An Implementation of a DASH Client for Browsing Networked Virtual Environment

Thomas Forgione, Axel Carlier, Géraldine Morin, Wei Tsang Ooi, Vincent Charvillat, Praveen Kumar Yadav

An Implementation of a DASH Client for Browsing Networked Virtual Environment

Thomas Forgione
Université de Toulouse - IRIT
thomas.forgione@irit.fr

Axel Carlier
Université de Toulouse - IRIT
axel.carlier@enseeiht.fr

Géraldine Morin
Université de Toulouse - IRIT
morin@enseeiht.fr

Wei Tsang Ooi
National Univ. of Singapore
weitsang@nus.edu.sg

Vincent Charvillat
Université de Toulouse - IRIT
charvi@enseeiht.fr

Praveen Kumar Yadav
National Univ. of Singapore
praveen@comp.nus.edu.sg

ABSTRACT

We demonstrate the use of DASH, a widely-deployed standard for streaming video content, for streaming 3D content in an NVE (Networked Virtual Environment) consisting of 3D geometry and associated textures. We have developed a DASH client for NVE to show how NVE benefits from the advantages of DASH: it offers a scalable, easy-to-deploy 3D streaming framework. In our system, the 3D content is first statically partitioned into compliant DASH data, and metadata is provided in order for the client to manage which data to download. Based on a proposed utility metric for geometry and texture at the different resolution, the client can choose the content to request depending on its viewpoint. We effectively provide a Web-based client to navigate through our sample 3D scene, while deriving the streaming requests from its computation of the necessary online parameters, in a receiver-driven manner.

ACM Reference Format:
Thomas Forgione, Axel Carlier, Géraldine Morin, Wei Tsang Ooi, Vincent Charvillat, and Praveen Kumar Yadav. 2018. An Implementation of a DASH Client, for Browsing Networked Virtual Environment. In *2018 ACM Multimedia Conference (MM '18), October 22–26, 2018, Seoul, Republic of Korea.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3240508.3241398

1 INTRODUCTION

Dynamic Adaptive Streaming over HTTP (DASH), or MPEG-DASH, is now a widely deployed international standard for streaming adaptive video content on the Web. The merit of DASH is its simplicity and its scalability. DASH uses the existing World Wide Web infrastructure and protocols, and so provides a scalable, easy to deploy video streaming framework in an HTTP supported network. In this technical demonstration, we demonstrate how DASH can be used for another application and media type — a freely navigable, networked virtual environment (NVE) consisting of 3D meshes with textures.

Zampoglou et al. have been the first to propose using DASH to stream 3D content [3]. The authors organize the content, following DASH terminology, into a scene graph containing multiple resolutions for each model of the scene. Their approach targets several objects but does not handle the case of large NVEs, for which view-dependent streaming is desirable.

In [2], view-dependent 3D streaming has been investigated and uses frustum and back-face culling by the server to decide the set of polygons to be streamed to a client. The drawback of this method is its lack of scalability.

Using the DASH methodology, we are able to reorganize the 3D data into DASH compliant segments and provide metadata, so that the client computes the necessary parameters to determine which parts of the content it should download. The server is only used to store data segments and related information.

2 CONTENT PREPARATION

This section details the DASH compliant organization of the 3D data modeling the NVE, that is, a polygon soup and texture files, and the associated metadata. This corresponds to the offline computation of the 3D content stored in the server.

The MPD file. The Media Presentation Description (MPD) file is an XML file defined in DASH to provide global information about the content storage and access. In our 3D NVE use case, this information is based on spatial location, content resolution, or size. In the next paragraphs, we describe the organization of the 3D data to be streamed, and the corresponding MPD.

Geometry management. We spatially partition the faces of the scene using a k-d tree and further divide each cell into segments of a fixed number of faces, grouping faces of comparable area. Each segment is stored in a `.obj` file. We encode in the MPD the coordinates of each cell's bounding box, as well as the total 3D area of their constituting polygons and the size (in bytes) of their associated OBJ files.

The uncommonly large faces are stored in a separate segment (e.g., the water in figure 1), since they are essential to the model and do not fit into cells.

Texture management. Each texture of the model is stored in multiple PNG files at different resolutions. The MPD contains the size in bytes of each image file, the number of faces that use this texture, as well as the average color of each texture so that a client can render the corresponding faces with a uniform color when the

https://doi.org/10.1145/3240508.3241398

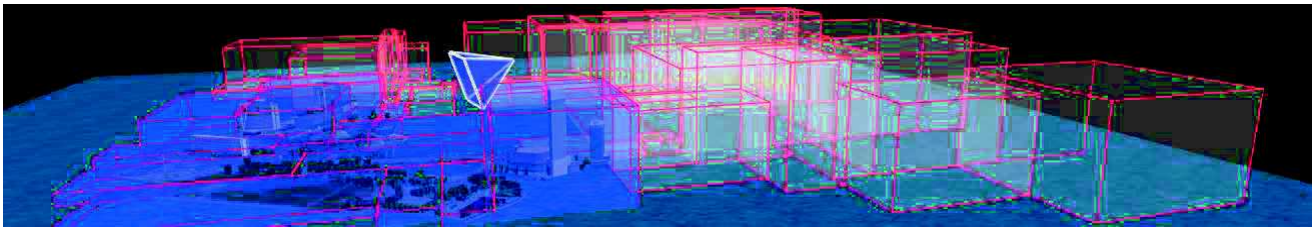

Figure 1: The sample 3D scene and an arbitrary viewport, with partitioning bounding boxes delimited with red edges. In white, the regions that fall outside the field of view of the camera; in blue, the regions inside the field of view of the camera.

