A Web-based Framework for Compressed 3D Objects: Downloading and Rendering

Umer IJAZ[1], Emanuele QUACCHIO[2] and Daniele ALFONSO[2]
[1]*Department of Electronics and Telecommunications, Politecnico di Torino, Italy*
[2]*STMicroelectronics, Italy*
umer.ijaz@polito.it, emanuele.quacchio@st.com, daniele.alfonso@st.com

Abstract— ***This paper is focused on proofing the concept of a web based receiver to download, decode and render 3D objects. It demonstrates the effectiveness in performances when compressed 3D objects are used. It also enlightens the advantages for downloading and rendering in web browsers at different compression environments. It revealed that web-based receiver was useful for downloading, decoding and rendering 3D objects. Downloading time of compressed and uncompressed files was dependent on internet speed while decoding time upon client processing power. The 3D compressed objects took less download/decoding time as compared to 3D uncompressed download time at low (512 Kbps) internet speed. At 2 Mbps, time was not dependent to file sizes. At 10 Mbps, the process was amazingly inversed as times taken by compressed files were more than uncompressed files. This performance evaluation confirmed effectiveness of using compressed 3D objects over uncompressed ones at low speed. The work can also be extended for dynamic 3D objects and animations in future.***

Keywords: **MPEG 4, JS, PhP, X3D, X3DOM, MPEG DASH.**

I. INTRODUCTION

Computer graphics technologies are being used today in many applications, from 3D games and immersive 3D environments for social networking, to 3D virtual world for medical and training services. There is an increase in demand of a unique receiver which can download, decode and render 3D objects, which leads to the development of a web-based receiver based on JavaScript (JS) and Hypertext Processor (PhP) to handle those tasks. JS function enforces download of 3D objects on client machine. Decoding was managed by MPEG 4 Part 25 (MP25) [1] compression tool, which was a 3D graphics compression standard that defined the compression on arbitrary eXtensible Markup Language (XML)-based scene representation [2]. Rendering was achieved through Extensible 3D Document (X3DOM) [3] which provided framework for integrating and managing Extensible 3D (X3D) [4] objects as Hyper Text Markup Language (HTML) applications. Both decoding and rendering were achieved by shell execution of MPEG 4 Part 25 and X3DOM in PhP handled by JS. Moreover, the network-based distribution of 3D graphics over internet was difficult, as amount of 3D data in such applications were usually very large, and processing was too complex. This lead to the development of using compressed 3D objects, which demonstrated effectiveness in performance compared to using uncompressed 3D objects.

The main scope of this research article is to proof the concept of a web-based receiver to download, decode and render 3D objects. It also demonstrates the effectiveness in performances when compressed 3D objects are used. The rest of the paper is organized as: Section II details the system components, Section III shows the system description, while Section IV is based on system evaluation. This work concludes with discussion on future work with acknowledgement and references to follow.

II. BACKGROUND / SYSTEM COMPONENTS

A. 3D Graphics Formats

XML based formats provide fast and accurate sharing of 3D data. They are easy to understand, having HTML like tags. They can be opened in a text editor. Most popular XML based formats were COLLAborative Design Activity (COLLADA) [5], X3D and eXtensible MPEG-4 Textual format (XMT). X3D is used for 3D data exchange in our system architecture. X3D is VRML successor, open source and more importantly international standard created by Web3D consortium. It is an interchange format and can be deployed on web. It is used for representing 3D scenes /objects plus user interaction. It provides run time model, real time delivery and flexible run time power. It is ideal for web applications and provides complete integration with web based formats like Asynchronous JavaScript and XML (AJAX) and Document Object Model (DOM) programming. It is supported by a wide range of authoring tools. Many open source X3D libraries are available online. Last but not the least X3D files can be very extensive but complex.

B. 3D Graphics Compression Tools

In order to reduce the size of X3D data prior to delivery over constrained networks, an open-source implementation of MPEG-4 Part 25 (MP25) [1] was used. MP25 implemented the encoding of XML-based scene graph formats (COLLADA, X3D and XMT) into an mp4 file. It also included decoder for demuxing and decoding the compressed mp4 data.

