International Journal of Advanced Smart Convergence Vol.8 No.1 65-74 (2019)
http://dx.doi.org/10.7236/IJASC.2019.8.1.65

IJASC 19-1-8

A Study on the Performance Comparison of 3D File Formats on the Web

Geon-hee Lee[1], Pyeong-ho Choi[2], Jeong-hwan Nam[3], Hwa-seop Han[3], Seung-hyun Lee[1]
Soon-chul Kwon[1†]

[1]Graduate School of Smart Convergence, Kwangwoon University, Seoul, Korea
[2]Department of Plasma Bio Display, Kwangwoon University, Seoul, Korea
[3]MANO Landscape Design Group
[1]rjsgml5698@gmail.com, [2]phchoi@hotmail.com, [3]manogroup@choi.com,
[3]han@vrmano.com,{[1]shlee,[1†]ksc0226}@kw.ac.kr

Abstract

3D file formats typically include OBJ (Wavefront file format), STL (STereoLithography), and FBX (Filmbox). Each format has limitations depending on its configuration and usage, and supported formats are different depending on the software application. glTF helps uniform integration of 3D file formats and allows for more efficient transmission of large 3D geometry files by organizing them in a binary format. This paper presents explanation on OBJ, FBX, and STL which are major examples of existing 3D file formats. It also explains the concept and characteristics of glTF and compares its performance with other 3D file formats on the web. The loading time and packets of each 3D file format are measured according to the web browser environment by means of Google Chrome, Firefox and Microsoft Edge. Experimental results show that glTF is the most efficient and that it exhibits the best performance. As to STL, relatively excessive traffic was observed. This study is expected to contribute to reducing rendering time on the web as 3D file formats are used.

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

1. Introduction

Major 3D modeling file formats include OBJ, FBX, STL, glTF. Since such 3D file formats were developed for different purposes, their sizes and attributes are different from one another [1]. Among these, glTF features the multi-function support and light-weight in order for standardization of 3D file formats. In addition, glTF can save large-size elements such as geometry in a binary format and refers to such binary files through JSON, making the rendering process faster. This study presents details of glTF which is a multi-functional, light-weight 3D file format in comparison of its attributes with those of other 3D file formats being currently used such as OBJ, FBX, and STL. Additionally, the rending efficiency of each 3D file format in a web application is analyzed based on the loading time.

Manuscript Received: Jan. 13, 2019 / Revised: Jan. 21, 2019 / Accepted: Jan. 26, 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

2. Background Theory

2.1 OBJ, FBX, STL Importer and Converter Structures

Among OBJ, FBX, and STL which are major 3D file formats, OBJ was developed by Wavefront Technologies specifically for Visualizer. Data may be saved in an ASCII form (extension: obj) such as DXF[2] and IGES[3] or in a binary format (extension: mod) [6]. Since it uses a separate material file entitled MTL, MTL files that specify the texture map and material data need to be transferred as well in addition to model information. FBX was developed by Kaydara, Autodesk has the exclusive ownership. It may be expressed in a binary format disc or ASCII data. STL is one of the international standard formats that represent 3-D data, and it is used commonly in an input file for most 3D printers. STL may be read with binary or ASCII codes in a disc, but no information on model colors is saved. Figure 1 shows the process steps of existing 3D file formats. It is impossible to apply files directly to Graphic APIs such as WebGL immediately after such files are created by an authoring software tool such as Blender and Maya. 3D model data shall be converted by means of the corresponding Importer and Converter depending on the Runtime applications. The process of conversion by means of the Importer and Converter may cause a time cost to the user.

