International Journal of Internet, Broadcasting and Communication Vol.11 No.2 31-42 (2019)
http://dx.doi.org/10.7236/IJIBC.2019.11.2.31

IJIBC 19-2-5

Performance Comparison of 3D File Formats on a Mobile Web Browser

Duckkyoun Nam[1], Daehyeon Lee[2], Seunghyun Lee[3], Soonchul Kwon[2*]

[1]Department of Holography 3D Contents, Kwangwoon University, Seoul, Korea
[2]Graduate School of Smart Convergence, Kwangwoon University, Seoul, Korea*
[3]Ingenium College, Kwangwoon University, Seoul, Korea
*dknam@sk.com, {dleogus13813, shlee, *ksc0226}@kw.ac.kr*

Abstract

As smartphone H/W performance and mobile communication service have been enhanced, large-capacity 3D modeling files are available in smartphones. Common formats of 3D modeling files include STL (STereoLithography), OBJ (Wavefront file format specification), FBX (Filmbox), and glTF (open GL Transmission Format). Each format has different characteristics depending on the configuration and functions, and formats that are supported are varied depending on the applications. Large-size files are commonly used. The 4th generation mobile communication network secures loading of 3D modeling files and transmission of large-size geometric files in order to provide augmented reality services via smartphones. This paper explains the concepts and characteristics of major 3D file formats such as OBJ, FBX, and glTF. In addition, it compares their performance in a wired web with that in the 4th generation mobile communication network. The loading time and packet transmission in each 3D format are also measured by means of different mobile web browsers (Google Chrome and MS Edge). The experiment result shows that glTF demonstrated the most efficient performance while the loading time of OBJ was relatively excessive. Findings of this study can be utilized in selecting specific 3D file formats for rendering time reduction depending on the mobile web environments.

Keywords: *3D Modeling, glTF, Comparison, Quantitative data, efficiency, Mobile WebGL*

1. Introduction

As the advancement of information and communication technology is accelerating recently, 3D objects are drawing keen attention.Particularly with the development of smartphones and wireless networks, it is possible to operate 3D objects in a mobile web by means of the high-speed/large-size data transmission technology as the 4th generation mobile communication service is available. Common formats of 3D modeling files include STL (STereoLithography), OBJ (Wavefront file format specification), FBX (Filmbox), and glTF (open GL Transmission Format). File sizes and characteristics are varied depending on each 3D file format since the development purposes are different [1-3]. Accordingly, the present study aims to compare their loading

Manuscript Received: Feb. 27, 2019 / Revised: Mar. 6, 2019 / Accepted: Mar. 13, 2019
Corresponding Author: ksc0226@kw.ac.kr
Tel: +82-2-940-8637, Fax: +82-50-4174-3258
Graduate School of Smart Convergence, Kwangwoon University, Korea

performances in a mobile web.

This paper consists of the following sections: Chapter 2 examines OpenGL Graphics operable in mobile devices, basic concepts of 3D file formats such as STL, OBJ, FBX, and glTF, and the generation and processing procedures of such 3D file formats. Chapter 3 compares through experiments the 3D model data loading time when existing 3D file formats such as STL, OBJ, and FBX are used with that when glTF file format is used. Chapter 4 compares characteristics of 3D file formats comprehensively and analyzes the difference in loading rates among these formats. Chapter 5 presents conclusions based on experiment results as well as future research plans.

2. Background Theory

2.1 OpenGL Graphics

Figure 1 shows the procedures of processing various types of 3D data stored in a web server by means of WebGL (OpenGL ES 2.0) installed in a mobile terminal. WebGL is executed by the GPU of a mobile device. Codes executable in the GPU need to be prepared, and these codes are provided in a function pair. Each of the pair is called Vertex Shader and Fragment Shader. They are designed in GLSL (Graphics Library Shader Language) which is a language as strict as C/C++. These two as a set are called Shader Program. Vertex Shader calculates Vertex locations. Depending on the output location, WebGL can rasterize various types of primitives such as dot, line, and triangle. As these primitives are rasterized, Fragment Shader function is called on in the second step. Fragment Shader calculates colors of every pixel of the current primitives. Every data set that functions should access need to be provided to the GPU. Shader may receive data in any of the following 4 ways: Attribute and Buffer (location, normal line, texture coordinate, peak color), Uniform (global variable), Texture (image data), and Varying (rendering). Finally, Framebuffer converts memory bit maps into video signals that are displayed on the terminal monitor [4].

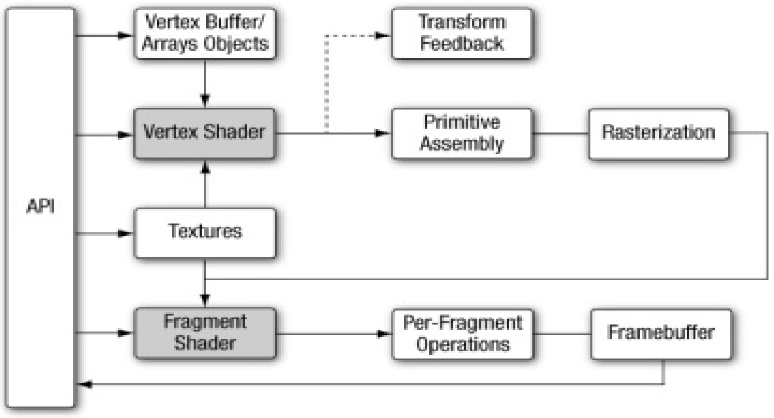


Figure 1. OpenGL ES 2.0 Graphics Pipeline [5]

2.2 3D File Format Attribute

Table 1 shows characteristics of each 3D file format. glTF supports important functions such as color setting, animation, CSG, detailed mesh work, texturing, camera, lighting, and relative position. CSG, which stands for Constructive Solid Geometry, utilizes boolean operators. CSG combines simple objects so that a modeler can design a complicated surface or object.