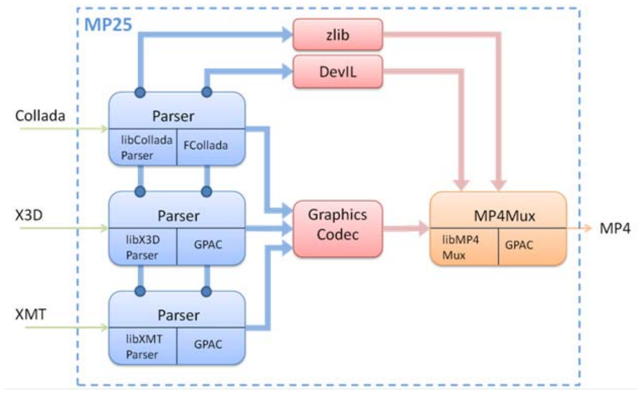

Figure 1: MP25 encoding structure

MP25 encoder is composed of three stages (i.e. parsing, compression and muxing) as depicted in Figure 1. In parsing, different layers of data (i.e. meshes, textures, animations and binarized data) are extracted from an XML file. This data is encoded using different tools during compression as follows:

I. 3D meshes are compressed using Scalable Complexity 3D Mesh Compression (i.e. MPEG4/SC3DMC [6]);
II. Animations using Bone-Based Animation (MPEG-4/BBA [6]);
III. Textures are loaded using an Image (DevIL [7]);
IV. Binarized data are compressed using common a zlib encoder [8].

Finally, all compressed components are muxed into an mp4 file.

The MP25 decoder is implemented using the reverse order. It recovers the original XML-based scene graph formats (COLLADA, X3D and XMT) from an mp4 file.

C. 3D Rendering Tools

Various 3D rendering tools were available for interactive web-based applications, but they lacked interpretability issues. X3DOM [3] was used to solve this issue. X3DOM provided open source framework for integrating and managing X3D objects as HTML applications. It allowed defining 3D-scene description and runtime behavior declaratively, without any low-level JS or OpenGL Shading Language (GLSL) coding. It provided style sheet library and supported various sound, image, movie formats etc. It allowed X3D as a run time format in browser applications. It supported 3D authoring tools by providing them X3D exporter's plugins. It allowed camera navigation in various modes i.e.; examine, walk, fly, look at, game. Its behavior was similar to HTML and provided various logs to monitor performance.

D. The MPEG DASH

In Hyper Text Transfer Protocol (HTTP) streaming the media content is fragmented in shorts size files (named segments or chunks) prior to start the content download. The encoded segments are hosted in an HTTP web server jointly with a manifest file, which contains information on location and format for the media fragment. A client requests the segments from the web server in a linear fashion and downloads them using plain HTTP progressive download. After download the client plays back the sequence of segments in linear order; the granularity of the content gives to users the possibility to watch it in any intermediate point without having to prior download the complete video. The "adaptive" part of the solution comes into play when the video/audio source is encoded at multiple bit rates. The client can then monitor playback related parameters and decide to change accordingly the average bit rate of the media. Adaptive HTTP streaming has already been widely deployed today in several commercial solutions (i.e. Apple HTTP live streaming), and in 2012 the MPEG committee released a standard aiming at defining an unified approach to the technology for MPEG media, known as Dynamic and Adaptive Streaming over HTTP (DASH), [10].

MPEG DASH primarily defines two kind of information:

I. The Media Presentation Description (MPD) [9],[10] which describes the media representations and resource identifiers (which are exclusively HTTP-URLs)

II. The segment format, which specifies the way in which the media content need to be prepared in order to reply to HTTP GET and partial HTTP GET requests issued by a client.

The MPD provides sufficient information for a client to access the streaming service; however, the DASH standard does not describe a normative behavior of the client once the MPD file is received. The DASH client model is depicted in Figure 2.

MPEG DASH currently supports efficient provision of live, on demand and time shift video streaming services over Internet Protocol (IP). Several profiles have been defined for MPEG based media format (International Standards Organization

Based and MPEG2 Transport Stream), enabling features like trick-modes, advertisement insertion, DRM etc.

Though focused on audio-video contents, DASH standard, and in particular the MPD manifest, can be used for the delivery of any multimedia format (2D, 3D, animation, graphics, text etc.), not only Audiovisual (AV).

In this work, the MPD manifest was adapted as defined in [2], in order to setup a streaming framework enabling the description, delivery and composition of 3D objects and scenes.

