Journal of Engineering and Applied Sciences 14 (Special Issue 1): 3934-3937, 2019
ISSN: 1816-949X

A Study on Compression of 3D Model Data and Optimization of Website

[1]Geonhee Lee, [2]Seunghyun Lee and [3]Soonchul Kwon
[1]Department of Smart Systems, School of Smart Convergence, Kwangwoon University Graduate, Seoul, Korea
[2]Ingenium College of Liberal Arts, Kwangwoon University, Seoul, Korea
[3]Research Institute of Human Factors Convergence, Academy of Kwangwoon, Seoul, Korea

Abstract: Since, the invention of the web has made many advances. A lot of research has been done in the graphics field. As a result, a GPU was created inside the browser to render graphics on the web. The way to run the GPU is to use an API called Web Graphics Library (WebGL). But WebGL for implementing graphics on the web is difficult to use. So, Three.js was developed to make WebGL easy to use. It is a 3D graphics JavaScript library that has the advantage of being easy to use WebGL. However, it can also be a problem after web graphics are easily configured using Three.js. Sometimes it takes too much time to load on your website. The reason is that the size of the 3D Model data such as OBJ is too large or the source code that constitutes it is not optimized. To solve this problem, it is necessary to compress 3D Model data files such as OBJ and source code. OBJimg.js is a compression method of 3D Model data. OBJ and MTL files into a single PNG image, resulting in about 70% storage saving. The best way to optimize the source code is to use gulp.js, CDN, JavaScript and CSS compression techniques. In this study, we propose the quantitative analysis on compression of the 3D Model data and optimization of the loading time of the website. This research is expected to be helpful for the compression of 3D Model data and the optimization of the website.

Key words: WebGL, Three.js, OBJimg.js, 3D graphics, optimization, quantitative data

INTRODUCTION

The first start of the internet was ARPAnet developed by the US Department of Defense to facilitate networking research. ARPAnet connected mainframe computers from various universities and after 20 years, ARPAnet was the internet. British scientist Tim Berners-Lee invented the World Wide Web (WWW) at CERN, a Swiss physics research institute. About 2 years later, a browser-like WWW was launched. This was the world's first browser. Many browsers have begun to emerge from the beginning of this innovation. In 2008, Chrome was developed by Google and it was ranked #1 in browser rankings. As the web browsers progress, WebGL made many advances. But it is not as powerful as 3D professional authoring tools. Therefore, it is necessary to optimize 3D Model data and related source codes. This study quantitatively measures the compression of 3D Model data and the three proposed methods (Nath *et al*., 2014).

MATERIALS AND METHODS

Materials and simulation: WebGL (Web Graphics Library): WebGL is a JavaScript library for rendering 2D and 3D graphics in a web browser. WebGL is used in HTML5 by <canvas> element which provides 3D graphics through the introduction of OpenGL ES 2.0 compliant APIs. WebGL does not need a separate plug-in and 2D and 3D graphics are played back in the browser as soon as the code is written. WebGL works on almost all browsers and uses less GPU than CPU, so, it does not burden CPU. Figure 1 shows the outline of WebGL.

WebGL supported browser and Three.js: WebGL supports from Chrome to Samsung Internet browser as shown in Fig. 2. It has an environment that almost everyone can experience and develop. WebGL, however, is difficult to use linguistically by inexperienced users. Therefore, there is a JavaScript library developed for easy use by newbie users. The most used library is Three.js. Three.js was first released in April 2010 by Ricardo Cabello on GitHub. WebGL is the most widely used JavaScript library and Application Programming Interface (API). When implementing a 3D Model in a Web browser, developers can save time and money by simplifying existing WebGL syntax (Li *et al*., 2016). Three.js can create GPU-accelerated 3D animations using the JavaScript language as part of their website without relying on proprietary browser plug-ins.