STL, OBJ, and FBX reflects limited characteristics. STL supports brief modeling functions of the geometry, omitting above-stated important functions such as color setting, animation, CSG, detailed mesh work, texturing, camera, lighting, and relative position. OBJ supports brief modeling functions, CSG, color setting, and materials. As mentioned earlier, OBJ utilizes a separate material file called MTL, and thus the MTL file that specifies data on texture maps and materials need to be transferred in addition to model information [6]. FBX does not support detailed mesh work functions.

Table 1. Attributes by 3D File Type [7]

File Type	Geometry		Appearance					Scene		Animation
	Brief Mesh	Detailed Mesh	CSG	Color	Material	Texture	Camera	Light	Relative Position	
STL	O	X	X	X	X	X	X	X	X	X
OBJ	O	X	O	O	O	X	X	X	X	X
FBX	O	X	O	O	O	O	O	O	O	O
glTF	O	O	O	O	O	O	O	O	O	O

2.3 glTF Import and Converter Structure

Figure 2 shows the relation between the importer and converter in the context of glTF generation and processing. 3D file formats such as STL, OBJ, and FBX need to go through the process of conversion into 3D modeling data specifically for compatible applications in order to be embodied as a browser by means of Graphics APIs [8].

It is impossible to apply files directly to Graphic APIs such as WebGL right after they are generated by means of authoring tool software programs such as Blender and Maya. 3D model data needs to be converted again by means of the Importer and Converter according to the Runtime applications. The process of conversion by means of the Importer and Converter may cause inconvenience to users in terms of time and cost efficiency. glTF removes the need for a separate Importer and the process of conversion. Once a 3D model is defined by means of glTF, the compatibility is secured for most applications where Graphics APIs are to be executed.

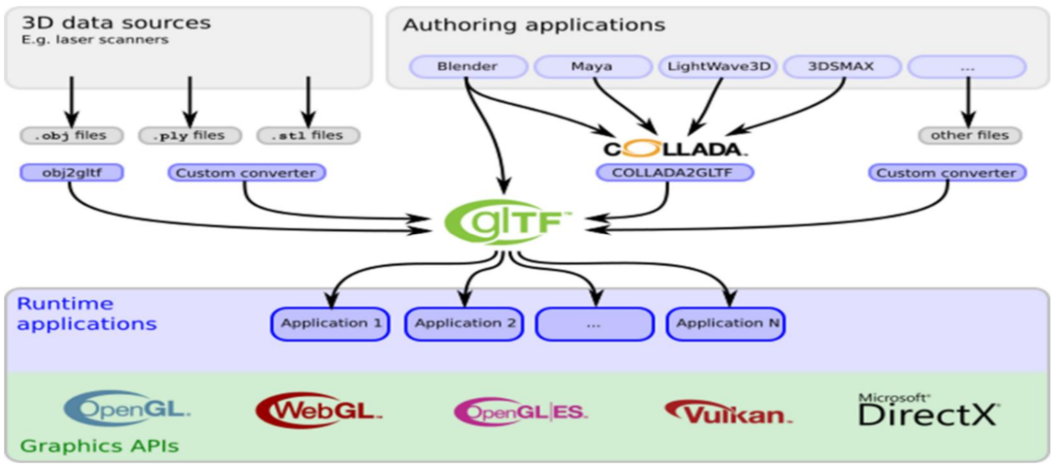


Figure 2. Import and Convert of glTF [9]

3. Experimental Environments

For experiments, files were generated in each format of STL, OBJ, FBX, and glTF by means of the Blender. Each 3D file format was rendered to the mobile web, and quantitative data was extracted depending on the loading time. OBJ was loaded onto the mobile web browser by means of OBJ_Loader.js and MTL_Loader.js. STL was rendered to the mobile web browser by means of STL_Loader.js. For FBX, FBX_Loader.js was utilized. glTF could be rendered to the mobile web by means of GLTF_Loader.js. In order to secure objective standards for comparison, the step of uploading through the Loader was omitted. A website was designed in a hosting server, and then 3D objects shown in Figure 3 were uploaded to it and downloaded to a mobile device.