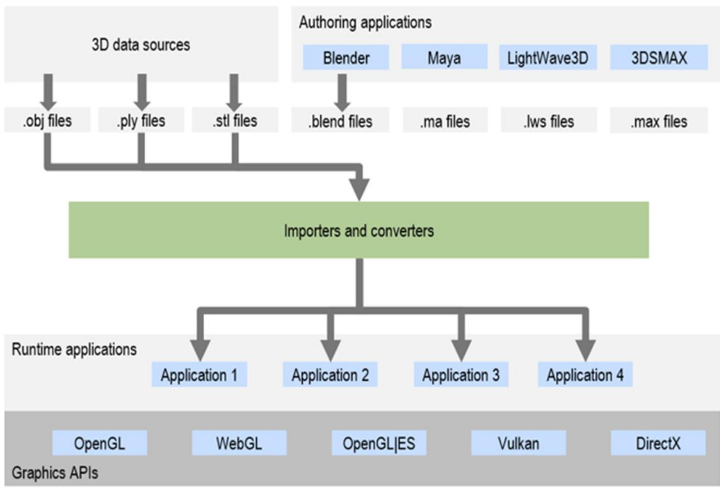


Figure 1. Importer and Converter of OBJ, FBX, and STL

2.2 glTF Structure

The format called glTF was designed specifically for Khronos Group in order to secure efficiency transmission of 3D contents between networks [4]. One basic element of glTF is JSON file which specifies the structure or formation of a scene that includes 3D models. Figure 2 shows the concept of glTF Asset and related top-level elements. Scene and Nodes are the basic structures of Scene, and Camera is an element that sets the time point of a scene. Mesh indicates the geometric information of 3D objects, and Buffer[5], buffer view, and accessors indicate information related to references and layouts. Materials define rendering methods of an object. Texture, Images, and Samplers represent the outer appearance of objects [6]. Skin indicates the peak skinning information, and Animation indicates information on time series changes. These elements are arranged in an array format. The reference corresponding to an object is a number that indicates a certain object in the array. This may be called an index number. A single binary glTF file includes a section where the entire information called an Asset.

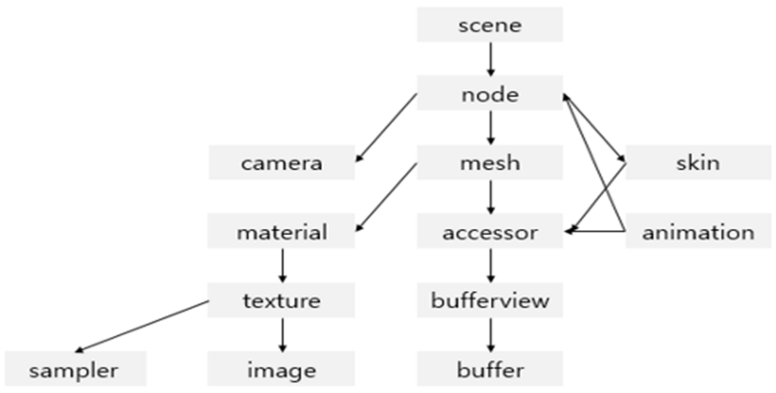


Figure 2. Structure of glTF

2.3 glTF Importer and Converter Structure

Figure 3 shows the relation between the Importer and Converter as glTF is created and processed. 3D file formats such as OBJ, FBX, and STL need to go through the process of 3D model data conversion in a form specialized for applications that are compatible to one another so that these formats can be embodied as a browser by means of Graphics APIs [8]. glTF needs no such conversion process or separate Importer. As a 3D model is defined by means of glTF, it is compatible with most applications that execute Graphics APIs.

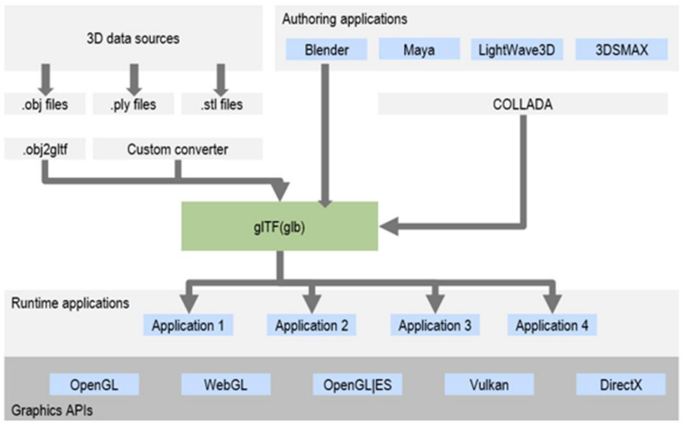


Figure 3. importer and Converter of glTF(glb)

3. Experiment Environments

For an experiment, OBJ, FBX, STL, and glTF(glb) files were created by means of the Blender. Quantitative data needs to be extracted based on the time of rendering and loading each 3D file format on the web. In the case of OBJ, a file needs to be loaded onto a web browser by means of OBJ-MTL Loader.js. For STL, STL Loader.js is used for rendering to a web browser. FBX needs FBX Loader.js. glTF(glb) needs GLTF Loader.js for rendering on the web. In order to secure objective criteria of comparison, therefore, the process of uploading by means of a loader was omitted, and a 3D model viewer supported by Autodesk was used instead [9].

