MATEC Web of Conferences **77**, 09003 (2016) DOI: 10.1051/matecconf/20167709003
ICMMR 2016

3D Web-based HMI with WebGL Rendering Performance

Atitayaporn Muennoi, Daranee Hormdee

Embedded System R&D Group, Computer Engineering, Faculty of Engineering, 123 Khon Kaen University, Thailand

Abstract. An HMI, or Human-Machine Interface, is a software allowing users to communicate with a machine or automation system. It usually serves as a display section in SCADA (Supervisory Control and Data Acquisition) system for device monitoring and control. In this papper, a 3D Web-based HMI with WebGL (Web-based Graphics Library) rendering performance is presented. The main purpose of this work is to attempt to reduce the limitations of traditional 3D web HMI using the advantage of WebGL. To evaluate the performance, frame rate and frame time metrics were used. The results showed 3D Web-based HMI can maintain the frame rate 60FPS for #cube=0.5K/0.8K, 30FPS for #cube=1.1K/1.6K when it was run on Internet Explorer and Chrome respectively. Moreover, the study found that 3D Web-based HMI using WebGL contains similar frame time in each frame even though the numbers of cubes are up to 5K. This indicated stuttering incurred less in the proposed 3D Web-based HMI compared to the chosen commercial HMI product.

1 Introduction

A Supervisory Control and Data Acquisition (SCADA) system is commonly used in the factory or building automation fields. A Human-Machine Interface (HMI) plays the essential role in a SCADA system. It not only presents data from sensors to a human operator, but also accepts input from the operator to send commands to control devices. Traditionally, the vendors of SCADA system provide proprietary HMI software as a part of their software product. However, along with the growth of web technology, the modern rendering technology especially, 3D graphics and the need of real-time communication can be utilized in order to exhibit production line in action [1], a 3D web-based SCADA HMI should gain popularity.

This paper studied on the performance of 3D Web-based HMI with WebGL Rendering in order to serve large 3D HMIs. Furthermore, rendering performance of 3D Web-based HMI is compared to a standard web HMI product to find out how 3D Web-based HMI can achieve a rendering performance when compared with a commercial product.

The rest of the paper is organized as follows: related work in Section 2, experimental descriptions of the 3D Web-based HMI are explained in Section 3, the performance evaluation of the 3D Web-based HMI is provided in Section 4, conclusions and ideas for further work are given in Section 5.

2 Literature Review

This section consists of previous works, evaluation metrics, 3D Web-based HMI rendering technology and Genesis64 HMI product detail, the benchmark for this work.

2.1 Previous works

Various works have been done on 3D HMI include HMI for java platform [1] and 3D HMI product which can deploy to the web [2].

An example of HMI products, Genesis64 is an HMI/SCADA automation software suite from ICRONICS Company. It presents a GraphWorX64 component that can publish HMI platforms to the web via WPF or Silverlight [3-6]. However, on Microsoft Windows, GraphWorX64 integrated with WPF for full function 3D HMI platform, DirectX and .NET Framework need to be installed then the HMI needs to be run on Internet Explorer [5, 7].

Traditional 3D web HMI usually found limitations such as specific platform and software installation required. To overcome those problems, a new approach is proposed. With 3D Web-based HMI with WebGL and WebSocket study [8], user can run the HMI on any platforms and no need to install any software. Moreover, WebSocket is used for rapid data communication between server and client instead of HTTP. The research results showed that 3D Web-based HMI could achieve a good resolution when data had a rapid changing [8].

2.2 Metrics

The indicator of successful web applications is how fast it can turn the data into pixel on screen and it still can satisfy user experience. Web HMI is also considered as one of web applications. To investigate the rendering performance of 2D/3D applications, a common metric is a frame rate or Frame per Seconds (FPS) [9-13]. Normally, web applications should produce more than 30 FPS, so human visual system cannot detect a frame skipping, which creates a good user experience [10, 11]. However, general applications are usually targeted at 60 FPS in order to be consistent with refresh rate 60Hz of most display devices [11, 12]. In other words, applications should take 16.67ms to prepare for each frame or also called frame-budget or frame time [11, 12], which frame time is the inverse of FPS. Although FPS is a common metric, but there is a defect because FPS is not linear against with the time that use to render a frame [12]. To clarify, even though applications contain the high FPS, it does not guarantee that applications can display smoothly if there is highly frame time variance. Thus, this study also analyzes the frame time in order to validate a frame time delay which causes a stuttering.

