

Scene Description for Mobile devices

Dr. Hari Kalva
Florida Atlantic University
hari@cse.fau.edu

Manish J Grover
Florida Atlantic University
mgrover1@fau.edu
COT 6930 Spring 2004

Abstract

There is a growing need for video to be delivered to mobile devices. As advances in technology bridge the gaps in communication, more and more people are using mobile devices such as cellular phones, PDAs, and laptop computers. The requirements for mobility also require increased use of wireless connectivity. This paper will explore the current video communication standards and will attempt to find optimizations in the way scenes are described. Considering the limitations of the mobile network and end terminals that exist today, the paper will attempt to examine the scene description standards in such a way as to increase the efficiency at the terminal while also benefitting the bandwidth requirements. A comparison with an emerging standard will also be included with comments on how it will apply to the objective of video on demand for mobile devices.

1. Introduction

We know that the concepts of MPEG-4 are fast catching up as they have been found to include several advantages that were not available before. For example, adaptation for bit rate and bandwidth adjustments may be made in real time by resorting to scalability. Also, video object based coding enable the endpoints of the communication link to choose which objects to transmit and display depending on their bandwidth and processing requirements and based on their relevance to the context. With MPEG-4, evolved the concept of binary format for scene representation. A video scene is composed of multiple components, for example, the background of a scene and the person in

the scene are two discrete components, as also is the furniture in the scene. Representing this information in binary format has an advantage because it takes up less bandwidth on the network. MPEG-4 also provides capabilities that allow update commands to be sent that modify the current scene description. For example, if the original video sent only contains the component that represented the person (due to low bandwidth), then when bandwidth is available, an update command can be sent to include the background as well. Individual components or objects are coded as separate streams and knit together into a meaningful presentation using the scene description language.

In such a generic scene description, there are flexibilities that make it bulkier, more resource intensive and less suitable for mobile use. This paper will explore some of these options and evaluate other standards to determine an optimal scene representation amenable to the limitations of the mobile world.

2. Mobile usage and limitations

As we place emphasis on the limitations of the mobile devices, let's elucidate on what these really are. Typically a mobile device referred to here is a hand held, portable device like a cellular phone or a PDA. These devices are coming into vogue because of the extended features they are providing for routine communications (paging service, web browsers, short messaging text, address book, schedulers etc). Infact, many of the high end devices are also capable of housing a full fledged development environment which allows the user to customise the device by either programming new components or by downloading readily available components. There have been several

frills added to the device, some of which are a digital camera, games for leisure, ability to play stored video etc, ability to receive and play low bit and frame rate video. Needless to say, the users are looking at the portable device as their companion in many of the routine life scenarios. However, just as there is no silver bullet, the portable device has its limitations. There is most definitely a limitation on screen size and thus resolution. Also the processors are not meant to be workhorses and the memory available for processing is limited to a few megabytes at the most. Apart from all these, the power requirements of a mobile device play a crucial role. Most of these can last for about 2-4 hours of normal service usage. And if the user is mobile, it is not expected that avenues to recharge the battery will be readily available.

Given these constraints (and ideally even without), it is important that the processing power of the device be “under utilized” as much as possible while still displaying the video at optimum quality. This leads us naturally to how the components of a video scene are knit together and interpreted. Also, we will examine which aspects of video can be dispensed with given the current state of technology and general applicability

Our long term goal is to deliver video to the mobile devices without loss in quality or features.

3. MPEG-4 and BIFS

MPEG-4 includes several features that are well suited for rate adaptation and low bit rate transmission. MPEG-4 provides for lower bit rates because it has more advanced temporal and spatial resolution techniques. MPEG-4 also provides for object based encoding that allows us to move away from rectangular frame encoding and transmission. This new capability gives rise to the concept of the scene description stream. This stream is transmitted along with the rest of the object streams and specifies how the objects are to be displayed and animated on the terminal. The scene description stream is also transmitted in a very efficient binary format and is called BIFS (Binary Format for Scenes). Since the streams are independent objects, it is natural that we would like to have the capability to add or remove certain objects from the scene. The reason could be that an object is no longer applicable, or that the bandwidth does not allow us to transmit it any more.