A development tool provided by Firefox, Chrome, and Microsoft Edge was used to measure the time of rendering 3D model data on the web [10]. Such development tools indicate the time of rending on the web in detail. Major panels frequently used for debugging include Elements, Console, Network, and Sources Panel.

In the experiment stated above, Network Panel was used to measure the loading time of 3D model data.

Table 1. Experiment Environment

Classification	Information		
Measurement Tool	Google development tool		
	Firebug		
	Microsoft development tool		
Experimental site	https://viewer.autodesk.com/		
Used Browsers	Firefox, Chrome, Microsoft Edge		
Hardware	Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz (with SSE4.2)		
OS	64-bit Windows 8.1, build 9600		
Application	Dump cap (Wireshark) 2.6.4 (v2.6.4-0-g29d48ec8)		
3D File format	Verts	Faces	Tris
STL	542,969	1,084,030	1,084,030
OBJ	542,972	1,084,009	1,084,009
FBX	542,980	1,084,015	1,084,051
glTF(glb)	542,980	1,084,013	1,084,049

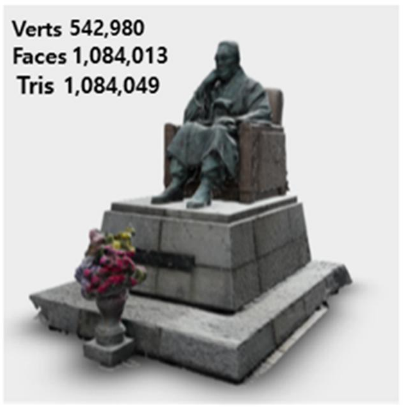
Table 1 shows elements used in the experiment including the browser development tool, experiment site, browsers, hardware performance, OS, packet measuring program, and numbers of dots, faces, and triangles of each 3D file format. Figure 4 shows OBJ, FBX, STL, and glTF(glb) in the Autodesk Viewer. Other file formats except STL include color information. Hence, textures were applied to the rest except Figure 4(d). Since STL has peak coordinate values but no color information, no texture is applied to it.

Verts 542,980
Faces 1,084,015
Tris 1.084.051

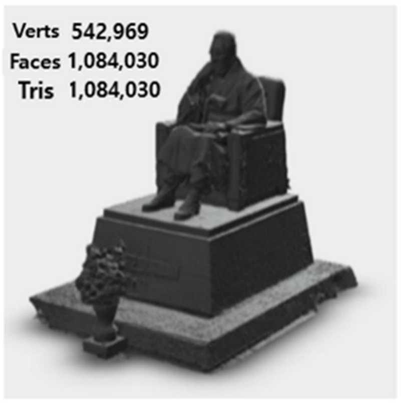
(a)

Verts 542,972
Faces 1,084,009
Tris 1,084,009

(b)



(c)



(d)

Figure 4. 3D model example - (a) OBJ (b) FBX (c) glTF(glb) (d) STL

4. Experiment and Result

4.1 3D File Format Attributes

3D file formats may be often selected regardless of the use, but it is necessary to choose one depending on the purpose since each 3D file format is developed for specific uses or purposes. Table 2 shows attributes of each 3D file format. glTF supports important functions such as color, animation, CSG, detailed mesh work, texture, camera, light, relative positioning, etc. 'CSG' stands for Constructive Solid Geometry. It uses a boolean operator, through which a modeler can create a complex surface or object by combining simple objects. OBJ, FBX, and STL reflect limited attributes. STL supports brief expression functions of geometry while important functions such as color, animation, CSG, detailed mesh work, texture, camera, light, relative positioning, etc. are omitted. OBJ supports certain functions such as brief expression, CSG, color, and material. As mentioned earlier, OBJ uses a separate material file named MTL, and thus it is necessary to transfer an MTL file that specifies data of the texture map or material in addition to the model information. FBX does not provide a detailed mesh function.