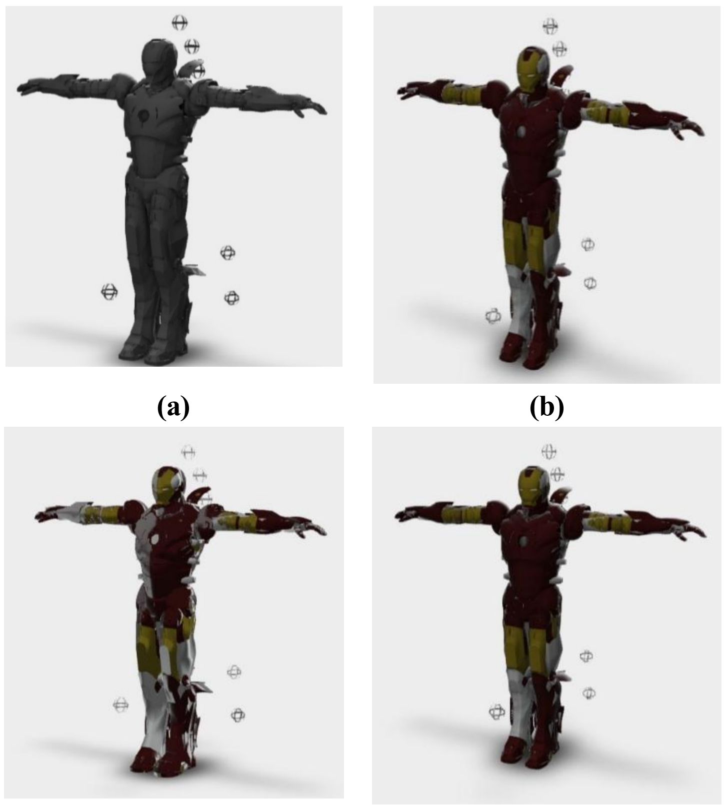


Figure 3. 3D Model Example
(a) STL (Verts: 104,963, Faces: 210,424, Tris: 210,424)
(b) OBJ (Verts: 129,757, Faces: 149,827, Tris: 217,038)
(c) FBX (Verts: 129,757, Faces: 149,827, Tris: 217,038)
(d) glTF (Verts: 131,044, Faces: 217,050, Tris: 217,050)

Figure 4 illustrates the system configured in order to measure the time of rendering 3D model data on the web by means of a development tool available in Google Chrome and Microsoft Edge. The development tool shows the time of rendering to the mobile web comparatively. Major panels often used for debugging include Elements, Console, Network, and Sources. In experiments, the Network panel was utilized to measure the loading time of 3D model data.

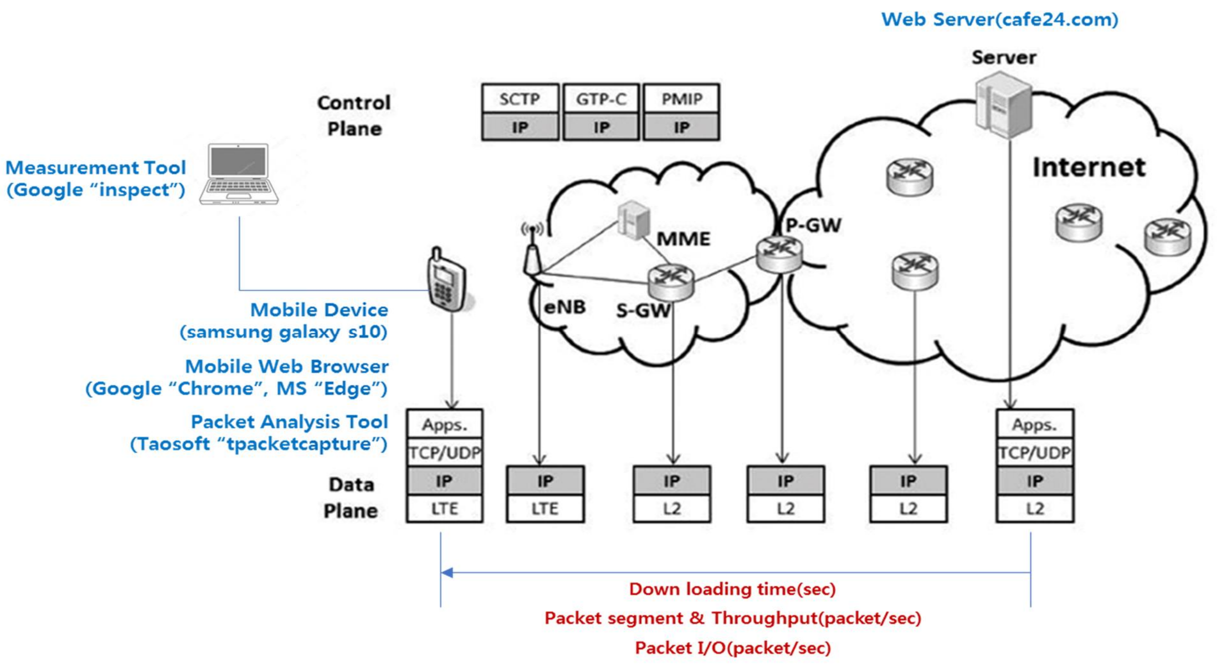

Web Server(cafe24.com)
Server
Control Plane
SCTP
GTP-C
PMIP
IP
IP
IP
Measurement Tool (Google "inspect")
Internet
MME
P-GW
eNB
S-GW
Mobile Device (samsung galaxy s10)
Mobile Web Browser (Google "Chrome", MS "Edge")
Packet Analysis Tool (Taosoft "tpacketcapture")
Data Plane
Apps.
TCP/UDP
IP
LTE
IP
LTE
IP
L2
IP
L2
IP
L2
Apps.
TCP/UDP
IP
L2
Down loading time(sec)
Packet segment & Throughput(packet/sec)
Packet I/O(packet/sec)

Figure 4. Experimental System Configuration

Table 2 shows the browser development tool, experiment site, mobile WebBrowser/WebGL, hardware performance, OS, and packet measuring program that were used in experiments.