Associated JavaScript library: OBJimg.js is a JavaScript library developed by Jordan-delcros. OBJimg.js compresses OBJ and MTL files into one PNG image which

Corresponding Author: Soonchul Kwon, Research Institute of Human Factors Convergence, Academy of Kwangwoon, Seoul, Korea

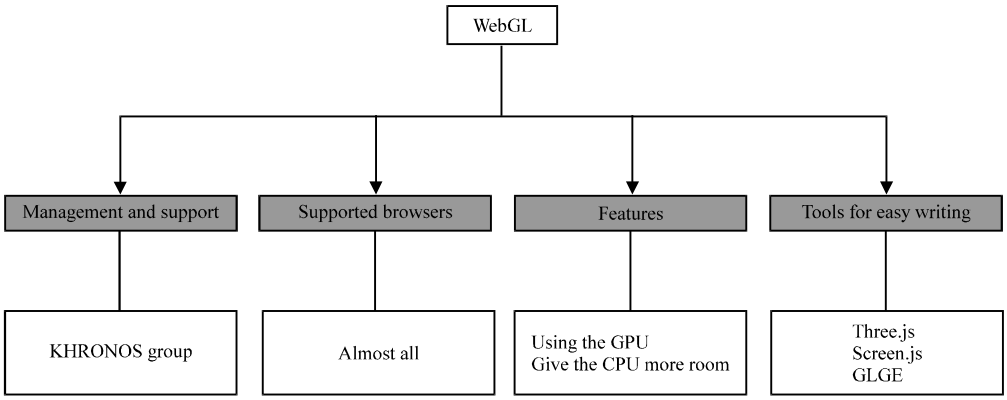

Fig. 1: A simple summary of WebGL

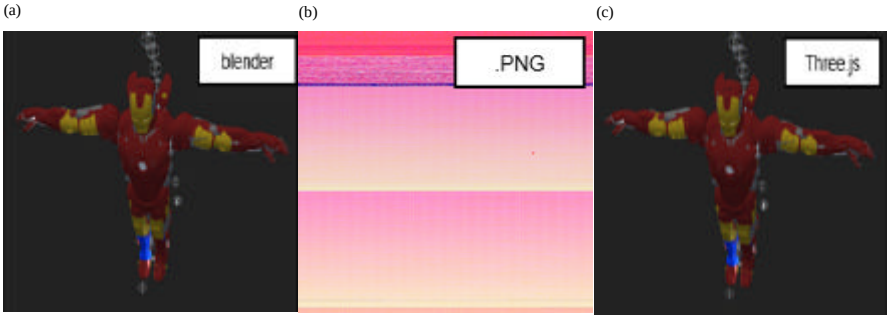

Fig. 2: The process of the 'compressed' method for: a) IronMan.obj; b) IronMan.obj. PNG and c) Rendered result

Table 1: Difference in capacity between OBJ file and PNG images

Type	IronMan.obj	IronMan.obj.PNG
Capacity	16.194 kB	2.949 kB

Table 2: Comparing the time difference of compression degree

Type	Compressed	Uncompressed
Time to fully load	14.006 sec	34.557 sec

results in reduced file size and improved rendering speed. For convenience and understanding, the method using OBJ-MTL Loader.js is described as 'uncompressed method' and the method by OBJimg.js is defined as 'compressed method'. This study focuses on the algorithm of OBJimg.js. Through the algorithm of OBJimg.js, 3D Model composed of OBJ and MTL is converted into PNG, thereby achieving speed efficiency by reducing the file size. Table 1 shows that the capacity is reduced through the compression method.

Figure 2 shows the OBJ and MTL files are converted to a single PNG and then rendered by the OBJimg.js algorithm. Figure 2a shows a 3D Model in a blender using OBJ and MTL files. However, if the 3D Model is downloaded and then rendered through Three.js, the size may not be appropriate. Blender is used to adjust the size of the 3D Model. Figure 2b shows a PNG image compressed using the OBJimg.js algorithm. Figure 2c shows the process of extracting the components of the 3D

Table 3: Experiment environment

Classification	Location
Measurement tool	http://webpagetest.org
Experiment site	http://regiit.org
Used browsers	Firefox, Chrome, Microsoft Edge
Test location	Dulles, VA USA

Model contained in pixel by the algorithm and rendering it in the web browser. To extract the information from the 3D Model from the PNG, the algorithm of OBJimg.js should be used again. Apply the codes generated by the algorithm according to Three.js. Then the 3D Model is loaded on the web. As can be seen in Table 2, therefore, it is much faster than 'uncompressed' method. Future experiments target compressed PNG images.