Table 2. Attributes by 3D file type

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 (glb)	O	O	O	O	O	O	O	O	O	O

4.2 Performance Comparison on the Web

3D model data loading is executable in Chrome, Firefox, and Microsoft Edge browsers. It is also possible to convert 3D model data into 3D file formats such as glTF, glb, OBJ, FBX, and STL. The loading time was measured based on the 3 criteria: domcontentloaded, load, and Finished. The domcontentloaded event indicates the time point when a DOM tree has been completed but an external resource (img etc..) has yet to be loaded. The load event indicates when every resource (img, style, script etc..) has been loaded on the browser. 'Finish' indicates a time period from the beginning and end of 3D model data loading after every source is downloaded. Table 3 shows average values after 20 times of 3D model data reloading by each of Chrome, Firefox, and MS Edge browsers.

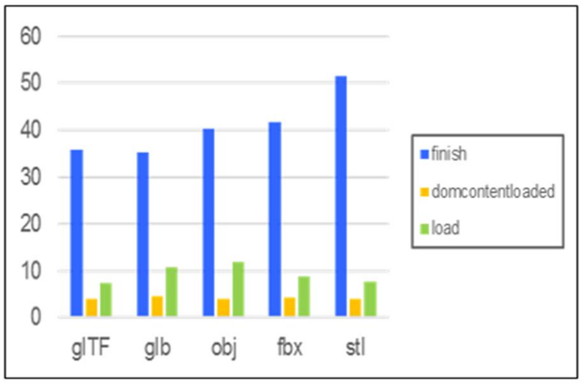
Table 3. Browser-specific loading time

Browser		Domcontentloaded(s)	Load(s)	Finished(s)
Chrome	glTF	2.304	6.712	36.222
	glb	4.831	7.529	38.598
	OBJ	5.212	12.966	41.721
	FBX	2.242	14.543	43.898
	STL	2.632	10.456	65.544
FIrefox	glTF	1.704	11.322	43.692
	glb	1.812	23.534	46.53
	OBJ	1.964	23.296	46.7
	FBX	1.764	22.408	44.588
	STL	1.896	8.45	70.014
MS Edge	glTF	2.304	6.712	36.222
	glb	4.831	7.529	38.598
	OBJ	5.212	12.966	41.721
	FBX	2.242	14.543	43.898
	STL	2.632	10.456	65.544

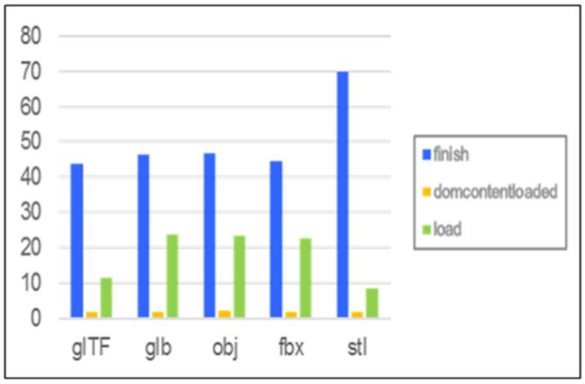
Figure 5(a) shows a graph of loading time in Chrome. The 3D model data loading time of glTF is shorter than that of OBJ, FBX, and STL. That of glTF was similar to that of glb. The length of loading time was in the order of OBJ, FBX, and STL.

Figure 5(b) shows a graph of loading time in Firefox. Just as in the case of Chrome, the 3D model data loading time of glTF was shorter than that of OBJ, FBX, and STL in Firefox as well. However, the ranks of OBJ and FBX in terms of loading time length were the opposite to those in Figure 5(a). This indicates that the measurement could be different depending on the browser and network environments.

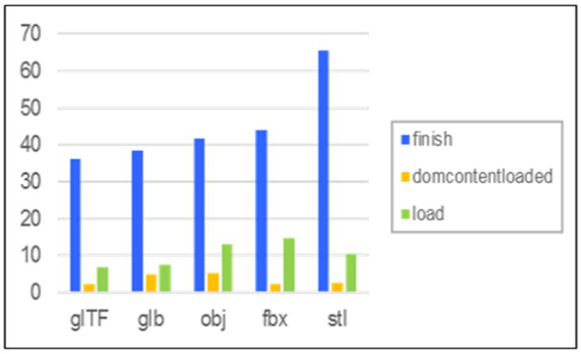
Figure 5(c) shows a graph of loading time in MS Edge. The results are almost the same with Figure 5(a). In general, STL took a longer loading time than glTF, glb, OBJ, and FBX. Lack of RAM occurred frequently.