texture has not been loaded yet. We also encode the MSE between the texture and its highest available resolution, so that a client is able to tell how important it is.

3 DASH 3D CLIENT

Once the 3D content is organized as described in the last section, we simply put all this data on a regular static HTTP server. In this section, we describe the implemented greedy DASH NVE client that exploits the preparation of the 3D content in an NVE for streaming.

3.1 Implementation of the client

We propose a Web-based client that uses XML HTTP requests to download content. The client takes into account some known parameters (e.g., position, the field of view, etc.) to decide which segment to request next. When a segment is received, it is added to a scene that is rendered using `Three.js`, and another one is queried. Since we run the server locally, in order to have realistic results, we configure the local network to limit the bandwidth to 5 Mbps.

3.2 Streaming policy

The first data fetched by our client is the MPD file. Once this file is retrieved, the client is able to identify the segment containing the large faces (mentioned in Section 2) and to understand the k-d tree structure of the scene. It downloads the outlying faces as well as the corresponding textures. Based on its current viewpoint, the client computes a utility for each segment, in order to decide whether it should download geometry or texture, and which resolution to download. The segment s^* to be downloaded is the one that optimizes the criterion:

$$s^* = \underset{s \in S}{\operatorname{argmax}} \frac{\texttt{utility}(s)}{\texttt{size}(s)}.$$

The client computes the intersection between the k-d tree BB (bounding boxes), its frustum, and sets the utility of a BB outside the frustum to 0. Otherwise, utility is proportional to the total 3D area of the faces that it contains (MPD information) and inversely proportional to the squared distance between the camera and the center of the corresponding bounding box (computed online by the client). The client weight the utility of a texture based on the utilities of the received geometry segments that refer to it. More information is available in [1].

3.3 User interface

For browsing through the scene, we implemented classical interactions in 3D navigation. First, the pointer lock API allows the user to turn the camera just like in a first-person view video game. The user moves the camera with the keys W, A, S and D. Unlike in video games, this interface offers no physical simulation so the user can fly around the scene. The height also increases or decreases using the mouse wheel. Pressing the space key increases the speed of the camera, which is useful to move quickly to another part of the scene; releasing this key allows more precise motion.

3.4 Rendering optimization

For rendering, we limit the number of WebGL calls: we keep faces that share the same texture in the same vertex buffer. The MPD indicates how many faces are needed for each texture; thus the client can allocate the exact size for the vertex buffers. When more faces are received, the preallocated vertex buffer corresponding to the right texture is just filled. Thus the client benefits from the MPD information to avoid successive memory allocation and copy and to keep only one face buffer per texture. The textures are initialized by an image containing a single pixel of the average color, and whenever a higher resolution image arrives, the client updates the texture.

4 DEMONSTRATION

The demonstration will display our DASH client, described in Section 3, and allow users to navigate in a 3D model of the Marina Bay area in Singapore (552K faces stored on 67MB, 161.6MB of textures). In particular, it will demonstrate streaming dependency to the viewpoint, e.g., the effect of camera motion, such as heavy rotation. A video presenting this demonstration can be seen at the following address: https://youtu.be/tGruvbs_H0g

REFERENCES

[1] Thomas Forgione, , Axel Carlier, Géraldine Morin, Wei Tsang Ooi, Vincent Charvillat, and Praveen Kumar Yadav. 2018. DASH for 3D Networked Virtual Environment. In *2018 ACM Multimedia Conference (MM '18), October 22–26, 2018, Seoul, Republic of Korea.* Séoul, South Korea. https://doi.org/10.1145/3240508.3240701

[2] Thomas Forgione, Axel Carlier, Géraldine Morin, Wei Tsang Ooi, and Vincent Charvillat. 2016. Impact of 3D Bookmarks on Navigation and Streaming in a Networked Virtual Environment. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16).* ACM, Klagenfurt, Austria, Article 9, 10 pages. https://doi.org/10.1145/2910017.2910607

[3] Markos Zampoglou, Kostas Kapetanakis, Andreas Stamoulias, Athanasios G. Malamos, and Spyros Panagiotakis. 2016. Adaptive streaming of complex Web 3D scenes based on the MPEG-DASH standard. *Multimedia Tools and Applications* (Dec 2016), 1–24. https://doi.org/10.1007/s11042-016-4255-8