WebGL on Mobile Devices

William Almnes, *University of Oslo*
Marko Andjic, *University of Oslo*
Matthias Armbruster *University of Mannheim*
and Paul Steinhilber, *University of Mannheim*

Abstract—Augmented Reality (AR) is not a completely new concept but it has in recent years, slowly but steadily, been gaining in popularity as the use and development of smartphones exploded. Relevant digital information layered on top of the users view has given us a new paradigm in the information use and interaction, and we have already seen new types of AR dedicated devices being introduced. WebGL, on the other hand, is a graphic library based on well-known OpenGL ES standard that opens the possibility for native 3D animations in a browser and on mobile devices. Even though the technology is still young it has been in the center of the research community and standards organizations which resulted in first implementations on almost all major web browser. Our task in this paper is to investigate to which extent WebGL as a new and prospecting technology can be used in future AR applications, integrated in browsers on mobile devices. To test the performance and support of WebGL on mobile devices, a 3D application was implemented and a comparison of several WebGL applications on different mobile devices was conducted.

Index Terms—WebGL, Augmented Reality, JavaScript, mobile devices, benchmarks

1 Introduction

The technologies which allow people to interact with the world around them evolve constantly. Whereas it was common to simply display information in a static form in the early days of the web, at present an ever increasing amount of content is presented in dynamic and interactive ways.

Most of the applications on smartphones are native apps; a growing number of applications are based in the browser, though, and the pace will accelerate the more powerful the mobile web browser gets. Moreover, it is predicted that web applications will completely replace binary software [1]. Many web sites already offer functionalities which were previously only found in native applications, e.g., word processing using Google Docs [2] or creating presentation using 280 Slides [3], thus narrowing "the gap between them" [4]. In the effort to, among other things, expand internet browser functionality to natively support three-dimensional graphics, HTML5 and WebGL have been developed. A native usage of 3D-functionality enhances these "web applications" even further.

Augmented Reality (AR) provides a bridge between digital information and the physical world, by enhancing a users view with additional information [5]. Application areas of AR can be as various as ranging from health care [6] to education [7], [8] to tourism [9]. The topic has gained momentum in the recent years thanks to the rise in smartphone usage and the popularity of some AR apps. "To make the world itself the user interface [...] may revolutionize the way information is accessed and presented to people in the future [10], [11].

The rise of smartphones is growing with a fast and still accelerating pace and enabled displaying information in a new way in a truly mobile context to many people [12], [13]. They

- *William W. F. Almnes: wwalmnes@student.matnat.uio.no*
- *Marko Andjic: marko.andjic@usit.uio.no*
- *Matthias Armbruster: marmbrus@rumms.uni-mannheim.de*
- *Paul R. E. Steinhilber: ps@paulsteinhilber.de*
- *Daniel Schön: schoen@informatik.uni-mannheim (advisor)*

Submitted just before April 23, 2012.

offer a much higher power than feature phones and even claim to offer the ”real web” experience with ”real browsers” [14]. There are differences, however, between the way information can be accessed from a desktop system and a mobile device, influenced by factors like screen size, processing power, and input methods.

3D support on mobile devices is still in an early phase. Adobe abandoned Flash in late 2011 [15], leaving WebGL as the main technology for providing interactive 3D content on the mobile web, even though it is still in an initial phase.

This paper analyzes the status quo and potentials of WebGL on mobile devices and answers the following questions:

1) **Are Augmented Reality applications possible on mobile devices using only the browser?**
2) **Is 3D-augmentation of these applications possible using WebGL?**
3) **How fast is WebGL on mobile devices?**

The paper is structured as follows: Chapter 2 gives background information on the relevant topics augmented reality and WebGL with mobile devices and their evaluation criteria; