Gianluca DiCagno, «Systèmes multimédia et qualité d'expérience», 10/2004
    - Mariam Kimiaei Azadi, «Adaptation de Contenu Multimédia avec MPEG-21: Conversion de Ressources et Adaptation Sémantique de Scènes», 06/2005
    - Cyril Concolato, «Descriptions de scènes multimédia : représentations et optimisations», 07/2007
- Participation à des jurys de thèse :
    - G. Chevallier (rapporteur) 05/1993, LIP6
    - F. Pétrot (rapporteur) 07/1994, LIP6
    - F.E. Moraes (rapporteur) 11/1994, LIRMM
    - C. Dabrin (rapporteur) 04/1995, LIRMM
    - N. Dictus (rapporteur) 06/1996, LIP6
    - J-M. Trivi (rapporteur) 07/2002, Inrialpes
    - Tien Tran Thuong (rapporteur) 02/2003, Inrialpes
    - D. Deuff (invité), 07/2003, Irisa
    - T. Lemlouma (examinateur) 06/2004, Inrialpes
- Participation active à MPEG.
- Contrats européens: management technique et contribution à un projet ACTS, un projet ESPRIT, 3 projets IST FP5 et 2 projets IST FP6.
- Contrats industriels: 6 sur des sujets liés à MPEG.
- Contrats nationaux: 3
- Activités de recherche : MPEG-4 et technologies Internet/mobiles : standards multimédia nouveaux (MPEG-4, Java, VRML/X3D, SVGT, SMIL, XML, XSLT, MPEG-7), authoring MPEG-4, conception d'applications nouvelles, qualité de service multimédia, représentation de scènes multimédia, MPEG-4 sur mobiles.

1987-1990 Chercheur : France Telecom R&D, Grenoble

4.2. Standards

MPEG Systems :

- Responsable du groupe Integration de MPEG de 2002 à 2006.
- Editeur et principal contributeur à ISO/IEC 14496-20 Lightweight Application Scene Representation and Simple Aggregation Format (LASeR and SAF) .
- Auteur et coauteur de plus de 300 contributions MPEG.
- Editeur de ISO/IEC 14496-4:2000 Conformance Testing, ainsi que de nombreuses éditions ultérieurs et documents d'amendements.
- Editeur de ISO/IEC 14496-5:2000 Reference software, ainsi que de nombreuses éditions ultérieurs et documents d'amendements.
- Coauteur de ISO/IEC 14496-1:2001 AMD2, eXtensible MPEG-4 Textual Format.
- Editeur de ISO/IEC 14496-11:2005, Scene description and application engine

3GPP:

- Participant au groupe SA4, contributeur technique majeur au standard DIMS (Dynamic Interactive Multimedia Scenes).

OMA:

- Participant au groupe BAC MAE (renommé BT MAE, renommé MCE), contributeur technique majeur au standard RME (Rich Media Environment).
- Participant au groupe BCAST.

W3C:

- Participant au groupe SVG, contributions à SVG Tiny 1.2 (Scalable Vector Graphics)
- Participant au groupe CDF (Compound Document Format), contributions à CDR (Compound Document by Reference) / WICD 1.0 (Web Interactive Compound Document)

DVB:

- Participant au groupe DVB TM-CBMS (groupe auteur du standard IPDC – IP DataCast) and TM-MIS.

4.3. Education

1985-87	Ecole Nationale Supérieure des Télécommunications, Paris (Ingénieur du Corps)
1983	PhD Informatique, Ecole Normale Supérieure/Université Paris VI
1980-85	Ecole Normale Supérieure de Paris (rue d'Ulm)

Langues étrangères

Anglais	Bilingue
Allemand	Courant

4.4. Publications

4.4.1.1. Livres

1. J. C. Dufourd, « BIFS: Scene Description », Chapitre 4 de “The MPEG-4 Book”, édité par Fernando Pereira et Touradj Ebrahimi, pp 103-148, Prentice Hall, 2002