Experiment environment: The experimental environment is shown in Table 3. A good computer shows the loading speed of a very fast 3D Model on the local host but the experiment is not related to the performance of the computer, since, it was performed on an external public server. Tools for measuring the loading speed of 3D Models were performed at webpagetest.org. Speed measurement in individual browsers such as Firefox, Chrome and Microsoft Edge is also important. In Korea, 3D Model loading time is shorter than other countries because the server is based on Dulles, Virginia, the USA to meet the global standard.

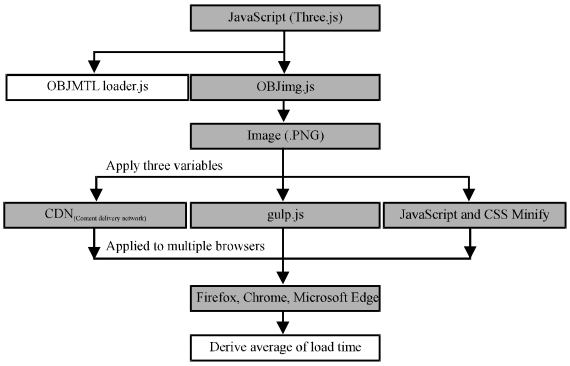

Fig. 3: The flow diagram of the process of 'compressed' method

Experiment composition: Nine tests were performed to measure the loading time. The standard of the comparisons is the average of the loading time. It is based on the PNG image computed by OBJimg.js which is compressed. Three methods including CDN, gulp.js and JavaScript and CSS compression were applied in the experiment. A CDN can fetch data at high speed from a nearby server (Aloui *et al*., 2018). gulp.js combines multiple JavaScript codes into a single file(). Compressing JavaScript and CSS makes your code light. These methods were applied to Firefox, Chrome and Microsoft Edge, to measure the final load time of 3D Model. Figure 3 is a flow chart that illustrates the experiment flow diagram.

RESULTS AND DISCUSSION

Table 4 shows the loading time when three methods including gulp.js, CDN and JavaScript&CSS compression are utilized and when they are not utilized. When three methods were used, it was shown that the time was reduced by about 15% rather than the state where nothing was applied.

Figure 4 shows the results of Table 4 in the form of graphs. Figure 4a shows a graph of the final loading time result when nothing applied. The highest loading time for Chrome browser's 3D Model is slow that reaches the 27.30 sec. Because Chrome uses a lot of resources to speed things up, it tends to slow down if the state of the server does not properly download the resources on time. Figure 4b shows the final loading time when three methods are applied. The Chrome browser showed fewer time variations. It complements an unstable server when the resources required for loading through the CDN are not downloaded properly. Both graphs show that Microsoft Edge loads faster than other browsers.