In this study, FRAPS [10, 11, 13] is used to capture the amount of FPS and frame times while web HMIs are running.

2.3 3D Rendering of 3D Web-based HMI

3D Web-based HMI uses WebGL to render a 3D scene. WebGL is a cross-platform web graphics library based on OpenGL ES 2.0 [14], which program in JavaScript. WebGL execute directly on a computer's Graphics Processing Unit (GPU) and all drawing contents are contained in the <canvas> element on HTML5 browser. To reduce complexity of using WebGL which is a low-level API, Three.js [15], a JavaScript 3D Library, is used. In Three.js, WebGLRenderer() is enabled to render a scene of 3D Web-based HMI by using a capability of WebGL. Furthermore, 3D Web-based HMI uses ability of RequestAnimationFrame() to run animations match with a display refresh rate for smooth and continuous display. [12, 16]

2.4 Genesis64, GraphWorX64 and Desktop Platform

To study rendering performance of 3D Web-based HMI, a standard 3D web HMI product was examined simultaneously in order to investigate how 3D Web-based HMI had rendering performance when it was compared with the other. Then, Genesis64, GraphWorX64 component from ICRONIC Company [7] was selected.

Genesis64 is HMI/SCADA software solution which consists of various components inside. One of those is GraphWorX64 that has a Web Publishing Wizard, which can convert a GraphWorX64 HMI files into a HTML file format and then publish the HTML file to a web server [7]. The publishing can be done in many platforms depending on a display type. This experiment focused on Desktop platform (.gdfx file) because this platform provides a full function 3D view. To use the desktop platform, client machines need to install upper .NET Framework 4.5. After this platform is published on a server, user can access the HMI via Internet Explorer [7].

3 Experiment Descriptions

The experiment was designed into 4 parts to compare our HMI against GraphWorX64;

First is Display Performance Evaluation. This was done by measuring the FPS while increasing the number of 3D cubes from 1 to 5K, where the objects could hardly move, hence, no need to explore further. Two web browsers, Internet Explorer V.11.0.26 and Chrome V.49.0.2623.110, were used. Unfortunately, GraphWorX64 can only run on IE here.

Second is Frame Time Latency Analysis. Since exploring only FPS cannot guarantee the display smoothness. Although the average FPS maybe adequate, rendering time for each frame can still swing a lot. And that could certainly deliver a terrible result with a chance of stuttering on the display. In order to analyze in depth about the rendering performance, the period each frame used for rendering for 60s was captured for rendering on only IE here. Six experiments have been conducted here for a single cube vs. 1K, 2K, 3K, 4K and 5K cubes.

Third is Execution and Rendering Time Analysis on our HMI. The purpose of this section is to identify time spent on each process. Time stamps for both execution time and rendering time were captured simply on IE.

Finally, CPU, memory and GPU usages were monitored via Windows Performance Monitor [17] and GPU-Z [18], respectively.

The entire experiment has been done on a computer with Intel® Core™ i5-3450 CPU 3.10GHz up to 3.50GHz, 8GB RAM and Graphic adapter NVIDIA Geforce GTX 550Ti with Graphics Clock 900MHz and Standard Memory Config 1024MB and Windows7 operating system.

In order to investigate the display behavior of our 3D Web-based HMI and GraphWorX64 HMI, it is necessary that both HMIs have the same initialization properties such as animation update rate, canvas size, testing object geometry, and so on. The environment of the testing in this study is defined as follow: Rendering target in section 3.1, testing regulation in section 3.2 and rendering update rate in section 3.3. All environments are for our 3D Web-based HMI and GraphWorX64 Desktop platform.

