HTML5/WebGL vs Flash in 3D Visualisation

Senad Bahor[*]
Supervised by: Belma Ramic-Brkic[†]

Sarajevo School of Science and Technology
Sarajevo, BiH

Abstract

Due to the recent technological developments and advancements in the field of computer graphics, and by following the great leap of the web-based technologies such as the greatest iteration and upgrade yet of the HTML to its new variant HTML5, we are experiencing a burst on the scene of the new architectural representation methods that aid in the visualization process. New technologies, both web-based and methodological-based, have not been fully understood and implemented so far, leaving a great unexplored space for the developers and the people involved with the process of the digitalization and 3D visualization to grasp in order to use the full potential of the today's technological advances. This paper is dedicated towards introducing the new technologies available for the 3D visualization and investigates their implementation on the web, with a focus on the comparative analysis of the emerging web technologies for the 3D rendering and visualization, as oppose to the current methods of 3D graphics implementation (Flash).Additionaly, the purpose of this paper is to empower the usage of the new HTML5 supported tool called WebGL and to use its full potential while delivering the 3D visualization to the web. The user study was carried out and the results revealed that HTML5 had a significantly higher impact as opposed to Flash. This indicates that Flash is no longer meeting the purpose of rendering demanding 3D graphics content and providing quality user interactivity, whilst HTML5/WebGL should be highly utilized in the future as the primary technologies for delivering 3D graphics to the web.

Keywords: HTML5, WebGL, 3D Visualization, Flash-HTML5, WebGL, 3D Visualization, Flash

1 Introduction

Up until recently, it has been noted that the interactive 3D graphics had a poor availability on the World Wide Web. Even though nowadays almost all PC's, mobile and embedded devices have high processor based computational power and high-performance 3D graphic hardware, which is necessary for processing 3D content, they have not been utilized for an effective web-based 3D interaction. The part of the problem also lies in the fact that most of the popular web browsers, such as Mozilla Firefox, Microsoft Internet Explorer or Google Chrome, were not utilized for a full 3D graphics presentation and interaction. There were and still are some partial approaches towards presenting the 3D content on the web and the identified major flaw was that none of the web browsers had built-in 3D graphics logic for presentation that will not require a plug-in or add-in to the browser.

While currently experiencing a higher demand for applications that can be used and accessed with a broad specter of smartphones and tablets types and models, the developers are starting to narrow the gap between the PC/Desktop/Notebook 3D graphics quality (which nowadays mostly use separate graphic card for demanding 3D graphic representation and with exceptional processing capabilities) and the mobile version of it. Consequently, the Internet medium has to be affected with this change, in order to develop and present high-detailed 3D graphics by using the web browser on both desktop-based and mobile-based devices. Besides just developing and rendering 3D content, developers are starting to address a great user demand for the easiest but also visually appealing 3D design and navigation application, 3D visualization animations and 3D games that can be accessed from any mobile device.

The development of improved 3D graphics in Web-based applications took a step forward recently, when programmers began building WebGL into the Mozilla Firefox nightly builds, and into WebKit, which is used in Google Chrome and Apple's Safari browser [3]. WebGL is one of the most developed libraries which are supported by the HTML5.

HTML introduces new features such as animation, offline capabilities, audio, advanced graphics, typography, transitions, and more, which yields a new class of web standards and replaces the need for proprietary technologies, like Flash and native mobiles platforms [2]. As the new HTML5 mark-up language emerged, the need to re-evaluate the best approach for delivering 3D content to the web had to be conducted and be compared to the

[*]senad.bahor@ssst.edu.ba
[†]belma.ramic@ssst.edu.ba

current trends and best practices utilized up until now in order to present the 3D content on the web. The most used tool nowadays, amongst others, for delivering the 3D and multimedia content to the web, is the Adobe Flash.

In this paper, we investigate which of the two currently best available tools for presenting 3D content on the web is also considered as such by the viewers. A user study was carried out in which participants were shown a computer generated environment of the cultural heritage site, presented in HTML5 and in Flash. Accurate and real-time 3D presentation of complex digital content has many application possibilities, especially in archeology, online games and education. So far, the content of such type was scarce on the web, due to the several constraints. Now, as the technology evolved and HTML iterated to a brand new version that empowers the usage of 3D content directly on the web, it is important to compare the best-practice standards against this new and emerging web technology.