(a)



(b)



(c)

Figure 5. Browser-specific loading time graph
(a) Chrome (b)Firefox (c)MS Edge

Since it turned in the experiment out that memory was a deciding factor affected 3D model data rendering to a significant degree, the memory use during the rendering process was measured and is shown in Figure 6. While the memory consumption of glTF(glb), OBJ, and FBX was the same, that of STL increased twice as much as that in glTF (glb), OBJ, and FBX, which indicates that it is a cause of loading delay.

In order to examine the major cause, OBJ and STL were compared based on definite criteria of comparison, and additionally, FBX, glTF, and glb were selected for comparison. The patterns of data transmission on the network was examined by analyzing specific packets.

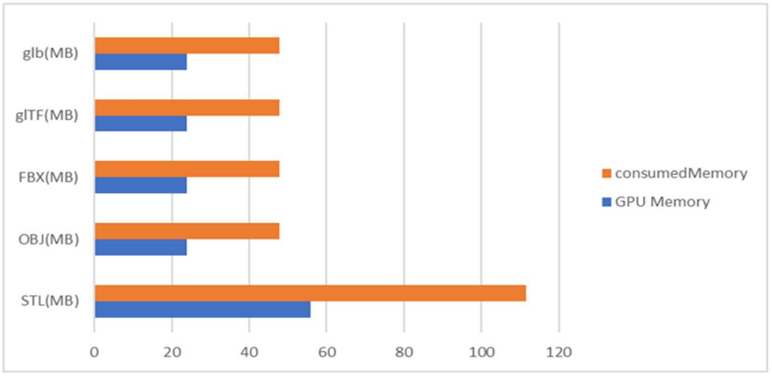
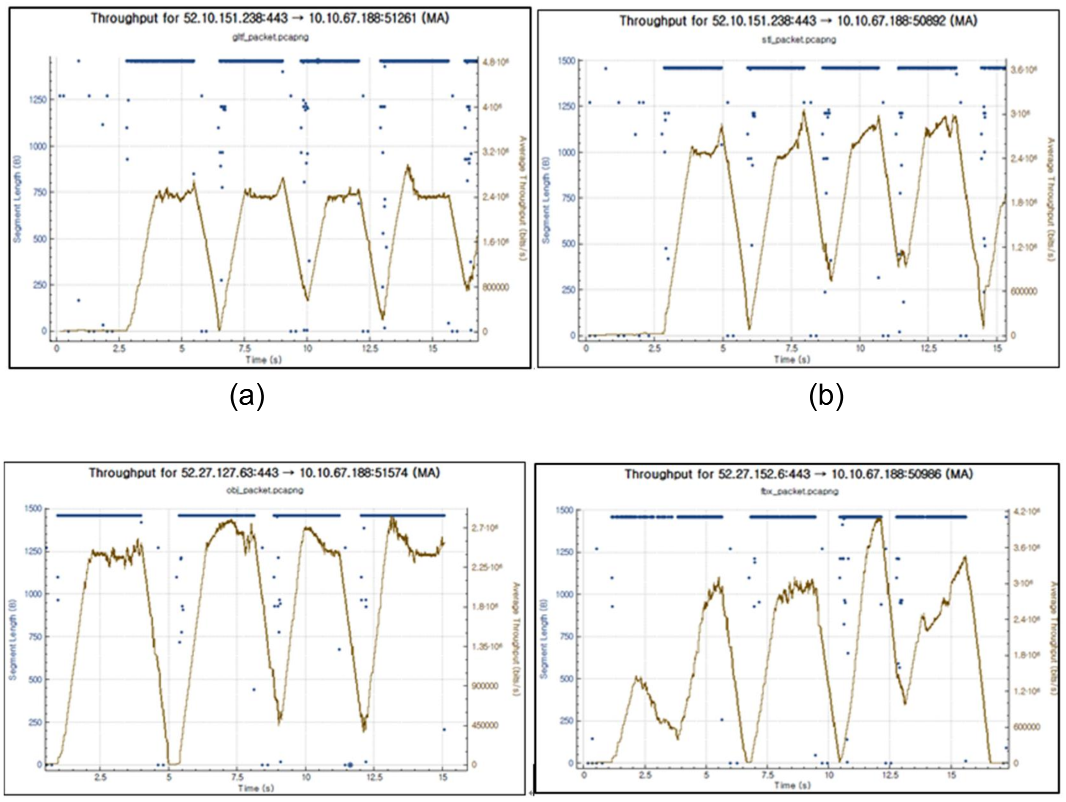