3.1 Rendering Target

In this experiment, both HMIs were specified to have the same target which could render animations continuously with rendering speed at 60FPS, which was determined in order to match with 60Hz display refresh rate of the testing display device. When the target was defined and almost-identical properties were initialized, both HMIs were ready to be investigated the rendering performance by observing the changing frame rate when both HMIs had more loads and the point where how much load effect on the frame rate to miss out of aimed target.

3.2 Testing Regulation

For the fair comparison, both HMIs must use the same testing regulations. 3D cubes were used here as the primary objects. All cubes were rendered on 1,920 x 1,080 pixel display resolution. To fit for view, five hundred cubes were stacked per level. In order to capture the changing frame rate, via one of the dynamic properties, those objects must spin around. Furthermore, this rotation movement also can illustrate on how 3D graphics have more benefit over plain 2D graphics in HMI.

3.3 Rendering Update Rate

As explained in Section 2, 3D Web-based HMI used RequestAnimationFrame() to drive animations on a screen. In the implement, it was used to increase a rotation degree of objects and it was used to update the scene and camera rendering for objects changed displaying on a screen.

Per GraphWorX64, the signal simulation was used to create an object movement and the parameters which derived from this signal used to specify an object degree. This implement uses the ramp up signal to animate the cube turns around its axis consistently. In addition, to display animations according with user requirements, GraphWorX64 needed to be setup the input property which is Update Rate. The Update Rate was used to determine the period during data will be displayed in millisecond. To study the best performance of GraphWorX64, the Update Rate 1ms was defined, which is the fastest Update Rate that GraphWorX64 can acceptable in other words GraphWorX64 can display animations continuously as fast as the HMI support.

4 Results and Discussion

Four experiments have been performed. The details are as follow.

4.1 Display Performance Evaluation

Figure 1 illustrated the results on display performance of the proposed 3D Web-based HMI on both IE and Chrome comparing with that of GraphWorX64 when increasing the number cubes. At the beginning, our HMIs can deliver the maximum Frame Rate (60FPS) for sometimes and even longer than those of GraphWorX64. And later all frame rates for all cases have dropped dramatically. Overall, our HMI on both web browsers can perform well and try to keep up with the commercial one throughout the time though not as good as GraphWorX64. Whereas at 30FPS, where human usually can detect skipping, our FPS (~1K cubes) on IE was approximately twice (only 1.3 times on Chrome) less than GraphWorX64's (~2K cubes), hence the choice of web browser is one of the factors effecting on the performance.

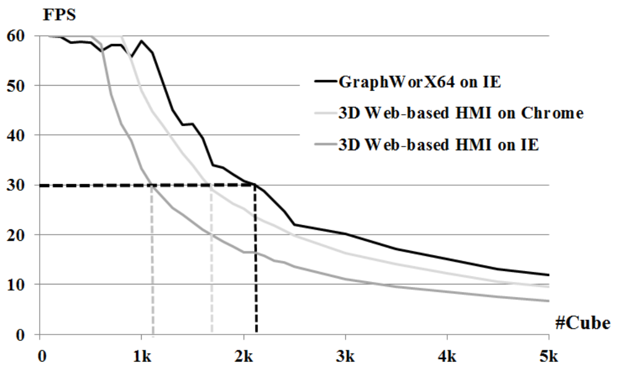

Figure 1. Frame per Second of GraphWorX64 and 3D Web-based HMI.

4.2 Frame Time Latency Analysis

Figures 2-7 shows frame time latency of 3D Web-based HMI and GraphWorX64. The display will have frequency as refresh rate of monitor (60 Hz) which means that display will be updated every 16.67ms although HMI can render frame faster than that. As shown in Figures 2, a single cube of our HMI can finish processing and rendering within less than 16.67ms. Hence, the result was displayed every 16.67ms whereas GraphWorX64 spent 16.67ms (refresh for displaying once) to 66.68ms (refresh for the same displaying four times). Spent many lengths of time rendering which means that there were different frame time as short or long frame time, so the display was inconsistent especially when the frame time was in the speed that human eyes can perceive (above 33.33ms). However, if the frame time on display is consistent (spending the same time in each frame), the display will run smoothly.