2 Background

Flash was always focused on delivering interactive 2d graphics on the web. Objects within the Flash IDE had two properties, x and y axis for animating and positioning object on a 2d development plane. With the release of Flash Player 10, the developers introduced the third, z axis, for every Flash object, thus enabling the object to be moved or transformed in some way in the third dimension. However, this support for third dimension object interaction was delivered in a limited fashion. Although the objects could be transformed, moved and altered in some way by using third dimension, they still appeared somewhat flat. Some of the features for third dimension altering included rotation, movement, placement and other similar 3D model based adjustments. The need to extend the 3D object rendering and interactivity was then widened with the introduction of the third party engines that could be combined with Flash in order to deliver full 3D control of the object, such as Papervision 3D. Papervision3D was made into a public beta version, thus being available for the user to use freely all of the features that this open source 3D engine could deliver to Flash-based animations. Currently, there coexist two versions of the engine, with the first version building up on the Adobe ActionScript 2 and, therefore, requiring the Flash Player 8 or higher for efficient web browser presentation, while the second version brings some advances while building on the newer version of ActionScript (ActionScript 3) and requires the client machine to use the Flash Player 9 as the minimal requirement [9].

As opposed to the Flash, in terms of who controls the technology and how it is distributed to the end users, the HTML5 is guided by the members of the W3C, whose members are web developers, browser developers, various academic level parties and anyone who want to participate in developing and thus evolving the HTML5 [2, 8]. Meaning, anyone can participate and use the markup language to develop and deploy various web-based content and thus enable the web content evolution, from the static-based textual content to the full 3D dynamic web presentation. HTML5 represents a collection of great number of new APIs and semantics for the web developers that enhance the web content presentation, flexibility, correctness and portability. One of the greatest features presented in the HTML5 is the canvas tag element, which enables the easy integration of multimedia content without using any of the third party software, plugins of widgets. Through the canvas tag, a vast number of 2d and 3D graphics content can be embedded within the web page, thus making it not the part of the web page, but the actual web page. Some of the other new features and APIs presented within the HTML5 semantics include:

- Web Workers a collection of methods used to run a web-based scripts in the background as the background tasks, thus enabling the multi-threading of the post and response functions;
- Web Sockets which provide the bi-directional communication from the client-server machines;
- CSS3 an evolved version of CSS2 with a number of new layout approaches, gradients and content animation possibilities;
- Faster JavaScript engine with the extension to handle the OpenGL engine.

WebGL is one of the most developed libraries which is supported by the HTML5 specification and is implemented through the canvas element [2, 3]. The library contains a set of classes and methods which are extending the known JavaScript programming language in order to support the 3D content creation and rendering on the web. The specifications for the WebGL were created and published by the Khronos Group in March 2011, since the Khronos Group is in charge for the OpenGL and OpenCL standards. WebGL library is used as an interface, and thus extends the web based language, between the JavaScript language and OpenGL architecture (or even with OpenGL ES for embedded systems). The JavaScript code in this way allows the OpenGL code to be executed by accessing the graphics card architecture. The direct access to the hardware, that is, to the computer graphics card leads to a greater performance in rendering the 3D content, as opposite to leaving the software to render the graphics by using the computer CPU (where software-based applications are doing the graphics calculations). Since every modern PC comes with a powerful graphics card (even the integrated graphics card are now capable of doing great number of calculations on their own GPUs), the operating system handles the WebGL requests through a set of OpenGL libraries which are installed on the system.

3 Implementation

For the purpose of the comparative analysis between the HTML5 and WebGL, a 3D visualization model of the alleged Bosnian Valley of the Pyramids historical site was created using the Maya 2010 software. The model consists of the digitalized real-world 3D terrain, captured through the process of satellite image tessellation and conceptual 3D objects that mimic the presence of the pyramids on the site. The process of recreating the 3D scenes (one scene that shows the past and one scene that shows the current state of the real-world terrain) was then created in a series of steps based on [6, 7, 10, 11]:

- The z-map image (see Figure 1), with the resolution of 5340x8442px was imported into the project and attached to displacement map on the lambert shader. However, due to the fact that the image was too big in order for the displacement to take place (since the number of polygonal triangles would be too high for Maya to process and due to the 32 bit nature of the image), the z-map image was scaled down to 768x1214px, as shown in Figure 1.
- The polygonal plane (with U and V vertices count values set to 11) was placed into the viewport and attached to the lambert material with the displacement map and alpha gain property set to a value of 2.5, with subdivision width and height set to 25, as shown in Figure 2.
- The model was then rendered by using the Mental Ray rendered in order to evaluate the initial terrain quality, with the two directional lights illuminating the 3D scene (see Figure 3).
- Since the polygonal plane is not yet transformed to the tessellated polygonal object (and since the WebGL would not render the tessellation based on only the displacement property of the plane), the displacement had to be converted to the polygonal object in order to get the full 3D object for further manipulation. Maya contains certain tessellation properties, such as setting the maximum triangle count on the newly created tessellated object, which can enhance the tessellation displacement process in order to get the final polygonal object as close to the rendered object quality.
- 3D objects were imported on the 3D terrain in order to enhance the visualization of the scene in focus.

Both models were then prepared for the final export to the web in order to enable the interactivity with the 3D scenes and to enable the full 3D visualization of the site as

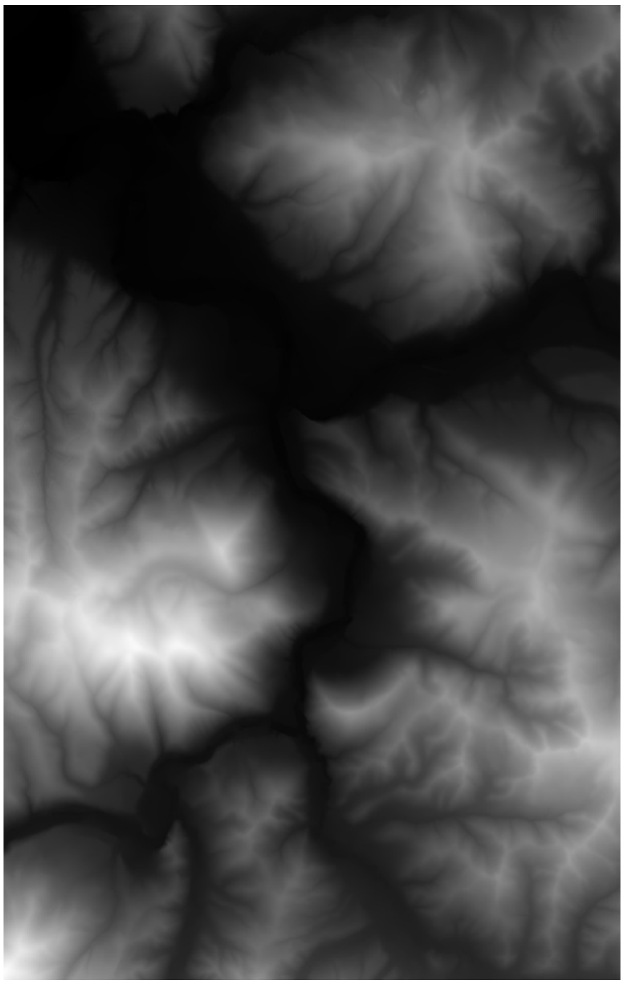


Figure 1: Z-map image

it evolved over the time. The visualization part (the time slider that switches the scene from the current state to the past state of the rendered 3D scene) can be added via the HTML controls (such as animated button than can be moved from left to right in order for user to slide between the scenes). For the purpose of creating the animation for the user study, the camera movement around the scene was created in Maya as well and exported via the Maya HTML5 plugin for the WebGL interpretation. Due to the complexity of the scene (taking into account a great number of poly surfaces out of which the accurate 3D model of the terrain is made of), the 3D scene had to be optimized in order to ensure that the final WebGL application can meet most of the hardware specification on the client machines, with a focus on mobile devices and portable PCs.