This done, the next logical step is the ability to be able to animate objects. For example, how do we show a person walking. This is handled by a separate animation stream which specifies continuous changes to an objects representation. The animation can also

be achieved through interpolators and route nodes but more about them later. Interactivity with the user is handled via the use of commands that can be sent back to the server or via the use of terminal based code that can respond to user requests and modify the representation accordingly.

The scene description is represented as a tree structure and indicates the inherent hierarchies between the nodes which are in fact the different audio-visual objects. An object is again a hierarchy of objects, traits or properties. Each object has a temporal and a spatial extent. In simple terms this is about when and where it is to be displayed. While it is being displayed, its local coordinates are transformed to the rendering coordinates as appropriate and as defined by the scene description and the nodes in the parent hierarchy.

While the actual content of the streams is huge factor in determining how video can be delivered, scene description is important because it can control the overhead in these cases.

4. Mobile use

Let us first take a look at what a typical mobile need for video can be. That will provide a perspective on what is really needed versus what can be considered as a “frill” and be kept for a time when the devices are more capable. We will examine this aspect by considering the following examples:

Consider a corporate environment where not all members of a conference can attend in person. They have a flexibility to dial in and so they can participate. However, typically in conferences, there are presentations and charts being shown and discussed. It is useful to get these to the “people on the phone”. Also, as speakers rotate, we may want to get the visual out to everyone. What is needed from a video scene composition perspective? We may not need a complex scene. Users typically will have the need (and capability) to rewind and fast forward in this scenario only when it is being transmitted from a storage server on request. As speakers change, the scenes will change to reflect the transitions.

Now let us examine the needs for user interactivity. Typically, videos are built so that the users can click on an object and another stream is added to the scene. In most cases, these new streams are being transmitted and received by the terminal but not being used until requested. From the perspective of a device which is already running low on resources, this can prove expensive. We need to ensure that (with limitations discussed later), a server command can be sent back on this user action requesting for this stream to be sent.

For example, if the user clicks on the globe rotating on the screen, the lower half of the scene will change to show details about the company offices around the world.

It is also being discussed that 3-D graphics may not be required for mobile video. Given the advances in technology, this may not remain a valid assumption. For examples, users would definitely like a globe or an image to appear on their screens as it should be. The screen size is growing because of developments of folding screens and voice activated sensors which are reducing the keyboard aspects to a minimum.

Coming to keyboard support, BIFS and MPEG-4 currently provide support in the form of the InputSensor node that provides feedback on events outside the scene. The TouchSensor node provides for mouse interaction. However, object selection by using voice navigation and keyboard is not yet fully developed because it needs support from the terminal hardware as well. It is desirable to add support for action that can be taken if a user presses a key (designated as shortcut) or on a voice command associated with the objects in the scene. For example, "show only background" can allow a conference scene to focus on the presentation being displayed on the board. Or as the scene changes, "show only active node" can result in only the speaker being displayed. All the other streams can not only be removed from the scene but may not be transmitted by the server at all thus saving bandwidth and terminal processing. When appropriate server commands can be sent which resume the stream relay from that point on. Given the reserved bandwidth on the wireless channels, this command channel may be fast enough to allow this interaction.

There is one problem in the above proposal when it comes to the broadcast scenario as in our video conferencing example above. Typically the server stream is being shared by multiple remote users. It's not possible for the server to transmit different kinds of video to different terminals. The terminal software should discard these streams if not required at that time. The server flexibility is desirable when the server is in control or if there are limited users of the stream.