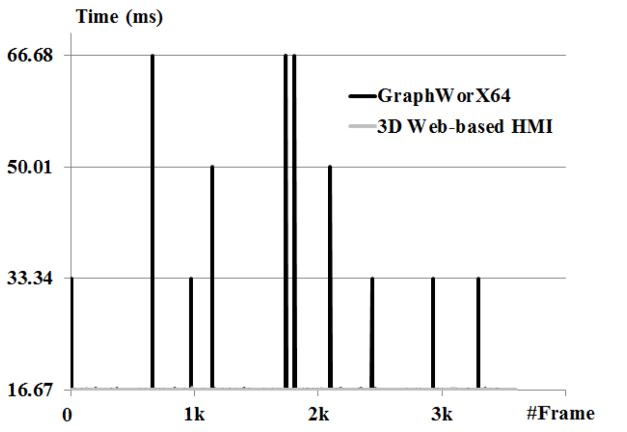

Figure 2. Frame Time Latency on 1 Cube Rendering

For 2K, 3K, 4K and 5K cubes (Figure 4-7), it can be clearly seen that even GraphWorX64 spent time producing display per 1 frame faster than our HMI (lower graph), length of time in displaying each frame was broader. That is, with the same number of objects, 3D Web-based HMI can display more smoothly except the case of 1K cubes (Figure 3) that cannot be directly compared to GraphWorX64 since any rendering that are lower than 16.67ms will only display at 16.67 ms; so, the experiment cannot detect result of GraphWorX64. On the other hand, our HMI was known that there are 2 lengths of frame time which are 16.67-33.34 and 33.34-50.01ms, and since some frames overlapped at 33.34ms, it could not finish within 2 times of refresh, and it was detected at 50.01ms (average frame time of 1K cubes is ~29ms and the time that our HMI spend for rendering per 1 cube will be explained in section 4.3).