Table 2. Experiment Environment

Classification	Information
Measurement Tool	Google Development Tool "inspect"
	Taosoftware "tpacketcapture"
Experimental Site	http://kwonlab.or.kr
Used Browsers	Google "Chrome", Microsoft "Edge"
Mobile Hardware	SM-G977N (Samsung gallaxy S10)
Mobile OS	Android 9
Analysis Program	Dump cap (wireshark) 2.6.4

The wireless internet communication conditions between the mobile device and web server were measured 10 times in the same place. Table 3 shows 3 quality items, the average value of each is as follows: that of Download Throughput was 245Mbps; that of Upload Throughput was 32.2 Mbps; and that of Ping (Latency) was 29.9 msec.

Table 3. Wireless Internet Network Quality at the Experiment Point

No.	Download	Upload	Latency
Average	245	32.2	29.9
1	227	43.4	36.2
2	244	40.9	30.8
3	244	38.6	30.4
4	238	27.0	31.2
5	242	26.8	29.0
6	254	26.1	28.4
7	231	24.7	28.3
8	225	25.1	29.2
9	291	25.8	27.9
10	251	43.4	28.0

4. Experiment and Result

4.1. Performance Comparison on the Mobile Web

3D model data loading was performed by means of Google Chrome and Microsoft Edge in a Samsung galaxy s10 (SM-G977N) terminal. 3D model data was converted into each of the 3D file formats: glTF, OBJ, FBX, and STL. The loading time was measured based on the three standards: DOMcontentloaded, load, and Finished [10]. The DOMcontentloaded event indicates the timing when the DOM tree was completed but an external resource (img etc.) had yet to be loaded. The load event indicates the time when every resource (img, style, script etc.) was loaded onto the browser. The Finish event indicates that the time until 3D model data loading started and ended after every source was downloaded.

Table 4 shows in a table the loading time in Google Chrome and MS Edge. This shows that the 3D model data loading time of glTF was shorter than that of OBJ, FBX, and STL. The loading time was in order of FBX, STL and OBJ. After 10 repetitions of the test, the standard deviation of glTF was the lowest, which indicates that its loading characteristics were stable in general.

Table 4. Browser-specific loading time (msec)

	Browser	Domcontent loaded	Load	Finished	SD_finished
Chrome	STL	784	783	8,039	2,150
	OBJ	804	802	10,926	3,560
	FBX	967	966	4,844	**769**
	glTF	853	850	**4,740**	788
Edge	STL	763	762	5,986	1,193
	OBJ	804	802	8,061	2,069
	FBX	877	874	4,439	**512**
	glTF	880	877	**3,877**	785

Table 5 shows in a graph the loading time in Google Chrome and MS Edge. As indicated by this data, the loading time of OBJ is shorter than that of STL, FBX, and glTF in general.

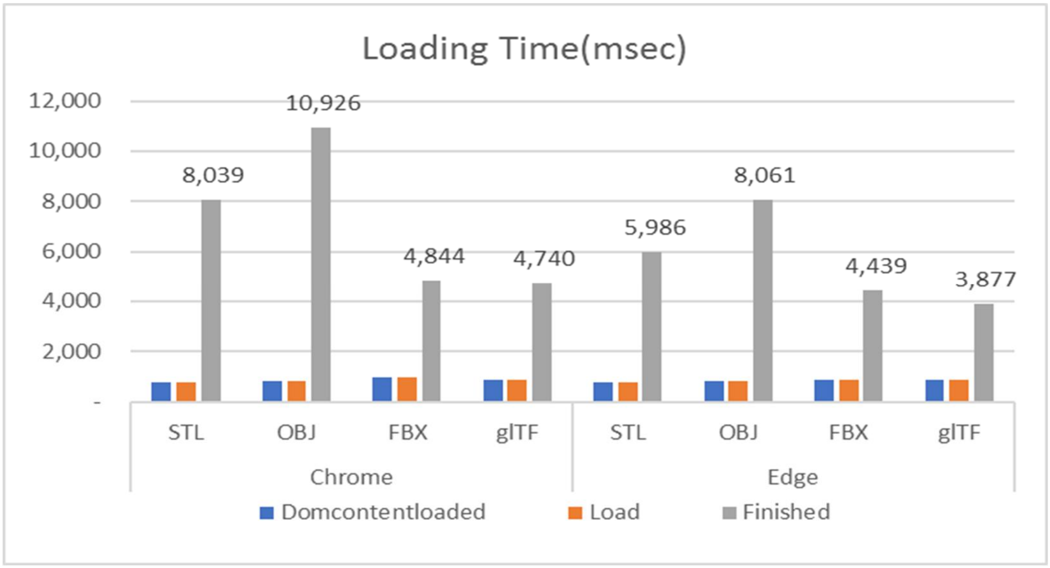
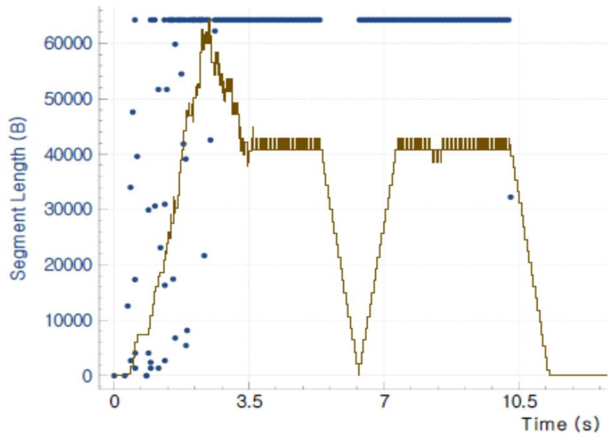