As previously described, the Flash based 3D content can be implemented within the web page through the Flash Player plugin, which allows the .swf object to be rendered on the web page through the manipulation of the player parameters. Usually, the developers create the .swf object

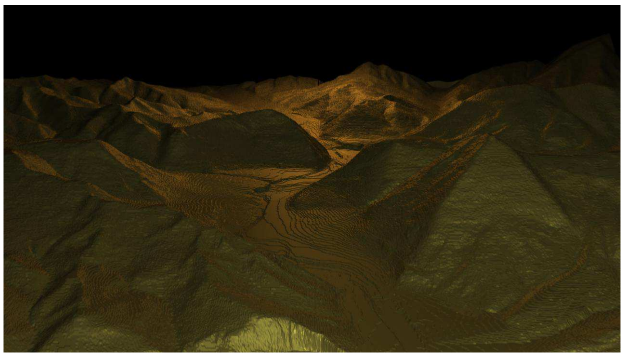


Figure 2: Result of the tessellation procedure - Accurate 3D model of the archeological site

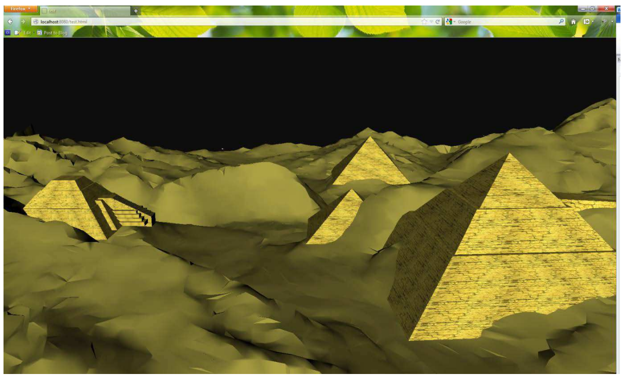


Figure 3: Exported HTML5 model on web browser

with a set of predefined parameters in order to enable the Flash object to be correctly rendered on the client machine.

With WebGL, there is no need to prompt the users to install any of the plugins in order to view and experience the 3D content on the site. The only prerequisite is that the web browser supports the HTML5 canvas element, through which the WebGL renders the content to the user. Furthermore, all Android OS based devices have a number of available web browsers (such as Mozilla Firefox, Google Chrome, Dolphin HD) that are able to read and render the WebGL, thus making it accessible to the wider audience as compared to the Flash Player problems with the smartphone and tablet devices(adding the fact that Adobe terminated the development and future enhancements for the mobile version of Flash plugin). In essence, WebGL can run on any platform and on any major systems that have the OpenGL capable graphic card and a browser that supports WebGL.

4 User study

The main purpose of the user study is to evaluate the two essential elements of experiencing 3D graphics, with a focus on web implementation: the technical element and the perception element. The technical element is comprised out of two evaluation parts: the performance and the usability part. The technical element is theoretically discussed, by highlighting key differences between two technologies, with survey results supporting certain benefits and drawbacks of both implementation approaches. On the other hand, the perception element and the evaluation of a user experience focusses on users themselves.

4.1 Technical elements

4.1.1 Performance

The performance aspect of both animations is evaluated through the process of how each technology is utilizing the processing power of the device on which the animation is running. Prior to the user study, the performance of both technologies was evaluated by going through the process of rendering the content for each technology in focus.

When it comes to Flash, the ability to reproduce the highly detailed 3D model and have it implemented on the web while maintaining smooth animation and higher FPS (everything above 20 FPS is considered fluent to the human eye), requires a deeper understanding of how Flash handles the 3D graphic content on deliverance.

Flash-based rendering relies heavy on software mode rendering (even with included third-party 3D engines, there is some overload on the software part of the system). Software rendering process depends heavily on the CPU capabilities to handle a great amount of 3D object rendering requests, process them and output them to the further ActionScript altering. Now, a 3D scene is defined as a group of 3D geometries called meshes, with each mesh being specified as a group of triangles. Each triangle is then made out of a group of vertices which really make up the 3D scene and add some additional information such as the color of the vertex points or texture information. Now, once the Flash viewport receives a stream of vertices that make the 3D scene, the internal engine would calculate the positions on the screen (render area) for the triangles and request the Flash Player to internally render triangles by processing them one by one. These renderings are conducted through a series of, so called, “fill” draw operations. No matter how the process of fill draw was optimized in the engine with a series of algorithms, the process is very slow and results in not particularly accurate rendering of initial 3D scene that was sent to the filling operations. The process is slow because the 3D scene is calculated, filled and rendered per triangle, instead of calculating it on the pixel basis. Rendering the content on the triangle basis often results in errors with the depth sorting, which would generate misplaced triangles due to their wrong depth calculation.