4.4.1.2. Revues avec comité de lecture

1. Jean-Claude Dufourd, « LASeR : The Lightweight Rich Media Representation Standard », Signal Processing Magazine, IEEE Volume 25, 2008, Pages 164-168, Issue : 6, DOI : http://dx.doi.org/10.1109/MSP.2008.929813.
2. Gianluca Di Cagno, Cyril Concolato, J. C. Dufourd, “Multimedia adaptation in end-user terminals”, Signal Processing: Image Communication vol. 21, n° 3, pp. 200-216, 2006.
3. M. Kimiaei-Asadi et J. C. Dufourd, “Support de transmodage de contenus multimédia dans MPEG-21, étude et validation d’un outil de description”, Revue des sciences et technologies de l’information, 24(7):815–835, July 2005.
4. J. C. Dufourd, O. Avaro, et C. Concolato, “An MPEG standard for rich media services”, IEEE Multimedia, 12(4):60–68, Decembre 2005.
5. Cyril Concolato, J. C. Dufourd, J. C. Moissinac, "Creating and Encoding of Cartoons Using MPEG-4 BIFS: Methods and Results", IEEE Transactions on Circuits and Systems for Video Technology, (T-CSVT), special issue on Image-based Modeling, Rendering and Animation, Volume: 13 Issue: 11 Nov 2003 pp. 1129-1135.

4.4.1.3. Revues

1. J.C. Dufourd, “La TMP n’est pas la TV sur mobile: le rôle de l’interactivité”, Revue Telecom, numéro TV Mobile, juin 08
2. J.C. Dufourd, “Services « Rich Media » mobiles”, Revue Telecom, septembre 06

4.4.1.4. Conférences avec comité de lecture

1. J.C. Dufourd, C. Concolato et J. Le Feuvre, “SVG Communicating Widgets”, 7th International Conference on Scalable Vector Graphics, October 2 to 4, 2009, Mountain View, California, USA
2. C. Concolato, J. Le Feuvre et J.C. Dufourd,“Declarative Interfaces for Dynamic Widgets Communications”, 9th ACM Symposium on Document Engineering, Sept 15-18, 2009, Munich, Allemagne.
3. J. Le Feuvre, C. Concolato et J.C. Dufourd, “Widget Mobility”, International Conference on Mobile Technology, Applications and Systems, Mobility 2009, 2 - 4 september 2009, Nice, France
4. J. C. Dufourd et M. Kimiaei-Asadi, “Context-aware semantic adaptation of multimedia presentations”, In IEEE International Conference on Multimedia and Expo, pages 362–365, Amsterdam, The Netherlands, July 2005.
5. P. Gioia, K. Kamyab, I. Wolf, G. Panis, A. Difino, M. Kimiaei, T. Digiacomo, A. Cotarmanac'H, P. Goulev, A. Graffunder, A. Hutter, B. Negro, C. Concolato, C.

Joslin, E. Mamdani, J. C. Dufourd et N. Thalmann, “ISIS: intelligent scalability for interoperable services”, (2004), 1st european Conference on Visual Media Production, Londres, Angleterre, pp. 295-304.