Figure 5. Browser-specific loading time in graph

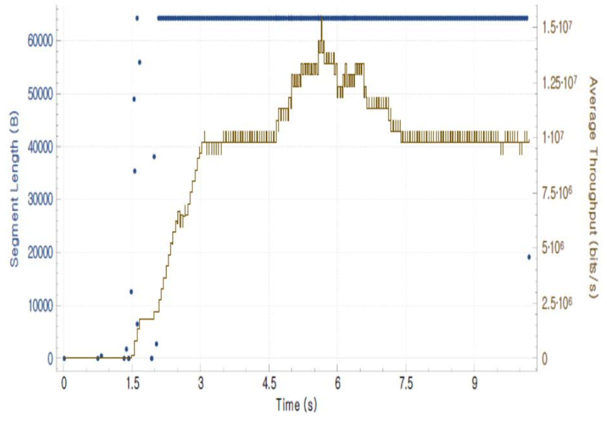
Figure 6 shows the packet segment length of 3D model data and the average bit per second. The packet segment length indicates the size of data loadable onto the TCP except the IP header and TCP header. The brown line indicates the average bit per second.

Figure 6(a) shows the packet segment length of 3D model data and the average bit per second of STL in Chrome. From the point of 0.75 sec. when the blue line started, packet transmission was initiated. Figure 6(b) shows the packet segment length of 3D model data and the average bit per second of STL in Edge. From the point of 2 sec. when the blue line started, packet transmission was initiated. Figure 6(c) shows the packet segment length of 3D model data and the average bit per second of OBJ in Chrome. From the point of 0.9 sec. when the blue line started, packet transmission was initiated. Figure 6(d) shows the packet segment length of 3D model data and the average bit per second of OBJ in Edge. From the point of 0.5 sec. when the blue line started, packet transmission was initiated. Figure 6(c) shows the packet segment length of 3D model data and the average bit per second of FBX in Chrome. From the point of 0.6 sec. when the blue line started, packet transmission was initiated. Figure 6(f) shows the packet segment length of 3D model data and the average bit per second of FBX in Edge. From the point of 1.7 sec. when the blue line started, packet transmission was initiated. Figure 6(g) shows the packet segment length of 3D model data and the average bit per second of glTF in Chrome. From the point of 1 sec. when the blue line started, packet transmission was initiated. Figure 6(h) shows the packet segment length of 3D model data and the average bit per second of glTF in Edge. From the point of 1.5 sec. when the blue line started, packet transmission was initiated.

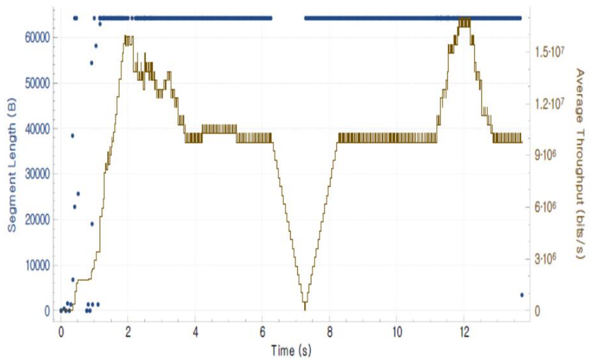
Figure 6 shows that there were changes depending on the network quality, and that packet transmission was initiated at the point where the blue line started. However, it was uncertain which 3D file format was faster than others after 1 or 2 inflection points and even upon initiation of packet transmission. Thus, it was necessary to examine the specific beginning and end of general packet transmission.


Segment Length (B)
Time (s)

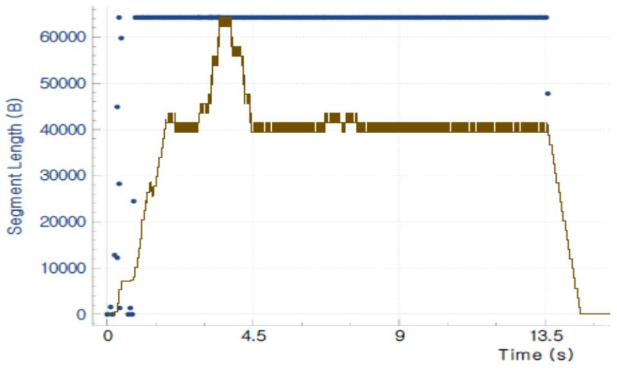
(a)


Segment Length (B)
Average Throughput (bits/s)
Time (s)

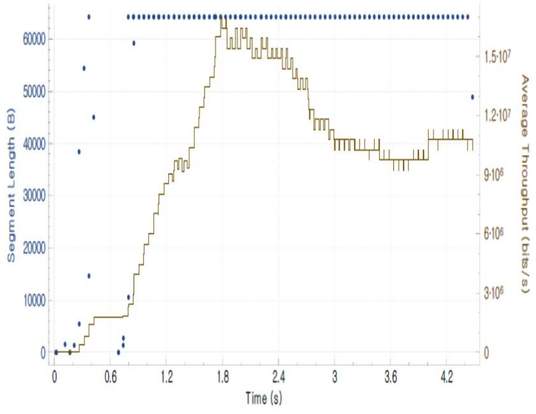
(b)


Segment Length (B)
Average Throughput (bits/s)
Time (s)