In Flash, a suitable polygon count to aim when designing a scene can be set at around 4.000 triangles [9]. This number represents the upper limit in order to get smooth frame rate in rendering content; as a result, the Flash Player (v.10) would be capable of rendering 3D scene with approximately 4000 triangles in order to enable the fluid and acceptable performance and visual correctness from the viewers perspective. Considering that there are number of third-party plugins and engines that enhance the Flash web-based 3D rendering in order to overcome these polygon count restrictions, I will focus on one in particular Stage3D.

Although Stage3D engine enables the Flash 3D content to be represented on the web site by using similar rendering logic as WebGL (processing the 3D scene on a GPU and correlated graphic driver basis), its major drawback is that it is not a standard API. Meaning, it is hard for the developers starting from scratch to go along the learning curve in order to comprehend the programming part (which is also the case with Direct3D and OpenGL) in order to get good results with 3D web-based rendering. It takes a lot of time and since Stage3D seems to be fairly poorly documented (the documentation is very sparse), it takes even more time to get the 3D graphics finally rendered within the web browser. One of the most problematic processes in Stage3D is the shader format, which is actually a programming assembler that uses a limited number of shader registers available when using a shading language such as AGAL. You can use a maximum of 8 temporary registers, instead of the 4096 available when coding registers in GLSL, with Shader Model 4.0, which WebGL can target. WebGL allows you to program in a convenient high level C like language. In other words, Shaders in Stage3D are designed to be very simple programs, as opposed to coding for more advanced hardware and Shader Models.

One of the most up-to-date performance tests between the Flash and WebGL canvas-based 3D content clearly shows that HTML5 canvas is starting to perform and generate higher frame rates while rendering 3D content on the web [5]. The results of the test show that WebGL is performing much faster in rendering 3D content and delivers higher frame rates for the 3D animations on the web, while comparing it with Flash. With WebGL, the process of rendering the 3D scene is switched from relying on the software and large CPU overloads to more OpenGL and GPU powered processing. This usually includes multiple subsequent draw jobs or “calls”, each of which is carried out in the GPU through a process called the rendering pipeline [2]. In WebGL, as it is a case with most of the 3D graphics that are being rendered in real/time, the lowest rendering unit is the triangle that makes up the 3D models. Further on, the drawing process is using the JavaScript in order to collect the information about where the specific triangle will be created and how it will be created. Additionally, every information set carries additional information, such as the types of shades that are out to be created on a specific triangle, the texturing model, color values associated to the triangle mesh etc. After the information is being collected by the JavaScript, the dataset is then being forwarded to the Graphics Processing Unit which will process the dataset and, by using the OpenGL algorithms and methods, render the 3D scene. As opposite to the Flash rendering procedure (or in case that Flash third-party engines fail to target the specific graphics driver), the rendering process is conducted mostly on the GPU, thus releasing the overload on the CPU.

Now, although the process of buffering and rendering the 3D scene on the GPU load looks complicated and with a heavy load, it actual performs much faster than the software based calculations, such as those that are relying on the CPU, which is the case for Flash Player 3D graphics rendering without any third-party engines attached to the process of rendering. Also, since the Flash Player poly triangle rendering limitation is set to around 4.000, it would be impossible to render anything that has higher triangle count than that. For instance, if the 3D model has a U and V vertices count set to 11 (the value which was used for this model) the tessellation procedure of the polygon model (based on the z-map texture) would render the polygon surface with approximately 32.000 triangles, a number that would be impossible to render in Flash Player.

4.1.2 Usability

On the usability part, the main focus was on how well each technology is rendering the content across multiple platforms and devices. In addition to that, it has been noted through the research which of the two can be combined and affected by other HTML elements in order to improve the user interactivity. This usability criteria is one of the most important criteria’s for the study, since the 3D virtual content is nowadays spanned across multiple devices, such as smartphones and tablets.