6. C. Concolato et J. C. Dufourd, (2004), « Adaptation de contenu MPEG-4 BIFS suivant la norme MPEG-21 », Conférence nationale Mcube MultiMédia Mobile, Montbéliard, France.
7. J. C. Dufourd, «MPEG-4 XMT versus SMIL/SVG», article invité, The Synchronised Multimedia Integration Language European Conference, Paris, February 12, 13, 14, 2003
8. Cyril Concolato, J. C. Dufourd, J. C. Moissinac, « Representing 2D cartoons using SVG », The Synchronised Multimedia Integration Language European Conference, Paris, February 12, 13, 14, 2003
9. C. Concolato, J.-C. Dufourd, “Comparison Of MPEG-4 BIFS And Some Other Multimedia Description Languages”, Workshop and Exhibitions on MPEG-4, San Jose, California, June 2002
10. J.C. Moissinac, C. Concolato, J.C. Dufourd, "Codage MPEG-4 de dessins animés", Journées Coresa, 12-13 novembre 2001.
11. F.Bouilhaguet, C.Concolato, S.Boughoufalah, J. C. Dufourd, "Adding Delivery Support to MPEG-PRO, an Authoring System for MPEG-4", Workshop and Exhibitions on MPEG-4, 2001, June 18-20 San Jose.
12. S. Boughoufalah, M. Brelot, F. Bouilhaguet, J.C. Dufourd, “A Template-Guided Authoring Environment To Produce Mpeg-4 Content For The Web”, International Conference on Media Futures, Firenze, May 2001.
13. F. Bouilhaguet, C. Concolato, S. Boughoufalah, J.C. Dufourd, “Adding Delivery support to MPEG-Pro, an Authoring System for MPEG-4”, Proceedings PacketVideo '01, Kyongju, Corée, 1er mai 2001.
14. S. Boughoufalah, J.C. Dufourd & F. Bouilhaguet - "MPEG- Pro, an Authoring System for MPEG- 4 with Temporal Constraints and Template Guided Editing", ICME 2000, New York, July 30 - August, 2000
15. S. Boughoufalah, J.C. Dufourd & F. Bouilhaguet - "MPEG-PRO an MPEG-4 Authoring System", ISCAS 2000, Geneva, May 28 - 31, 2000
16. S. Boughoufalah, J.C. Dufourd & F. Bouilhaguet - "Interactive Broadcast Digital Television The OpenTV Platform versus the MPEG-4 Standard Framework", ISCAS 2000, Geneva, May 28-31, 2000
17. P. Gerken, S. Schultz, G. Knabe, F. Casalino, G. Di Cagno, M. Quaglia, J.C. Dufourd, S. Boughoufalah, F. Bouilhaguet, M. Stepping, T. Bonse, U. Mayer, J. Deicke & M. Glesner - "MPEG-4 PC : Authoring and Playing of MPEG-4 Content", EMMESEC 99, Stockholm, June 21-26, 1999. Proceeding : Business and Work in the Information Society : New Technologies and Applications, Edited by Jean-Yves Roger, Brian Stanford-Smith & Paul T. Kidd
18. M. Bouzidi, J.C. Dufourd, « Estimation de consommation à un haut niveau d'abstraction: Etat de l'art et proposition d'une méthodologie standard », Workshop Faible Tension et Faible Consommation, 21 Novembre 1997, ISEP, Paris.

19. M. Bouzidi, J.C. Dufourd, « High-level Power Estimation Methodology », Workshop on Low-Power and Low-Voltage during ESSIRC'97, 19 Septembre 1997, Southampton.
20. A. Mezhoud, J.C. Dufourd, N. Darbel, "Performance-driven Interconnection Optimization based on Constraint Programming", Proceedings PACT 96, Londres, Avril 96.
21. A. Demeure, A. Lafage, J.C. Dufourd, E. Boutillon, JL. Marro, D. Rozzonelli, "Array-OL : Proposition pour un formalisme Tableau pour le traitement de signal MultiDimensionnel", Actes du 5ème Colloque GRETSI, Juan-LesPins, Septembre 1995
22. A. Mezhoud, J.C. Dufourd, N. Darbel, "Min-Cut Linear Arrangement based on Constraint Programming", CP 95 Workshop on Studying and Solving Really Hard Problems, Marseille, Septembre 1995.
23. S. Boumaraf, N. Darbel, J.C. Dufourd, "Technological Migration based on Physical Layout Skeletonization and Symbolic Layout Compaction", X SBMICRO / IBERMICRO, Canela, Brésil, Août 1995
24. S. Boumaraf, G. Chevallier, J.C. Dufourd, "Conception Symbolique et Algorithmes de Compactage : un regard sur l'état de l'art", Revue Internationale des Technologies Avancées, Décembre 1994.
25. J.C. Dufourd, JF. Naviner, "An optimizable model for process independent symbolic design", Proceedings. The European Design and Test Conference. EDAC, The European Conference on Design Automation. ETC European Test Conference. EUROASIC, The European Event in ASIC Design (Cat. No.94TH0634-6), p. 660, 28 Feb.-3 March 1994, Paris, France
26. A. Mezhoud, J.C. Dufourd, "Components Placement in VLSI DAtapath Based on Constraint Programming", International Logic Programming Symposium 1994, Workshop on Constraint Programming Languages, Systems and their Use in Problem Modelling, Ithaca, New York (USA) 18 novembre 1994.
27. J.C Dufourd, J.F. Naviner, Y. Mathieu, "aGAPE : A graphical Assembly Prototype Editor", IFIP WG 10.5, Grenoble, Mars 1992.
28. J.F Naviner, J.C. Dufourd, "Preforme/aGAPE : a synergy between symbolic cell design and assembly", Proc. of EUROMICRO conference, Paris, 1992.
29. J.C. Dufourd,"Preforme benchmarking", présentation à International Workshop on symbolic layout (OMCNC 92), 1992.
30. J.C. Dufourd, J.F. Naviner, F. Jutand, "PREFORM: A Process independent symbolic Layout System", International Conference on Computer Assisted Design 90, Santa Clara, USA, Novembre 1990.
31. J.C. Dufourd, "The STICKIZER : a layout to symbolic converter", International Conference on Computer Assisted Design 89, Santa Clara, USA, Novembre 1989.

4.4.1.5. Brevets

Inventeur de 24 brevets ou demandes de brevets, en tant qu'ENST ou que Streamezzo

1. WO02056595, Method and equipment for managing interactions in the MPEG-4 standard, Dufourd JC et al.
2. Demandes de Brevets Streamezzo: toutes ces demandes sauf la 6ème sont des demandes internationales.

Titre	Date de dépôt	N° Dépôt INPI	Inventeurs
Procédé de construction de scènes multimédia comprenant au moins un objet pointeur, procédé de restitution de scènes, terminal, programmes d'ordinateur, serveur et objet pointeur correspondants	29/03/2005	0503048	Jean-Claude DUFOURD
Procédé de contrôle d'interface d'une pluralité de types de terminaux de radiocommunication par la définition d'événements abstraits, programme d'ordinateur, signal, et terminal correspondants	14/09/2005	0509411	Jean-Claude DUFOURD, Julien PERRON
Procédé de transmission d'un contenu multimédia vers un terminal de radicommunication, programme d'ordinateur, signal, terminal de radiocommunication et serveur de diffusion correspondants.	14/09/2005	0509409	Jean-Claude DUFOURD, Cécric GEGOUT, Elouan LECOQ
Procédé d'optimisation de rendu d'une scène multimédia, programme, signal, support de données, terminal et procédé de réception correspondants.	02/11/2005	0511177	Jean-Claude DUFOURD, Christophe MICHEL, Arnaud CAIGNIET
Procédés de création et de restitution optimisés du rendu d'une scène multimédia comprenant au moins un objet actif sans modification préalable de la sémantique et/ou du format de description de scène	21/06/2006	0605563	Jean-Claude DUFOURD, Julien PERRON, Nicolas PIERRE
Procédé de modification d'un contenu, serveur, signal et produit programme d'ordinateur correspondants. **(demande française seulement)**	13/10/2006	0609014	Jean-Claude DUFOURD, Olivier AVARO, Gaelle MARTIN-COCHER
Procédé de gestion de mémoire dans un terminal client, signal, programme d'ordinateur et terminal correspondants.	18/10/2006	0609144	Julien PERRON, Elouan LECOQ Jean-Claude DUFOURD
Procédé de description de scène multimédia comprenant au moins une zone de rognage rectangulaire alignée sur des frontières de pixels	20/10/2006	0609240	Jean-Claude DUFOURD, Olivier AVARO, Gaelle MARTIN-COCHER
Procédé de gestion de canaux de transmission, signal et terminal correspondants.	27/11/2006	0610362	Jean-Claude DUFOURD, Elouan LECOQ, Nicolas PIERRE
Procédé de gestion de polices de caractères dans un terminal de radiocommunication, pour restituer des contenus multimédia sur un écran, et terminal correspondant	25/01/2007	0700526	Cédric GEGOUT, Jen-Claude DUFOURD
Procédé de transmission d'au moins un contenu représentatif d'un service, depuis un serveur vers un terminal, dispositif et produit programme d'ordinateur correspondants.	02/02/2007	0700765	Ronan KEREBEL, Elouan LE COQ, Cédric GEGOUT, Jen-Claude DUFOURD
Procédé de création d'un contenu, procédé de suivi des actions d'utilisation d'un contenu, terminal et signaux correspondants	14/05/2007	0703460	Bertrand AUDINET, Benoît CANTIN, Jean-Claude DUFOURD, Vincent DUPAIN, Philippe LAFOUCRIERE, Laëtitia ORSINI