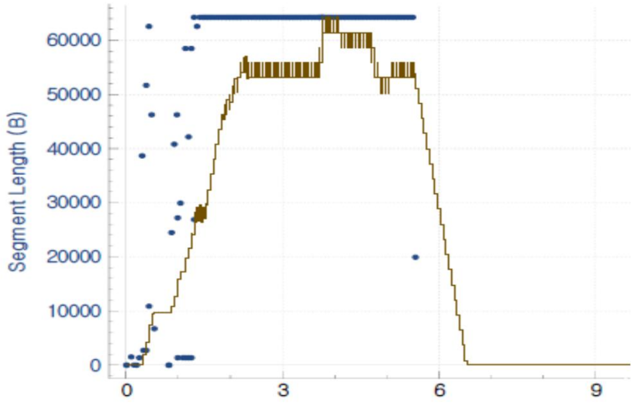
(c)


Segment Length (B)
Time (s)

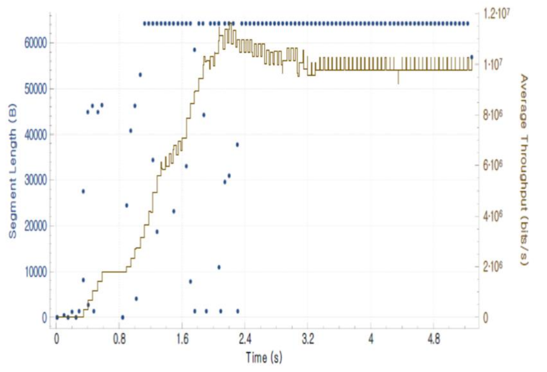
(d)


Segment Length (B)
Average Throughput (bits/s)
Time (s)

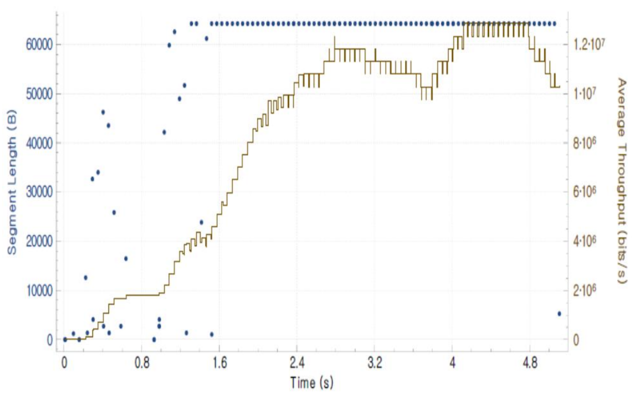
(e)


Segment Length (B)

(f)



(g)



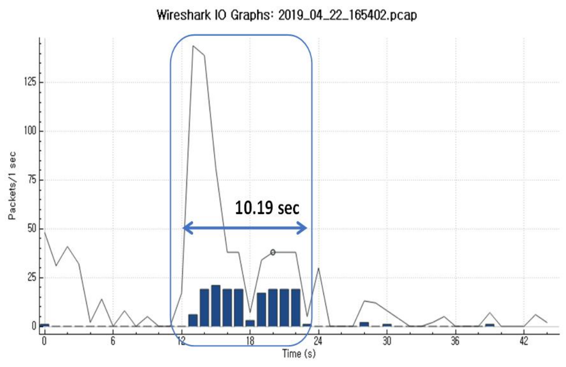
(h)

Figure 6. Packet Segment and Throughput Graph

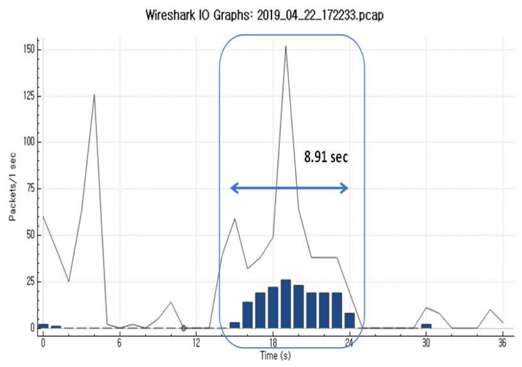
(a) Chrome_STL, (b) Edge_STL, (c) Chrome_OBJ, (d) Edge_OBJ,

(e) Chrome_FBX, (f) Edge_FBX, (g) Chrome_glTF, (h) Edge_glTF

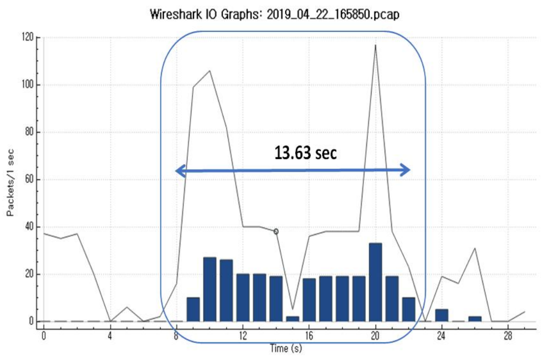
Figure 7 shows the beginning and end of 3D object packet transmission based on the quantity of packets transmitted per sec. In Figure (a), 3D model data packet transmission started at the point of 13 seconds and ended at the point of 23 seconds with no delay. In Figure (b), 3D model data packet transmission started at the point of 15 seconds and ended at the point of 24 seconds with no delay. The blue bar graph shows the number of packet retransmission requests. In Figure 7(c), OBJ data packet transmission started at the point of 9 seconds and ended at the point of 22 seconds with no delay. In Figure 7(d), OBJ data packet transmission started at the point of 6 seconds and ended at the point of 19 seconds with no delay. In Figure 7(e), FBX data packet transmission started at the point of 4.5 seconds and ended at the point of 9 seconds with no delay. In Figure 7(f), FBX data packet transmission started at the point of 2 seconds and ended at the point of 11 seconds with no delay. In Figure 7(g), glTF data packet transmission started at the point of 34 seconds and ended at the point of 39 seconds with no delay. In Figure 7(h), glTF data packet transmission started at the point of 5 seconds and ended at the point of 10 seconds with no delay.