Figure 6. Browser-specific loading time

Figure 7(a) shows the packet segment length of 3D model data transmitted for 1 to 15 seconds the average bit rate per second in the case of glTF. The packet segment length indicates the size of data that TCP can bear except IP and TCP headers. The brown line indicates the average bit rate per second. Since the brown inflection point was repeated 4 times for 15 seconds, there was a change over time depending on the quality of the network. The blue points indicate the length of packet segments, which are 3D model data packets. The blue line consisting of blue points indicates that packet transmission proceeded properly. At the point of 2.6 seconds where the blue line began, packet transmission was initiated. Figure 7(b) shows the length of 3D model data packet segments transmitted for 1 to 15 seconds and the average bit rate per second in the case of STL. At the point of 2.7 seconds where the blue line began, packet transmission was initiated. Figure 7(c) shows the length of 3D model data packet segments transmitted for 1 to 15 seconds and the average bit rate per second in the case of OBJ. At the point of 0.3 seconds where the blue line began, packet transmission was initiated.

Figure 7(d) shows the length of 3D model data packet segments transmitted for 1 to 15 seconds and the average bit rate per second in the case of FBX. At the point of 2.2 seconds where the blue line began, packet transmission was initiated.

Figure 7 indicates that there was a change over time depending on the quality of the network, and packet transmission was initiated at the point where the blue line started. However, it is uncertain which 3D file format proved a higher rate than the others even at the 4 inflection points and initial points of packet acceptance. Thus, it was necessary to examine the beginning and end of the general packet transmission.



(a) (b)

(c) (d)

Figure 7. Packet Segments and Throughput (a) glTF(glb) (b) STL (c) OBJ (d) FBX

Figure 8(a) shows the throughput at the beginning and end of packet transmission. This experiment was conducted in order to examine the general status at the beginning and end of packet transmission in the case of glTF. The packet transmission of 3D model data was completed within about 27 seconds with no delay.

Figure 8(b) shows the throughput at the beginning and end of packet transmission. in the case of STL. Delay occurred from the point of 32 seconds, and the packet transmission took more than 90 seconds. This result indicates that while most of the 3D model data were transmitted, some part of the packet data failed to be transmitted quickly, which resulted in a delay. Figure 8(c) shows the throughput at the beginning and end of packet transmission. in the case of OBJ. There was no delay in packet transmission. until its completion in 25 seconds. Figure 8(d) shows the throughput at the beginning and end of packet transmission in the case of FBX. There was no delay in 3D model data packet transmission. until its completion in about 36 seconds.