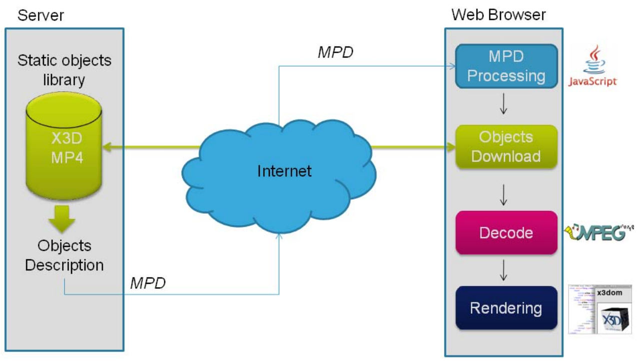

Figure 2: System Flow Diagram

III. SYSTEM DESCRIPTION

X3D and mp4 files were used for static 3D object description. X3D file were created in 3ds max and compressed to mp4 through MPEG part 25 tools. These mp4 files were placed along with MPD files at server. MPD files hold IP and other information of mp4 files. Client downloads MPD file. It extracts IP address and downloads mp4 files on its machine. Downloading was followed by the decoding and rendering process which were achieved through MPEG tool and X3DOM, respectively.

A. MPD usage for 3D objects description

MPEG DASH currently supports efficient provision of live, on demand and time shift video streaming services over IP. Several profiles have been defined for MPEG based media format (ISO Based and MPEG2TS), enabling features like trick-modes, advertisement insertion, Digital Rights Management (DRM) etc.

Though focused on audio-video contents, DASH standard, and in particular the MPD manifest, can be used for the delivery of any multimedia format (2D, 3D, animation, graphics, text etc.), not only AV. The basic structure of the DASH manifest is depicted in Figure 3.

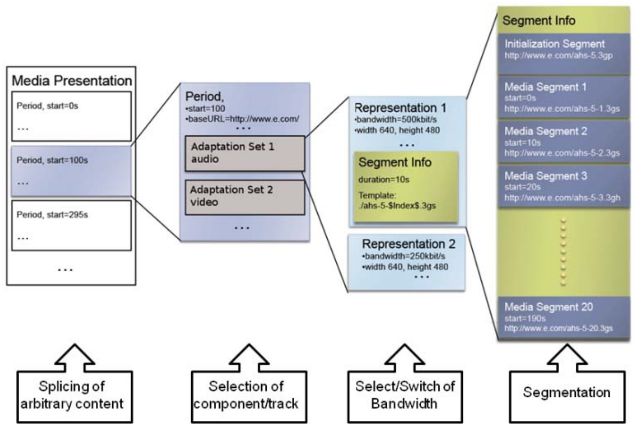

Figure 3: Structure of the DASH manifest

Some examples of MPEG DASH manifest were used as reference publicly available [11], which was customized in order to include some high level information on 3D objects. Such objects have been for simplicity grouped only in two categories, background and avatars, each of which is embodied in an adaptation set of the MPD, as shown in Figure 4.