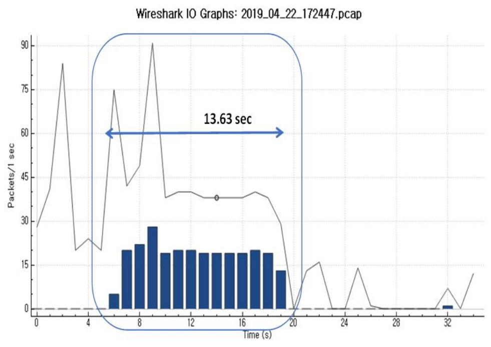
(a)



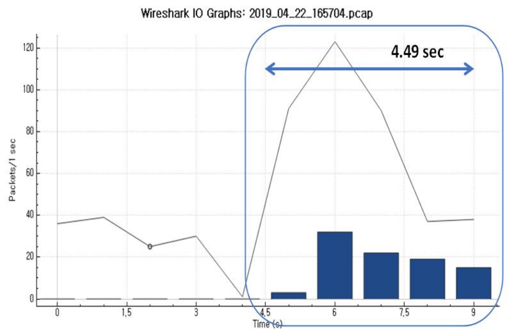
(b)



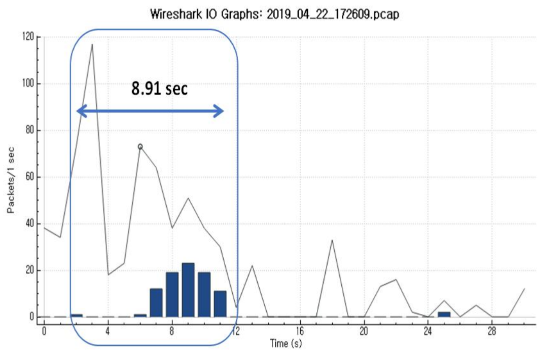
(c)



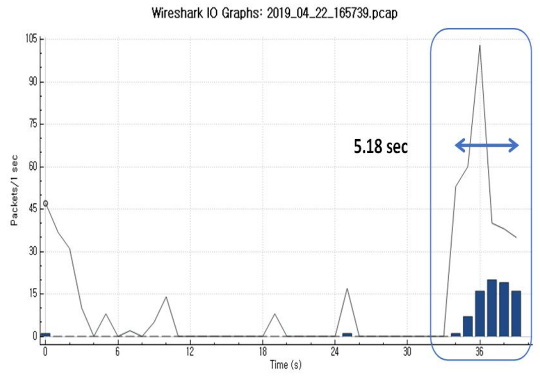
(d)



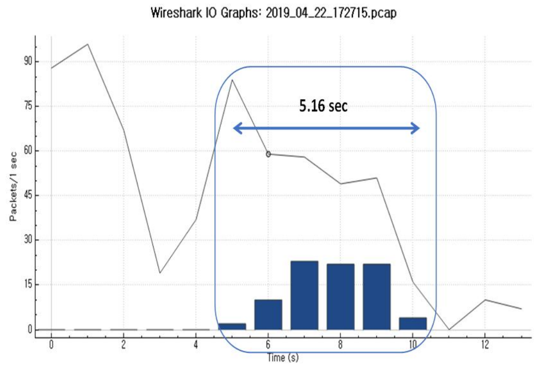
(e)



(f)



(g)



(h)

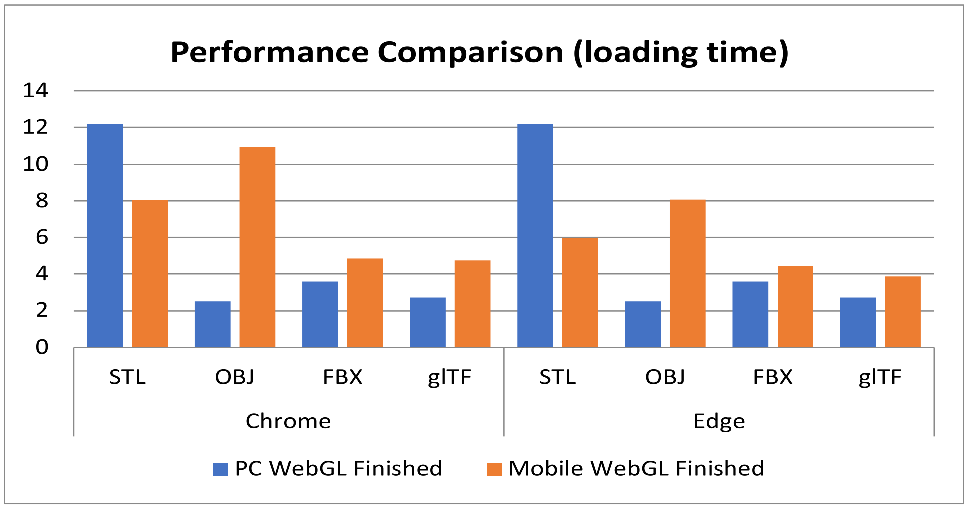
Figure 7. Packet I/O graph

(a) Chrome_STL, (b) Edge_STL, (c) Chrome_OBJ, (d) Edge_OBJ,

(e) Chrome_FBX, (f) Edge_FBX, (g) Chrome_glTF, (h) Edge_glTF

When the results of these experiments were compared with those of PC WebGL in previous studies, the loading time measurements depending on the 3D modeling file formats were different. In these experiments, the longest loading time was measured in the OBJ file format while the STL file format showed the longest loading time in the case of PC WebGL.