The Flash Player based content is wrapped inside the .swf object and is predetermined while developing the model. The 3D model inside the .swf model in encapsulated and is under the ActionScript determined behavior, meaning that no further alterations to the model can be done afterwards. Once the .swf model is deployed to the web site, it is wrapped with the Flash Player in order to integrate the Flash based content within the HTML markup language. However, neither on the web page can the HTML content penetrate the .swf object in order to do some alteration to the model or animation sequence. Also, Flash Player is known to cause instability issues when combined with JavaScript, due to the similarity of two programming languages and is often causing a web browser crash once the Flash Player content is present on the site with loads of JavaScript based content. Also, the Flash Player remains rather distant from the HTML elements and prohibits the interactivity between the HTML elements and the inner Flash 3D objects. Once stepping out of the box of the initial constraints that the native Flash engine opposes and taking into account third-party engines such as Stage3D, new problems arise. Stage3D is developed and implemented in such a manner that it targets specific hardware/ graphic driver specifications. In the end, with Stage3D, you do not really target specific hardware in order to take advantage of the specific power available in each platform. You just code generically for the virtual Stage3D platform; Stage3D acts as a layer between your code and the actual hardware. Therefore, the main drawbacks are based on the ability to develop one application for multiple platforms. By targeting all platforms at the same time with a single unified API, Stage3D cannot take advantage of the advanced features that are only present in the most powerful 3D graphics. To ensure that one app fits all, Stage3D has to abstract a 3D hardware device that is a common denominator, in terms of graphics capabilities, among all the platforms that are targeted.

With HTML5 WebGL, the canvas element that supports the 3D graphics rendering and presentation is actually a HTML element. This leads to a number of opportunities in a sense of combining two or more HTML elements in order to enhance the web interactivity. For instance, a user can render a set of geometrical figures within the canvas elements and create an ordinary HTML form with the labels and textboxes, rulers and button and connect the canvas elements functionality to the functionality of those form elements. As a result, the user could input a certain value in the form, press the button and experience the canvas 3D object being altered by that value entered. Then, the value would have to be forwarded to the JavaScript in order for the WebGL engine to do the calculations based on the value entered and apply that information to the 3D object that is sitting in the canvas element. In that way, a user can alter the model and enhance the interactivity by observing the model behavior. And, actually, that is what the 3D graphics is all about, about altering the 3D model and see it change on the time basis. This is something that can currently be achieved only with the usage of the HTML5 and WebGL engines on the web, since HTML5 provides interoperability between various elements (both traditional and HTML5 based) while inducing the JavaScript to handle the rendering procedure through the WebGL process.

4.2 Participants

16 participants, ages ranging from 22 to 27, mixed sexes (9 females and 7 male) from the postgraduate student population volunteered to participate in this study. All participants reported normal, or corrected to normal vision. The majority of them had taken a course in computer graphics and was familiar with concepts such as image quality and aliasing.

4.3 Design

HTML5 animation was prepared by exporting the Maya animation using the third-party HTML5 exporter named Inka3D. Inka3D is an Autodesk Maya to WebGL exporter, developed by Jochen Wilhelmy [1], which integrates within the Maya plugins and offers a set of export properties which can be controlled during the exporting procedure. Due to the complex and scarce documentation on Stage3D, that would enable the full 3D Flash integration, and due to the potential problems in targeting specific hardware and graphic specification with the Stage3D approach (which would lead to animation being rendered on CPU, shaders not rendering correctly or animation not showing up at all) Flash animation was created out of animation stills, with each animation frame exported from Maya in .tiff format and then reassembled into a .swf animation by using the Adobe Premiere software. The purpose of this approach, which was then evaluated through the perceptive-based questions in survey, was to see whether the precompiled .swf animation can match the full WebGL 3D integration in a sense of quality of the animation. In other words, the purpose is to test whether the user can see the difference between the precompiled video animation and fully integrated 3D animation.

4.4 Equipment and materials

The test environment comprised of an empty room, so that the subjects would not be distracted by surrounding objects, with a focus only on the device used by the participant for the testing purposes. The subjects that used PC watched animations on a full screen on 17 monitor (resolution: 1280*1024 pixels). They were seated at a normal viewing distance from the monitor (60cm). The constant Internet bandwidth was ensured throughout the whole testing time-frame, peaking at 5,96 MBps download speed.