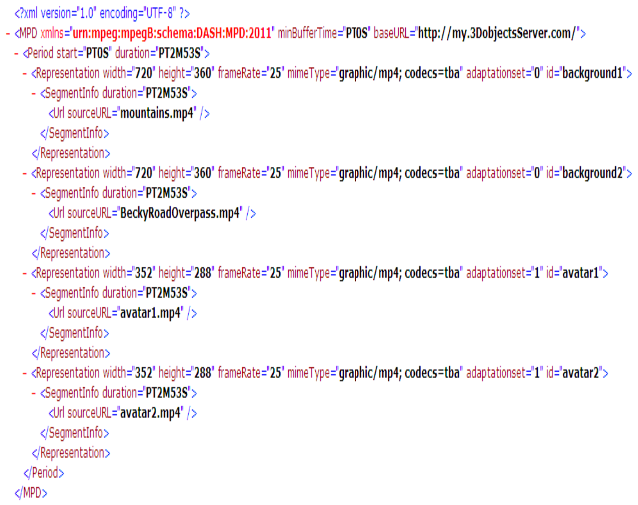
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <MPD xmlns="urn:mpeg:mpegB:schema:DASH:MPD:2011" minBufferTime="PT0S" baseURL="http://my.3DobjectsServer.com/">
  - <Period start="PT0S" duration="PT2M53S">
    - <Representation width="720" height="360" frameRate="25" mimeType="graphic/mp4; codecs=tba" adaptationset="0" id="background1">
      - <SegmentInfo duration="PT2M53S">
          <Url sourceURL="mountains.mp4" />
        </SegmentInfo>
      </Representation>
    - <Representation width="720" height="360" frameRate="25" mimeType="graphic/mp4; codecs=tba" adaptationset="0" id="background2">
      - <SegmentInfo duration="PT2M53S">
          <Url sourceURL="BeckyRoadOverpass.mp4" />
        </SegmentInfo>
      </Representation>
    - <Representation width="352" height="288" frameRate="25" mimeType="graphic/mp4; codecs=tba" adaptationset="1" id="avatar1">
      - <SegmentInfo duration="PT2M53S">
          <Url sourceURL="avatar1.mp4" />
        </SegmentInfo>
      </Representation>
    - <Representation width="352" height="288" frameRate="25" mimeType="graphic/mp4; codecs=tba" adaptationset="1" id="avatar2">
      - <SegmentInfo duration="PT2M53S">
          <Url sourceURL="avatar2.mp4" />
        </SegmentInfo>
      </Representation>
    </Period>
  </MPD>
```

Figure 4: MPD File Structure

The content is not fragmented, and each segment corresponds to the whole file. Ad-hoc and not-standard mime and codec type have been defined.

B. MPD Parsing

JS does MPD parsing by making HTTP request to access specified XML file. It opens XML file and stores HTTP response in a variable. The variable is filtered for various tags to fetch the required information. This information is displayed in the web browser at the client side.

C. Objects Downloading

Object downloading is done through a force downloading JS function, which downloads file on client machine. This downloading is network dependent and is directly related to internet speed.

D. Decoding and Rendering Integration

MPEG part25 tool and X3DOM were used for decoding and rendering, respectively. There integration was achieved through a combination of JS and PhP. JS imported "JQuery libraries" and a "request JS" to process specified PhP script, which does mpeg4 to X3D decoding and responds by playing X3D file on client browser with X3DOM.

IV. System Evaluation

The experiment was performed with two notebooks, one of which was server and other was client. Both notebooks (Intel Core i3 i3-M350/2.27GHz, 4GB RAM) were running Linux operating system with client using mozilla firefox 4 web-browser. The two notebooks were connected to internet at different speeds (i.e. 512Kbps, 2Mbps and 10 Mbps). Figure 5(a), (b) and (c) show comparison of downloading plus decoding time of 3D compressed with the download time of uncompressed at different internet speeds. The sizes of compressed file mountains, BeckyRoadOverpass, avatar1 and avatar2 are of 1.2KB, 1.7KB, 1.5MB and 1.4MB respectively. While, uncompressed files mountains, BeckyRoadOverpass, avatar1 and avatar2 size are of 256.5KB, 3.2KB, 50.3MB and 55MB respectively.

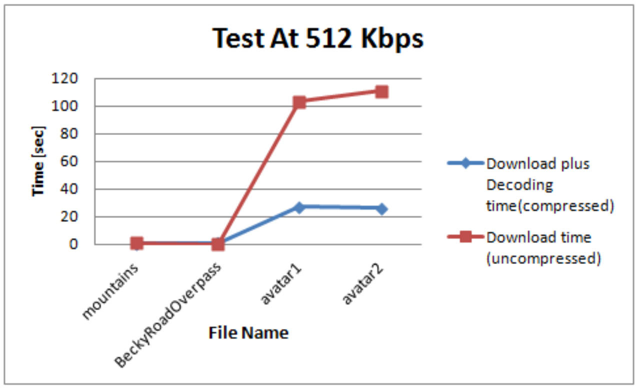

Figure 5(a): Test at 512Kbps

At 512 Kbps: Time taken by compressed and uncompressed, mountains and BeckRoadOverpass files, were almost same. While, time taken by compressed and uncompressed avatar1 files were 27.11seconds and 103.25seconds respectively, as shown in Figure 5(a)