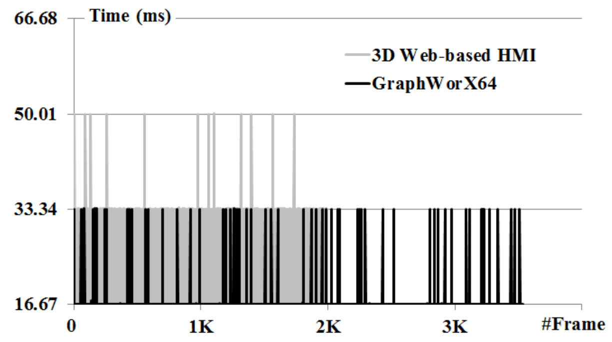

Figure 3. Frame Time Latency on 1K Cubes Rendering

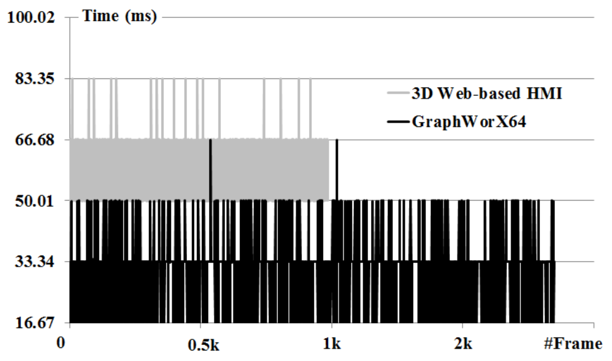

Figure 4. Frame Time Latency on 2K Cubes Rendering

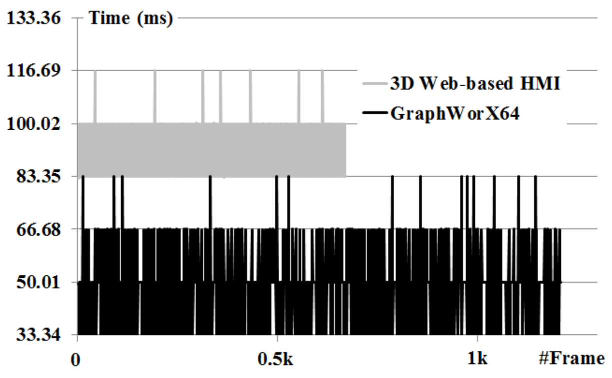

Figure 5. Frame Time Latency with 3K Cubes Rendering

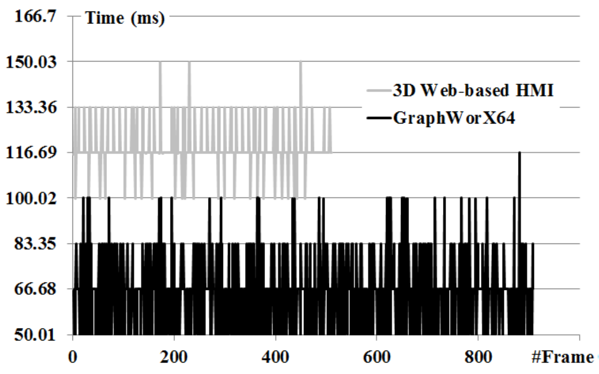

Figure 6. Frame Time Latency on 4K Cubes Rendering

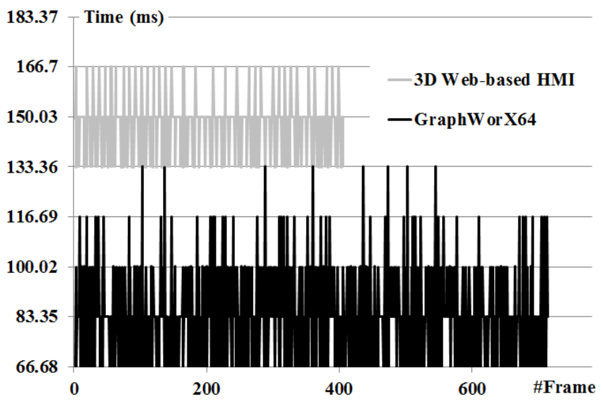

Figure 7. Frame Time Latency on 5K Cubes Rendering

4.3 Execution and Rendering Time Analysis

Examining the performance of our 3D Web-based HMI was found to be more instructive. One way to approach this is to ask, how much time does the process requires for creating the objects (Execution Time) vs rendering those (Rendering Time). Table 1 shows the results, however, execution and rendering time only for 1K-5K were large enough to be measured. Per all cases, over 96% of the time spent for displaying here went to rendering time, leaving only a small faction towards execution time. Execution Time can be reduced with a better algorithm used for resource management. While Rendering Time, caused by applying the original Three.js without any modification here, should be more focused on cause it was the majority. A well research on the code optimization, the Rendering Time can be significantly improved.

Table 1. Execution Time vs Rendering Time.

#Cube	T_{Exec} (ms)	T_{Render} (ms)	%Exec	%Render
1K	0.897	28.498	3.052	96.948
2K	1.815	57.249	3.074	96.926
3K	2.721	86.842	3.038	96.962
4K	3.655	113.928	3.109	96.891
5K	4.780	148.987	3.109	96.891

From the data, Rendering Time for a single object can then be calculated giving the results of 0.026-0.033ms. These results can endorse Figure that when the total rendering time for all objects less than 16.67ms is (refresh rate) our HMI can certainly obtain 60FPS. That complied with Figure 1 that with the number of cube under ~0.5K, our HMI can offer the maximum 60FPS.

4.4 Resource Usages

Lastly, the usages of CPU, memory and GPU have been explored. As the numbers of objects have been increased, both CPU and memory usages remained stable at 40% for CPU usage and ~240MB and ~180MB for memory for GraphWorX64 and our HMI respectively. While GPU usage seemed to be wobbly for both GraphWorX64 and our HMI, where ours used roughly 5% higher in average.

5 Conclusions and Future Work

The proposed HMI incorporated WebGL technology, a 3D graphics library commonly used in games and entertainment applications. This 3D Web-based HMI can be accessed simply via any HTML5 web browser on any platform without any specific/extra software requirement.