An enhancement could be to refer to the input streams not only by their Object descriptors but by their class of nodes as well. For example, background stream, main stream (talking speaker) etc. This can be selected easily by the user through use of predefined keyboard or voice commands and will generate tremendous interest.

```
ClassDescriptor
{
    classDescriptorId 1 #signify background
```

```
ObjectDescriptor
{
    objectDescriptorID 10
    esdescr
    [
    ]
    .
    .
    .
}
```

Consider a case where I am transmitting the video from a casino in Las Vegas to my home in Miami. It will be useful for my family to switch to see the background when I am referring to it and then switch back to see me. Given the handheld resource constraints this might be great for my cellphone because now I can transmit more number of times without having to recharge the batteries.

These commands where a user requests only certain parts to be sent is presently controlled by object descriptors. Exactly the same functionality can be achieved in the current process by specifying an object that translates user clicks to mean that a stream should be added or deleted. However, what we need here is a way to specify that only the background needs to be transmitted because the video belongs to a known category. Note that this may not be possible in cases where the scene composition is unpredictable. In cases where the server does not recognize this command because an appropriate video content is not applicable, this can simply be ignored. In the example above, to request only the background, the user presses a key and the server starts sending only the background thus reducing bandwidth and processing.

```
ServerCommand
{
    eventIn          SFBool          trigger
    exposedField     SFBool          enable
    exposedField     MFString        url
    exposedField     SFString        command
}
```

Here the URL may refer to a class or a particular stream. Ofcourse, a ServerCommandRequest node is to be used as well but those details are not required here. The syntax looks like this:

```
class ServerCommandRequest(BIFSConfig cfg)
{
    bit(cfg.nodeIDbits) nodeID;
    SFString          command;
}
```

The nodeID mentioned above is the node that routed the trigger. In this special case, this node will be the sensor that tracks keyboard events or could be an

object on the video scene that prompts the user to select his options.

Thus we see that some of the time synchronization complexity may be possible to be removed by keeping the object interaction really simple by only relying wherever possible on server interaction or flexible server control. This kind of interaction avoids overhead by transmitting something only when required. The resulting saves are on bandwidth and on the terminal end which receives and interprets less traffic. This scheme also provides the control to the user which can be used as an optional recourse.

ServerCommand can also be used to tell the server that the video quality is below optimal and thus to remove a particular node. This functionality is available today by way of rate adaptation and scalability mechanisms but these mechanisms tend to compromise on the video quality without really taking the user considerations into account. Note that in this case the user gets to make a decision about what he needs to see. It's upto the user to not degrade the quality but to set the priorities of the streams himself. For example, in a stored-video-on-request scenario, the priorities of the user can be communicated to the server before the session starts. The default option can rest with the server based on preset object priorities of the video.

Let's look at an example where the above command structure could prove handy. In order to control the media playback(start, stop etc), we use the media control nodes. In a condition where the media is being relayed from a stored location on request, the user may want to restart or rewind the media being received. Typically the stream would have to be buffered or stored on the terminal in order to do these operations which is limited by the amount of needed for space and associated bookkeeping by the processors. However, if we can simply send a server command requesting the server to go back and retransmit that stream by the specified time interval, then the storage problem is resolved. To further optimize, it only makes sense (in our common scenarios) that the user would like to replay the entire video and not just a part of it while the rest of it is playing as usual!. So a simplified media control means a simple server command that goes to the server with this request. Note that this kind of control is only possible on a stored media. For live media, it becomes complex and resource intensive to display old media while storing the new portions coming in. A typical media control node looks like this:

```
MediaControl
{
    url ["10"]
```

```
loop TRUE
mediaStartTime 5
mediaStopTime 15
}
```

5. Scalable Vector Graphics

Let's now look at another standard that's being proposed. This is a variation of the Scalable Vector Graphics (SVG) model. Its variations are being proposed as SVG Tiny (for cellular) and SVG Basic (for PDAs).