Procédé de gestion de la navigation, terminal et programme d'ordinateur correspondants, procédé de construction d'un graphe de scène et signal de description de scène	12/06/2007	0755706	Benoît LAGREE, Mathieu CORITON Arnaud CAIGNIET, Christophe MICHEL, Jean-Claude DUFOURD
Procédé de diffusion d'un élément complémentaire, serveur et terminal correspondants.	13/06/2007	0755745	Jean-Claude DUFOURD, Cécric GEGOUT
Procédé de création d'au moins un contenu, procédé d'optimisation du fonctionnement d'un terminal de radiocommunication, terminal, programme d'ordinateur et signal correspondants.	31/08/2007	0757291	Jean-Claude DUFOURD, Naresh SONI
Procédé de synchronisation d'une action RichMédia avec un changement audiovisuel, dispositif et programme d'ordinateur correspondants, procédé de création d'une présentation RichMédia et programme d'ordinateur correspondant.	21/12/2007	0760320	Pierre-Erwann GOUESBET, Stéphane BELLANGER, Jean-Claude DUFOURD
Procédé de décodage, terminal et programme d'ordinateur correspondants, procédé de traduction, serveur et programme d'ordinateur correspondants.	24/12/07	0760349	Jean-Claude DUFOURD
Procédé d'alimentation d'un mandataire de contenu multimédia, mandataire, et produit programme d'ordinateur correspondant.	24/01/2008	0850451	Vincent DUPAIN, Benoit CANTIN, Philippe LAFOUCRIERE, Jean-Claude DUFOURD
Procédé de redimensionnement d'un contenu vidéo, terminal et produit programme d'ordinateur correspondant.	25/01/2008	0850476	Yannick LE COLLEN, Pierre-Erwann GOUESBET, Jean-Claude DUFOURD
Procédé de restitution d'au moins un contenu multimédia personnalisé, terminal et programme d'ordinateur correspondants	14/3/08	08 51691	Jean-Claude DUFOURD, Elouan LECOQ, Sébastien BARIOU
Procédé de création d'un service, dispositif et programme d'ordinateur correspondants, procédé de génération de mise à jour d'un contenu, serveur, terminal et programme d'ordinateur correspondants	11/4/08	0852462	Christophe MICHEL, Julien PERRON, Jean-Claude DUFOURD
Procédé de sécurisation d'une scène évolutive, dispositif, signal et programme d'ordinateur correspondants, procédé de mise à jour d'une scène évolutive, dispositif et programme d'ordinateur correspondants.	23/4/08	0852742	Elouan LE COQ, Laurent MASSON, Erwann GOUESBET, Jean-Claude DUFOURD
Procédé de création d'une présentation, dispositif et programme d'ordinateur correspondants, procédé de restitution d'une présentation, dispositif et programme d'ordinateur correspondant	05/06/2008	0853719	Erwann GOUESBET, Guillaume FORET, Jean-Claude DUFOURD

4.5. Contrats de recherche (ENST)

4.5.1.1. Contrats européens

EMPHASIS : ACTS 105, contrat en commun avec le département INF, 17 partenaires dont STM, Philips LEP, Matra: "Architecture Software and Hardware for MPEG-4 Systems", 1995-98, contribution ENST : 90hm.

MPEG-4 PC : ESPRIT 23191, 5 partenaires (Q-Team Allemagne, TAO UK, ITK Allemagne, CSELT Italie) : "MPEG-4 System Implementation and Tools for Personal Computers"

1997-2000. Contribution ENST : 70hm. Le logiciel mp4tool a commencé à être développé dans le cadre de ce projet.

SoNG : IST FP5, géré par le département Images et Sons de l'ENST Bretagne, sur un lecteur MPEG-4 3D, notre contribution étant centrée sur les nouveaux senseurs MPEG-4 et le format XMT, intermédiaire entre MPEG-4 et SMIL. Contribution ENST : 24hm. 1999-2002. Le logiciel mp4tool a étendu à la 3D et a couvert l'ensemble de BIFS dans le cadre de ce projet.