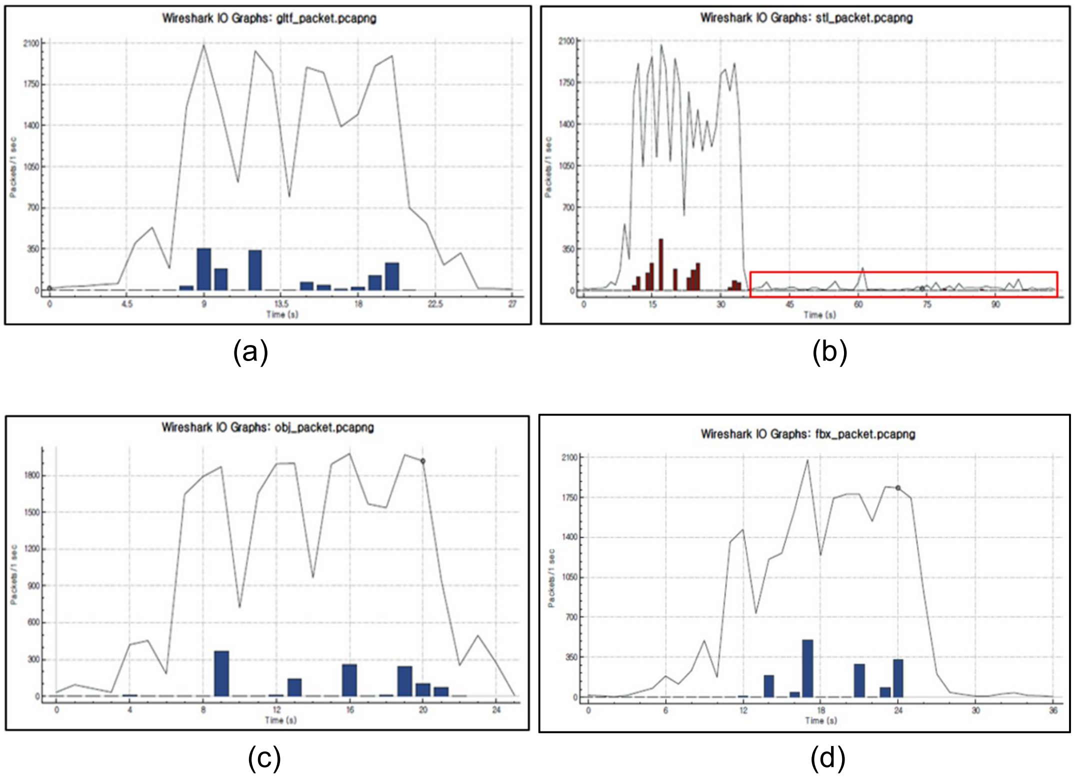


Figure 8. Packet I/O (a) glTF (b) STL (c) OBJ (d) FBX

4. Conclusion

This study presents an overview, structure, and principles of glTF in comparison with various other 3D file formats such as OBJ, FBX, and STL. glTF maximizes format efficiency by using JSON and binary files to reduce the overhead and starting parsing earlier, saving large size elements such as geometry in a binary file [18]. Since the release of glTF, Open GL, Facebook, Google, and Microsoft also have supported glTF format for 3D model data, and it is expected that the base will be expanded gradually.

The future study needs to embody improvements of STL. A delay occurs from the point of 32 seconds where 3D model data packet transmission is supposed to be completed, and the general transmission of packets exceeds 90 seconds. This result indicates that while most of the 3D model data were transmitted, some part of the packet data failed to be transmitted quickly, which resulted in a delay.

STL is used in limited areas compared to other 3D formats but advantageous in terms of user convenience as it is applicable to THINKERCAD which is a 3D modeling website. Therefore, STL is worth more research on functional enhancement. It is expected that this study is utilized as quantitative data useful for users who suffer inconvenience due to rendering delays in 3D model data transmission as well as STL researchers.

Acknowledgment

This work (Grants No. S2598342) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Ministry of SMEs and Startups in 2018.

References

[1] Zhenhua Zhao, Kai He and Ruxu Du, "The simulation of scary robot based on OpenGL and STL," Second International Conference on Mechanic Automation and Control Engineering, Hohhot, pp. 5429-5432, 2011.
DOI: https://doi.org/10.7236/JIIBC.2005.5.2.56.

[2] AutoCADDXF, https://en.wikipedia.org/wiki/AutoCAD_DXF

[3] IGES, https://en.wikipedia.org/wiki/IGES

[4] S. Kanamori, K. Fujiwara, T. Yoshinobu, B. Raytchev, T. Tamaki and K. Kaneda, "Physically Based Rendering of Rainbows under Various Atmospheric Conditions," Pacific Conference on Computer Graphics and Applications, Hangzhou, pp. 39-45, 2010.

[5] Cheng-Hsien Chen and Chen-Yi Lee, "Two-level hierarchical Z-Buffer for 3D graphics hardware," 2002 IEEE International Symposium on Circuits and Systems. Proceedings, AZ, USA, pp. II-II, 2002.
DOI: 10.1109/ISCAS.2002.1010972

[6] Cheng-Hsien Chen and Chen-Yi Lee, "Two-level hierarchical Z-Buffer for 3D graphics hardware," 2002 IEEE International Symposium on Circuits and Systems. Proceedings, AZ, USA, pp. II-II, 2002.
DOI: 10.1109/ISCAS.2002.1010972

[7] "IEE Colloquium on 'Binary Image Processing - Techniques and Applications' (Digest No.059)," IEE Colloquium on Binary Image Processing - Techniques and Applications, London, UK, pp. 0-1, 1991.

[8] Geonhee Lee, Seunghyun Lee and Soonchul Kwon, "A Study on Loading Speed of Web Browser for 3D Object," Advanced Engineering and ICT-Convergence Proceedings, Seoul, pp. 55-56, 2018.

[9] Autodesk viewer, https://viewer.autodesk.com/designviews

[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 (I2CT), Mumbai, pp. 937-940, 2017.
DOI: 10.1109/I2CT.2017.8226266