The SVG is based on the XML standard and that means that is fully a text based, tag based language. Here are some of the advantages and disadvantages:

1. An obvious advantage is that the images need not be coded as per the terminal resolution. Since the decoder will use the fonts and resolution available at the terminal, a description of what needs to be done will suffice. For example, to show a line from point A to point B, the line element can be supplied with the relevant parameters. Compare this to a bit mapped image sent from the server which may not scale well.
2. As mentioned before SVG is an XML based standard. That means that specifying tags and then the values for every element and it's many attributes will take up valuable bandwidth and processing power both at the server and the client. For simple images, a piece of scene description is seen to take less space than JIF and PNG. However when it comes to describing complex images, the text format takes a lot of space. A compressed version takes less space but then we have the overhead of compression, decompression and then interpretation at the terminal.
3. SVG Tiny also does not support server interactivity at this time. It only supports animation. So in the case of our examples above the users will have very little control over what they want to see. That may be very big drawback.
4. The current SVG only supports 2-D images and graphics. Subjecting users to such a limitation may not be a good idea especially when hardware and communication developments are removing all such limitations on processing and power.
5. To include components like video etc, there is a capability in the SVG format to link to external sources like JPEG, mp3 etc. However, then the advantages of scalable graphics is lost and we are back to where we started from (albeit with a little more on our plates)

6. SVG was not originally meant for scene description. However, given the kind of scenarios we have discussed above, that may not be very complex to include although by trying to make it an all rounder we might lose its advantages of simplicity.
7. There is also a need to verify that the XML is legal and coherent by verifying it against the DTD specified for the communication. If the XML is complex, this process can be a large overhead. Because once the text is validated, it has to be read, interpreted and then the action carried out.
8. An advantage of SVG is that since it is a plain text language, it can be generated very easily from databases and from sources that just want to draw graphics on the fly. Compare that to binary encoding of the BIFS.

Size comparisons (in KB):

<i>File</i>	<i>SVG</i>	<i>GIF</i>	<i>JPEG</i>	<i>PNG</i>
Simple	2	26	24	21
Complex	96	52	56	111

Compressed complex scene for SVG was 32KB

Details of the line interface are as follows. This is the interface definition and SVG provides for several other shapes:

```
interface SVGLineElement :
    SVGElement,
    SVGTests,
    SVGLangSpace,
    SVGExternalResourcesRequired,
    SVGStylable,
    SVGTransformable,
    events::EventTarget
{
    readonly attribute SVGAnimatedLength x1;
    readonly attribute SVGAnimatedLength y1;
    readonly attribute SVGAnimatedLength x2;
    readonly attribute SVGAnimatedLength y2;
};
```

The attributes are explained as below:

SVGAnimatedLength x1

Corresponds to attribute x1 on the given 'line' element.

SVGAnimatedLength y1

Corresponds to attribute y1 on the given 'line' element.

SVGAnimatedLength x2

Corresponds to attribute x2 on the given 'line' element.

SVGAnimatedLength y2

Corresponds to attribute y2 on the given 'line' element.

What we have just discussed above is the way media is delivered using SVG. However, let's get back to the focus areas which is scene description. At this time SVG does not provide full fledged scene description although it may be apparent that by delivering graphics it is capable of placing them anywhere on the screen and provide basic animation and interactivity. However, this precludes real time streaming under broadcast scenarios. Also, animation is primarily declarative. SVG does provide the capability of loading a graphics on user interaction.

The latest working draft of SVG 1.2 adds certain features like support for streaming which provides for an element to start playing as soon as it's tag is read and processed. This flexibility makes it possible to start playing certain media even before the entire document is downloaded. Also, the specification provides another flexibility that the document can be discarded if it is not needed. This helps reduce DOM overhead. The specification has also not included a previously considered feature to separate the document order from the drawing order.

6. SMIL

SMIL 2.0 (synchronised MultiMedia Integration Language) is a standard developed by the W3C for expressing media synchronization among objects of different types.