4.5 Procedure

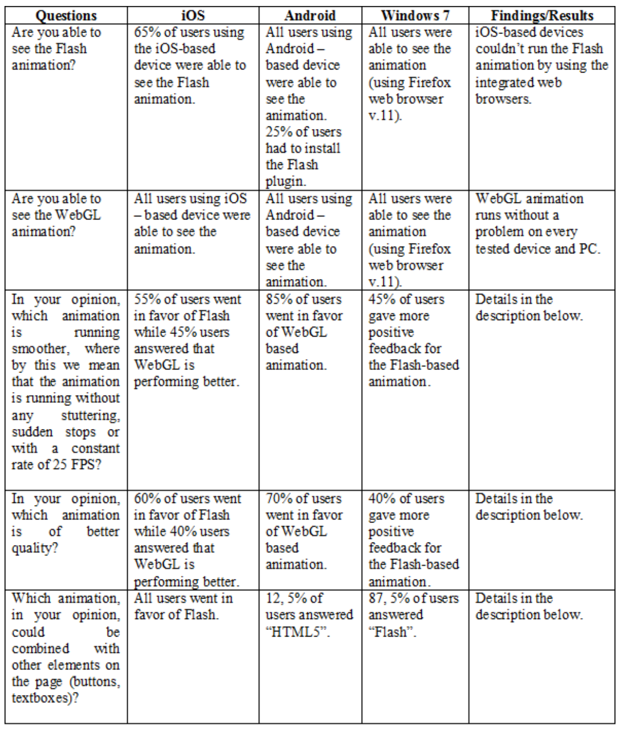
Each participant individually was shown two animations, with first animation being rendered by using Flash and another animation that utilizes WebGL. Before seeing both animations, viewers were asked to sign a consent form and only later, to answer the questionnaire and therefore anonymously supply some demographic information including details about their age, gender, eyesight, and their knowledge of computer graphics. Most importantly, they were asked to answer questions regarding the quality and smoothness of shown animations.

5 Results

The summary of the results is given in Figure 4. The results clearly show that Flash-based 3D graphics is close to impossible to present on the iOS-based devices, since the Adobe never supported iOS with its Flash plugin. There are some iOS based web browsers that are able to render the Flash content, but for most of the users (31.25% of participants) the native iOS web browser proved to be obsolete in rendering Flash-based content. Regarding the Android-based devices, the participants could see the official Adobe message across the Flash animation window stating that they are obliged to download the Flash plugin in order to see the content and that the Adobe will no longer support the Flash for Android devices in future. This was actually the greatest concern, because participants using Android devices did not actually know that Flash development for Android devices is about to be terminated, thus leading them to ask “What will replace the Flash content?”. HTML5 was one of the answers, as they experienced on the animation that followed the Flash-based one.

On the user perception evaluation side, which is directly addressed with the questions “In your opinion, which animation is running smoother, where by this we mean that the animation is running without any stuttering, sudden stops or with a constant rate of 25 FPS?” and “In your opinion, which animation is of better quality?”, most of the user answers went in favor of HTML5 animation. One of the reasons is that the rendered 3D scene was of better quality with the WebGL engine, due to the way that it handles shaders much faster and in greater quality and is capable of rendering the exact number of polygons that originated from the Maya based animation. Also, once the HTML5/WebGL animation is cached in the web browser, it starts up much quicker; almost instantly upon opening the web page where the animation is located. On a side note, users discussed how slow loading screens can affect the user focus and can result in user opening another page while waiting for the animation to show up on the first page (which is usually the case with the pre-compiled Flash content that cannot be cached in the web browser).

One of the key findings marked after the survey completion was that 85% users find that both animations are really high in quality and perform fast. Taking into account that one animation is a precompiled Flash animation loaded as a movie object, and another animation is the fully integrated 3D animation by using the HTML5 and web browser interpretation logic and JavaScript algorithms, it is remarkable that the WebGL animation has matched the movie-like Flash animation in every perceptive aspect. One of the most important results of the survey was the ones regarding the interoperability between the animation and other web elements. Only 4 participants (25%) knew that the HTML5 animation rendered throughout the canvas web elements and can be altered by other web elements, such as buttons of form elements. For future web development and integrating 3D visualization or any kind of 3D graphics, this needs to be addressed and researched.