ISIS : IST FP5, adaptation et scalabilité (statiques) des contenus avec application au domaine mobile, contribution à MPEG-4 et MPEG-21, contribution ENST: 32hm. 2002-2003. Le logiciel B4, développé en background, a été utilisé et complété dans le cadre de ce projet.

MELISA : IST FP5, pari sportif en temps réel avec MPEG-4 sur des terminaux grand public et des terminaux mobiles, contribution ENST : 17hm. 2003-2004. Le logiciel B4 a été utilisé dans le cadre de ce projet et des fonctionalités temps réel sont ajoutées.

TIRAMISU: IST FP6, gestion des droits sur les contenus MPEG-4 avec MPEG-21, contribution ENST : 38hm, nov. 2003-2005.

DANAE : IST FP6, adaptation et scalabilité dynamiques des contenus, MPEG-21, poursuite du contrat ISIS, contribution ENST : 63 hm, 2004-2006

4.5.1.2. Contrats nationaux

OpenWidget, renommé openHbb : labellisé CapDigital et Images&Réseaux, projet FUI8, implémentation et démonstration de HbbTV, une nouvelle norme de TV interactive, pressentie pour la TNTi, en collaboration avec le HD Forum (surtout France TV), Le télégramme, WizTiVi, mediatvcom, TDF, Streamezzo, V4X, 72hm financés, 2009-2011.

Contrat Priamm financé par le Centre National du Cinéma en collaboration avec la société MediaPEGS et le département INFRES : "L'utilisation de MPEG-4 dans le domaine du dessin animé", 2001. Ce contrat a étudié la traduction de dessins animés en MPEG-4 à partir d'un format propriétaire nommé ECS.

SAMP4 (Contrat RIAM) : extension de nos outils auteurs MPEG-4 pour prendre en compte la gestion des droits d'accès sur les contenus musicaux, en collaboration avec la jeune pousse de F. Bouilhaguet. (11hm financés), 2002-2003. Le logiciel mp4tool a fait l'objet d'extensions au cours de ce contrat.

MP4TV (Contrat RIAM) : MPEG-4 pour la TV interactive, contribution ENST : 22hm, 2004-2005

4.5.1.3. Contrats industriels

Les contrats avec CNET et VLSI Technologies, concernant la CAO de VLSI, sont omis.

MPEG-Pro : Mise en commun de ressources avec le CSELT et le CNET : "Programme commun de recherche sur un outil auteur MPEG-4 2D et 3D", 1998-1999. Une première version de la gestion du format de fichier MP4 utilisée dans mp4tool a été développée dans ce projet.

Contrat TDK de développement d'un outil auteur MPEG-4 grand public : notre partie est le cœur MPEG-4 de l'outil auteur. Les partenaires sont l'Ecole des Mines pour un composant « segmentation d'images », le CSELT pour la partie « lecteur MPEG-4 », Optibase (Israel) pour les « codecs », et TDK (Japon) pour le marketing. (900KF+royalties). 1999-2003. Une partie des développements de MPEG-Pro a été utilisée comme bibliothèque dans ce projet, mais aucun des développements du projet TDK n'a été réutilisé ultérieurement.

Transfert de technologie vers e-Vue (Boston, USA): "Collaboration pour le transfert de compétences sur le format de fichier MP4", 2000

DAM4 (Contrat de recherche FT R&D) : production de contenus MPEG-4 pour des applications Internet ou mobiles : étude de faisabilité de dessins animés en MPEG-4 sur GPRS. Ce contrat n'a pas fait l'objet de développement. 2002-2003

MULTAD (Contrat de recherche FT R&D) : contrat de financement de la thèse de Mariam Kimiaei Asadi. Le sujet est aligné sur ceux des contrats ISIS et DANAE. 2002-2004

4.5.1.4. Contrats d'étude

GEMINI : Projet sur crédits incitatifs GET en collaboration avec l'équipe Artémis de l'INT : "Interfaces génériques pour MPEG-4", 2000.