SMIL provides for the common aspects like animation, timing control, transition effects and also supports keyboard input. As opposed to BIFS, SMIL is a text based scene description. That means that scenes are represented using a tagged format such as XML. SMIL has also been enhanced to be compliant with XHTML and that means that typically all internet browsers can interpret the scene description.

SMIL provides for screen layout control by defining regions and then specifying the media that needs to occupy the region and its timing attributes if any. Also one thing to keep in mind is that SMIL is only a container for scene representation. It contains only references to the media data and does not contain the media itself.

Following is a very popular example taken from an SVG tutorial on the web. This example may not be

very complex but it will provide a general idea of the major capabilities.

Let's use SMIL to make a simple scene which will display a "video in Video" type of an effect. The two video streams can be anything that the client browser will understand with respect to the software on it's system.



The code for the above looks like this:

```
<smil>
<head>
<layout>
<topLayout width="320px" height="240px">
<region id="main-video"
left="0%" top="0%" width="100%"
height="100%">
<region id="corner-video" left="67%"
top="67%" width="33%" height="33%"
fit="scale" soundLevel="0%"/>
</region> </topLayout> </layout>
</head>
<body>
<par>
<video id="chameleon-video"
src="chameleon.mpg"
alt="an animated chameleon"
region="main-video"/>
<video id="earthquake-video"
src="earthquake.mpg"
alt="San Fransisco earthquake aftermath"
region="corner-video"
end="chameleon-video.end"/>
</par> </body> </smil>
```

Let's consider another example where we show the transition capabilities of the scene description given a separate media object. This shows a clockwise transition of the image:

```
<html
xmlns:t="urn:schemas-microsoft-com:time">
<head>
<?import
```

```
namespace="t" implementation="#default#time2">
<style>t {behavior: url(#default#time2)}</style>
</head>
<body>
```

```
<t:transitionfilter
targetelement="keyb"type="
clockWipe" begin="keyb.begin" dur="2s" />

</body>
</html>
```

The above examples can directly be played and tested in your browsers without need to install any separate software. Here is how we play audio in SMIL:

```
<html
xmlns:t="urn:schemas-microsoft-com:time">
<head>
<?import
namespace="t" implementation="#default#time2">
</head>
<body>
<t:audio src="liar.wav"
repeatCount="indefinite" type="wav" />
</body>
</html>
```

Now lets display the same in the textual format of BIFS using MPEG4 media. We need to include the media as two separate elementary streams with their own object descriptors. An initial update command will load the two video streams. Note that we need software to convert these to the binary stream as well as MPEG4 software that will bundle the elementary streams together in a playable MPEG-4 format.

First we define the two areas where the elementary streams will reside.

```
OrderedGroup {
children
[
Shape {
appearance Appearance {
material Material2D {
filled true
}
texture MovieTexture {
url [ "10" ]
}
}
}
geometry Rectangle {
```

```

        size 80.0 80.0
    }
}
Shape {
    appearance Appearance {
        material Material2D {
            filled true
        }
        texture MovieTexture {
            url [ "11" ]
        }
    }
    geometry Rectangle {
        size 80.0 80.0
    }
}
]
}

```

Now declare the initial object descriptor (extra stuff removed for clarity).

```

InitialObjectDescriptor {
    objectDescriptorID 1
    esdescr [
        ES_Descriptor {
            es_id 2
        }
    ]
    decConfigDescr DecoderConfigDescriptor {
        streamType 3
        isCommandStream true
        pixelMetrics true
        pixelWidth 100
        pixelHeight 100
    }
}
ES_Descriptor {
    es_id 1
    decConfigDescr DecoderConfigDescriptor {
        streamType 1
    }
}
]
}

```

Now add the videos streams.

```

AT 0 {
    UPDATE OD [
        ObjectDescriptor {
            objectDescriptorID 10
            esdescr [
                ES_Descriptor {
                    es_id 3
                }
            ]
            muxInfo muxInfo {
                fileName "helloworld.mp4"
            }
        }
    ]
}

```

```

    }
]
}
UPDATE OD [
    ObjectDescriptor {
        objectDescriptorID 10
        esdescr [
            ES_Descriptor {
                es_id 3
            }
        ]
        muxInfo muxInfo {
            fileName "manish.mp4"
        }
    }
]
}
}

```

Representing BIFS in text format is a little too verbose and tedious.