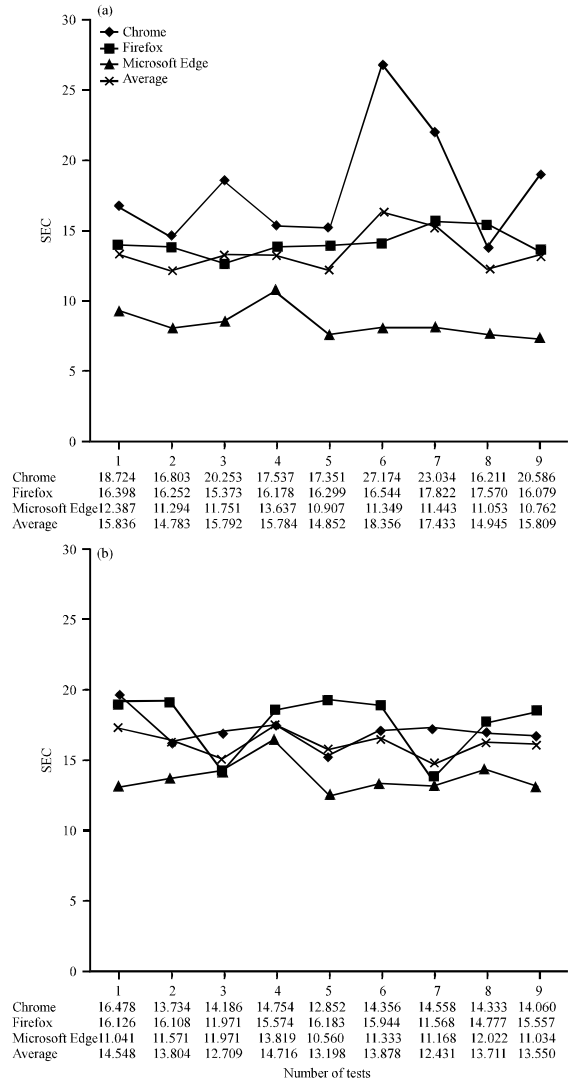

Fig. 4: Performance comparison: a) None applied and b) All applied

Table 4: Results of loading time test

Classification	Average time (sec)	Compare
Fully applied	13.616	.14.65%
Nothing applied	15.954	Standard

CONCLUSION

The study describes an optimization method for improving the loading speed and thus reducing the loading time of 3D Models in the Web environment (Rigby *et al*., 2016). The first step is to compress the 3D Model data. The 3D Model data is compressed through OBJimg.js. The second step is to optimize the source code using three methods: firstly, the CDN takes the data from the nearest server and reduces the time it takes to load the relevant sources. Secondly, gulp.js combines the number

of different script files into a single file. Thirdly, JavaScript and CSS files are compressed. In order to obtain the results of the compression of the 3D Model and the improved state of the code, it was applied to individual browsers and results are derived through by graphs. This study is expected to be a way to improve the loading speed of websites by compressing large 3D Models and optimizing source files (Rahman and Chung, 2018).

ACKNOWLEDGEMENTS

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2015-0-00448) supervised by the IITP (Institute for Information and Communications Technology Promotion).

REFERENCES

Aloui, M., H. Elbiaze, R. Glitho and S. Yangui, 2018. Analytics as a service architecture for cloud-based CDN: Case of video popularity prediction. Proceedings of the IEEE 15th Annual International Conference on Consumer Communications and Networking Conference (CCNC), January 12-15, 2018, IEEE, Las Vegas, NV, USA., ISBN:978-1-5386-4791-2, pp: 1-4.

Li, P., X. Yu and J. Wang, 2016. Progressive compression and transmission of 3D model with WebGL. Proceedings of the 2016 International Conference on Audio, Language and Image Processing (ICALIP), July 11-12, 2016, IEEE, Shanghai, China, ISBN:978-1-5090-0655-7, pp: 170-173.

Nath, K., S. Dhar and S. Basishtha, 2014. Web 1.0 to Web 3.0-evolution of the web and its various challenges. Proceedings of the 2014 International Conference on Reliabilty, Optimization and Information Technology (ICROIT), February 6-8, 2014, IEEE, Faridabad, India, ISBN:978-1-4799-3958-9, pp: 86-89.

Rahman, W.U. and K. Chung, 2018. An efficient rate adaptation algorithm for streaming over HTTP. Proceedings of the 2018 International Conference on Information Networking (ICOIN), January 10-12, 2018, IEEE, Chiang Mai, Thailand, ISBN:978-1-5386-2291-9, pp: 486-491.

Rigby, P.C., Y.C. Zhu, S.M. Donadelli and A. Mockus, 2016. Quantifying and mitigating turnover-induced knowledge loss: Case studies of chrome and a project at Avaya. Proceedings of the 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), May 14-22, 2016, IEEE, Austin, Texas, USA., ISBN:978-1-5090-2071-3, pp: 1006-1016.