Table 5. Performance Comparison Result



5. Discussion

This paper explores the structure and principles of Mobile WebGL and compares the mobile loading time among 3D file formats such as STL, OBJ, FBX, and glTF. Experiment results may be summarized as follows: First, the packet segmentation in packet transmission was unclear in the case of Mobile WebGL. Only 2 segmentations were observed as in Figure 7(a) that shows chrome_stl and as in Figure 7(c) that shows chrome_obj. It is thought that due to the instability of wireless conditions, the TCP Session was disconnected and reassigned. In mobile packet transmission, all of the data is transmitted at once upon session assignment.

Second, the glTF file format demonstrated the best characteristics as shown in Table 4 and Table 5. glTF reduced overhead by using JSON and binary files, starting parsing earlier. For large-size data such as geometry, the format efficiency was maximized by storing it in a binary file. This is the reason why Open GL, Facebook, Google, and Microsoft started to support glTF format for 3D model data since glTF was released, and the base has been expanded accordingly.

Third, mobile packet transmission involves transmission errors in wireless sections more frequently than in wired conditions. As shown in Figure 7 that shows the result of packet I/O analysis, packet retransmission occurred relatively often. As the 3D model file size increases, the general loading time is affected accordingly. The OBJ file was the largest in size (16 MB) among models, and the general loading time also was the longest. As 3D model data was as large as 70MB, the loading time was extended significantly in mobile WebGL.

6. Conclusion

glTF was demonstrated as a 3D modeling file format suitable for representation of various realistic contents such as augmented reality contents in mobile settings. However, as the file size was as large as 70MB, the loading took 60 seconds or longer. Due to the characteristics of the 4th generation mobile communication network, packet transmission involved errors and retransmissions frequently. As the 5th generation mobile communication network is stabilized, it will be able to compare the packet transmission rate in the same experimental conditions. It is expected that the findings of this study can be referred to as quantitative data not only by users who experience inconvenience due to the long rendering time of 3D model data transmission but also by researchers examining obstacles to Mobile WebGL continually.

ACKNOWLEDGEMENT

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2019-2015-0-00448) supervised by the IITP(Institute for Information & communications Technology Promotion)

REFERENCES

[1] Z. Zhao, K. He, and R. Du, "The Simulation of Scara Robot Based on OpenGL and STL," Second International Conference on Mechanic Automation and Control Engineering (SICMACE), pp. 5429-5432, Jul. 15-17, 2011. DOI: 10.1109/MACE.2011.5988251.

[2] Jahanzeb Hafeez, Seunghyun Lee, Soonchul Kwon, and Alaric Hamacher, "Image Based 3D Reconstruction of Texture-less Objects for VR Contents", International Journal of Advanced Smart Convergence (IJASC), Vol. 6, No. 1, pp.9-17, Jun 2017. DOI: 10.7236/IJASC.2017.6.1.9.

[3] Jongdeug Kim and Taehyun Jeon, "An Effective Solution for the Multimedia Telephony Services in Evolving Networks", International Journal of Advanced Smart Convergence (IJASC), Vol. 2, No. 1, pp.24-26, Feb 2013. DOI: 10.7236/IJASC.2013.2.1.024.

[4] WebGL, *https://webglfundamentals.org/webgl/lessons/webgl-fundamentals.html.*

[5] D. Ginsburg, Introduction to OpenGL ES 3.0, Addison-Wesley, pp. 4-6, 2014.

[6] M. Zhao and J. Zhang, "Rapidly Product and Optimize Facial Animation Methods for 3D Game," International Conference on Internet Computing in Science and Engineering (ICICSE), pp. 136-139, Jan. 28-29, 2008. DOI: 10.1109/ICICSE.2008.15.

[7] Geonhee Lee, Pyeong-ho Choi, Hwa-seop Han, Seunghyun Lee, and Soonchul Kwon, "A Study on the Performance Comparison of 3D File Formats on the Web," International Journal of Advanced Smart Convergence (IJASC), Vol. 8, No. 1, pp. 65-74, Mar 2019. DOI: doi.org/10.7236/IJASC.2019.8.1.65.

[8] Geonhee Lee, Seunghyun Lee, and Soonchul Kwon, "A study on Compression of 3D Model Data and Optimization of Website," Journal of Engineering and Applied Sciences (JEAS), Vol. 14, No. 1, pp. 3934-3937, 2019. DOI: 10.3923/jeasci.2019.3934.3937.

[9] glTF 2.0, *https://www.khronos.org/events/webinar-khronos-gltf.*

[10] P. H. Shroff and S. R. Chaudhary, "Critical Rendering Path Optimizations to Reduce the Web Page Loading Time," International Conference for Convergence in Technology (ICCT), pp. 937-940, Apr. 7-9, 2017. DOI: 10.1109/I2CT.2017.8226266.