7. A brief comparison

Here is a brief feature by feature presentation of BIFS vs SMIL. Note that the biggest difference of importance in our case is that SMIL does not have the capability to play live broadcast media as of now. 3-D objects are not supported but for mobile use, that may not prove to be critical.

<i>Feature</i>	<i>SMIL</i>	<i>BIFS</i>
Interactivity	Yes	Yes
Animation	Yes	Yes
Buffering	No	Yes
Streaming	No	Yes
2-D, 3-D description	Yes	Yes
3-D composition	No	Yes
Keyboard support	Yes	No
3-D sensor events	No	Yes
Loading from middle of a new presentation	Yes	No
Complexity of description	Low	High
Fomat	Textual	Binary

6. Standards in the making

There are several efforts under way to ensure development of standards that will allow efficient delivery of video to mobile devices. It is increasingly being assumed that BIFS is too complex to form the basis of this standard. Because of its VRML legacy features, it is being preferred to look for a new basis rather than scale BIFS down for these purposes. The MPEG committee has placed a call for proposals for a light weight scene description standard. This new standard is expected to have all the necessary features like animation, scalability, 2-D and 3-D support, save/restore scene states, interaction etc.

7. Summary

To summarize the above description, there are advantages and disadvantages of using a single standard alone. For example, SMIL does not have support for live streaming while BIFS does. BIFS does not allow you to load a new presentation in the middle while SMIL does. Also, SMIL looks to be simple enough for most common uses of a mobile user with the exception of live media streaming.

Also, the binary format of BIFS may be more network and terminal friendly when it comes to efficiency. SMIL proves to be more user friendly for editing purposes. SMIL is also now being coupled with HTML (XHTML) and can be interpreted directly by the client internet browsers while BIFS is a protocol that needs an explicit decoding module.

When it comes to mobile use however, we need a reality check. Using plain text and XML formats is a good thing, but it causes a tremendous amount of overhead at the receivers end because of the interpretation required. Also, when we talk about the user friendly scene description language, are we expecting everyone with a mobile phone to go about making their own presentations in a text editor. It is more realistic to believe that scene representation is best left to professional tools and techniques.

BIFS is a flexible and a very complex scene representation language and thus needs to be scaled down in order to be used for mobile devices. On the other hand, looking at a plain text format to replace it may not be correct. BIFS provides for enhanced user interactivity and live streaming media support both of which are essential for video delivery in the typical mobile users. An important factor for decisioning is the fact that SMIL does not support media that is being broadcast live.

On the other side of the spectrum is the SVG Tiny and SVG Basic standards. These are adaptations from

the original SVG standard and are intended for mobile video scene representation. However, it must be noted that SVG was originally meant for scalable 2-D graphics. It does not support audio and video and is being enhanced to support complex scene representation.

What we need to identify is our need. There are two completely different aspects of this topic. Scene description and the actual media delivery. BIFS and SMIL are primarily meant for scene description while SVG was originally defined for 2-D stationary graphics. SMIL can be played in any internet browser while both BIFS and SVG need specialized software or components to support their use.

Where we go from here will need a clear understanding of the differences without bias.

10. References

[1] <http://www.w3.org/TR/SVGMobile/>

[2] <http://www.w3.org/TR/SMIL/>

[3] http://gpac.sourceforge.net/tutorial/bifs_part4.htm

[4] <http://wdvl.internet.com/Authoring/Languages/XML/SVG>

[5] http://tft.fomo.us/en/tutorial/learning_to_smil/

[6] <https://www.alphaworks.ibm.com>