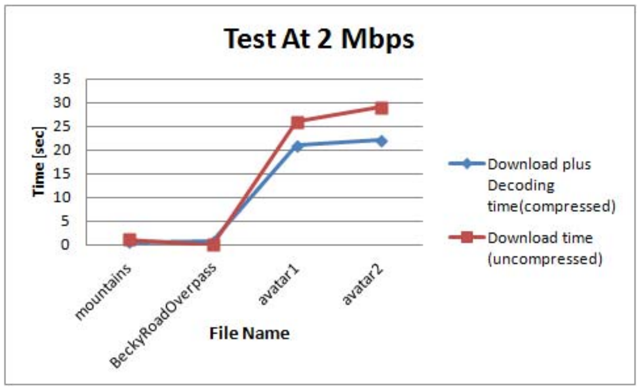

Figure 5(b): Test at 2Mbps

At 2 Mbps: Time taken by compressed and uncompressed, mountains and BeckRoadOverpass files, were almost same again. While, time taken by compressed and uncompressed avatar1 files were 21.18seconds and 26.09seconds respectively, as shown in Figure 5(b)

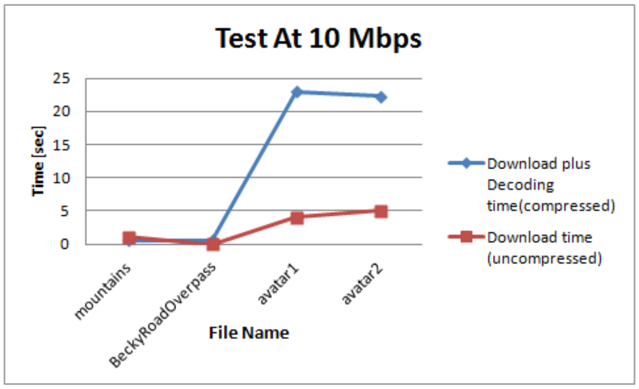

Figure 5(c): Test at 10Mbps

At 10 Mbps: Time taken by compressed and uncompressed, mountains and BeckRoadOverpass files, were almost same also. While, time taken by compressed and uncompressed avatar1 files were 23.11seconds and 4.23seconds respectively, as shown in Figure 5(c)

V. Conclusions And Future Work

It was concluded that web-based receiver was useful for downloading, decoding and rendering 3D objects. Downloading time of compressed and uncompressed files was dependent upon internet speed while decoding time depended upon client processing power. The 3D compressed objects took less download/decoding time as compared to 3D uncompressed download time at low (512 Kbps) internet speed. At 2 Mbps, time was not dependent to file sizes as time taken for both were almost same. At 10 Mbps, the process was amazingly inversed as time taken by compressed files, were more than uncompressed files. This performance evaluation confirmed effectiveness of using compressed 3D objects over uncompressed ones at low speed. The work can also be extended for dynamic 3D objects and animations in future.

ACKNOWLEDGMENT

The authors are thankful to REal and Virtual Engagement in Realistic Immersive Environments (REVERIE) for providing the test files in this work.

REFERENCES

[1] Le Bonhomme, B., Preda, M., and Preteux, F. 3D compression benchmarking with mymultimediaworld.com. In *3DTV Conference: The True Vision- Capture, Transmission and Display of 3D Video*, pages 105 –108, 2008.

[2] ISO/IEC . Iso/iec 14496-25:2011. Information technology coding of audio-visual objects – part 25: 3d graphics compression model, edition 2. Technical report, ISO, 2011.

[3] Behr, Johannes; Eschler, Peter; Jung, Yvonne; Zöllner, Michael. X3DOM - a DOM-based HTML5 / X3D integration model, 2009.

[4] Iso/iec 19775-1:2008, x3d (extensible 3d), part 1, architecture and base components, edition 2. Technical report, Web3D Consortium and ISO/IEC JTC1/SC24, 2008.

[5] Mark Barnes and Ellen Levy Finch, Sony Computer Entertainment Inc. Collada - digital asset schema release 1.5.0 specification. Technical report, Khronos group, 2008.

[6] GRIN, ARTEMIS, TELECOM Sudparis. Mymultimediaworld An MPEG-4 and MPEG-7 multimedia platform, http://www.mymultimediaworld.com/software/opensource/gc/, online, retrieved on 5 July 2007.

[7] Miwi. DevIL–A full featured cross-platform image library, http://www.openil.sourceforge.net/download.php, online, retrieved on 9 October 2007.

[8] Jean-loup Gailly and Mark Adler. Zlib-A massively spiffy yet delicately unobtrusive compression library, http://www.zlib.net/, online, retrieved on 2 May 2012.

[9] Thomas Stockhammer, Per Fröjdh, Iraj Sodagar and Sungryeul Rhyu. Text of ISO/IEC 23001-6: Dynamic adaptive streaming over HTTP (DASH), Guangzhou, China, Oct 2010

[10] Christopher Müller & Christian Timmerer. A test-bed for the dynamic adaptive streaming over HTTP featuring session mobility. Klagenfurt University, ITEC 2010-02-25

[11] Klagenfurt UNI. MPEG DASH data set, http://www-itec.uni- klu.ac.at/ftp/datasets/mmsys12/ , online, retrieved on 9 October 2012.