This paper has also presented the rendering performance evaluation of the proposed 3D Web-based HMI. The HMI contains 60 FPS when there are maximum 0.5K and 0.8K cubes (max performance) it contains 30 FPS when there are 1.1K and 1.6K cubes (minimum performance) when it was run on Internet Explorer and Chrome respectively. Frame time in each case study contain similar size, which means continuous and stable performance. For checking execution vs. rendering time, it was found that it spent most of the time in rendering as ~96%. And, it spent 0.026-0.033ms when rendering a single cube. Experiment contains CPU usage 40% for both; memory usage ~240MB and ~180MB for GraphWorX64 and our HMI respectively; GPU usage for our HMI was approximately 5% more than GraphWorX64.

As for future works, research on algorithm during execution time that can reduce the number of objects or polygons while maintaining to display the same model. The less components sent for rendering, the less loads in this process. This should yield to better HMI performance.

Acknowledgement

This work would have not been accomplished without the free product trial of Genesis64 HMI/SCADA software solution provided by ICONICS Inc.

References

1. O. Severa, M. Cech, and P. Balda, “New tools for 3D HMI development in Java,” in *Carpathian Control Conference (ICCC), 2011 12th International*, 2011, pp. 342–346.
2. R. Agrusa, V. G. Mazza, and R. Penso, “Advanced 3D visualization for manufacturing and facility controls,” in *2nd Conference on Human System Interactions, 2009. HSI ’09*, 2009, pp. 456–462.
3. Iconics.com, *GraphWorX64 - Vector-based Graphics Representing the Real World*, 2016. [Online]. Available: http://www.iconics.com/Home/Products/HMI-SCADA/GENESIS64/GraphWorX64.aspx#.VqXYvfmLTIU
4. Iconics.com, *Mobile HMI - Instant KPIs and Alerts for Mobile Devices*, 2016. [Online]. Available: http://www.iconics.com/Home/Products/Web-Solutions/MobileHMI.aspx#.VqXZufmLTIV
5. ICRONICS, Inc, “GENESIS64 Remote Client Options.” Sep-2013.
6. Iconics.com, *WebHMI - Web-Based Real-Time Automation Software*, 2016. [Online]. Available: http://www.iconics.com/Home/Products/Web-Solutions/WebHMI.aspx#.VqXcEvmLTIV
7. ICONICS, Inc, “GENESIS64 Standard Training Manual Version 10.61.”.
8. A. Muennoi and D. Hormdee, “3D Web-based HMI with WebGL and WebSocket”, in *International Conference on Embedded Systems and Intelligent Technology (ICESIT)*, Korea, 2014, pp. 59-64.
9. R. Hoetzlein, “Graphics Performance in Rich Internet Applications,” *IEEE Computer Graphics and Applications*, vol. 32, no. 5, pp. 98–104, Sep. 2012.
10. F. Hamady, A. Chehab, I. Elhajj, and A. Kayssi, “User experience-based mechanism for preserving energy in graphics-intensive applications,” in *New Circuits and Systems Conference (NEWCAS), 2013 IEEE 11th International*, 2013, pp. 1–4.
11. E. Rathgeb, K. Echtle, and B. Müller-Clostermann, *Measurement, Modelling, and Evaluation of Computing Systems and Dependability in Fault Tolerance: 15th International GI/ITG Conference, MMB & DFT 2010, Essen, Germany, March 15-17, 2010, Proceedings*. Springer Science & Business Media, 2010.
12. A. Anyuru, *Professional WebGL Programming: Developing 3D Graphics for the Web*. John Wiley & Sons, 2012.
13. Fraps.com, *FRAPS game capture video recorder fps viewer*, 2016. [Online]. Available: http://www.fraps.com
14. Khronos.org, *WebGL - OpenGL ES 2.0 for the Web*, 2016. [Online]. Available: https://www.khronos.org/webgl/
15. Threejs.org, *three.js - Javascript 3D library*, 2016. [Online]. Available: http://threejs.org/
16. Mozilla Developer Network, *Window.requestAnimationFrame()*, 2016. [Online]. Available: http://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame
17. Windows Performance Monitor, *Technet.microsoft.com*, 2016. [Online]. Available: https://technet.microsoft.com/en-us/library/cc749249.aspx
18. TechPowerUp, *TechPowerUp*, 2016. [Online]. Available: https://www.techpowerup.com/gpuz