Questions	iOS	Android	Windows 7	Findings/Results
Are you able to see the Flash animation?	65% of users using the iOS-based device were able to see the Flash animation.	All users using Android – based device were able to see the animation. 25% of users had to install the Flash plugin.	All users were able to see the animation (using Firefox web browser v.11).	iOS-based devices couldn’t run the Flash animation by using the integrated web browsers.
Are you able to see the WebGL animation?	All users using iOS – based device were able to see the animation.	All users using Android – based device were able to see the animation.	All users were able to see the animation (using Firefox web browser v.11).	WebGL animation runs without a problem on every tested device and PC.
In your opinion, which animation is running smoother, where by this we mean that the animation is running without any stuttering, sudden stops or with a constant rate of 25 FPS?	55% of users went in favor of Flash while 45% users answered that WebGL is performing better.	85% of users went in favor of WebGL based animation.	45% of users gave more positive feedback for the Flash-based animation.	Details in the description below.
In your opinion, which animation is of better quality?	60% of users went in favor of Flash while 40% users answered that WebGL is performing better.	70% of users went in favor of WebGL based animation.	40% of users gave more positive feedback for the Flash-based animation.	Details in the description below.
Which animation, in your opinion, could be combined with other elements on the page (buttons, textboxes)?	All users went in favor of Flash.	12, 5% of users answered “HTML5”.	87, 5% of users answered “Flash”.	Details in the description below.

Figure 4: The summary of the results

6 Conclusions and Future work

By using the new and emerging web standards and technology, combining the new HTML5 standard elements alongside with the WegGL in order to render high complex scenes, such as high polygon terrain scenes and objects that contain detailed textures that are necessary to preserve digitally correctly, the researchers and developers will find a lot of potential for displaying the 3D models and provide greater interactivity than by using other standards and tools. The interactivity of the exported WebGL model with other web elements can be easily and effectively achieved with the HTML5 and JavaScript that support the whole

WebGL logic, something that cannot be achieved with any other tools, e.g. Adobe Flash. Furthermore, due to the limitations of the currently available tools in a sense of the maximum triangle count on the 3D scene and the fact that they process the 3D scenes on the software side, thus congesting the CPU (also problem with the Flash-based third party engines not targeting correct graphic card drivers), the WebGL proved to be the most efficient tool to render the great number of polygon triangles by employing the GPU to do the heavy job of processing the vertex rendering requests. Also, the web page, due to the HTML5 mobility and accessibility, can be rendered on a great range of devices, including the iOS and Android devices, without users worrying about the presence of the plugins and player versioning.

References

[1] Inka3d, 2011.

[2] Luz Caballero. An introduction to webgl. 2011.

[3] Lin Edwards. Superior 3d graphics for the web a step closer. Online, June 2009.

[4] Craig Grannell, Victor Sumner, and Dionysios Synodinos. The essential guide to css and html web design. friendsofed (an apress company). November 2007.

[5] Etienne Levesque Guitard. Flash vs html5 performance (updated january 2012), 2011.

[6] R. Spallone M. Lo Turco, M. Sanna. Fourth dimension for representing and communicating architectural heritage. In *22nd CIPA Symposium*, Kyoto, Japan, October 2009.

[7] Domenica Constantino Maria Giuseppa Angelini and Nicola Milan. 3d and 2d documentation and visualization of architectural historic heritage. In *IXXIII CIPA Symposium*, Prague, Czech Republic, September 2010.

[8] Mark Pilgrim. Html5: Up and running. o'reilly media. 2010.

[9] Rob Bateman Richard Olsson. The essential guide to 3d in flash. friendsofed. 2010.

[10] Jean-Francois Lapointe Lorenzo Gonzo Sabry F. El-Hakim, George MacDonald and Michael Jemtrud. On the digital reconstruction and interactive presentation of heritage sites through time. In *The 7th International Symposium on Virtual Reality, Archeology and Cultural Heritage*, September 2006.

[11] N. Kawahara S. Koga T. Nakaya Y. Takase, K. Yano and T. Kawasumi et.al. Reconstruction and visualization of virtual time-space of kyoto, a 4d-gis of the city. In *ISPRS Proceeding*, September 2010.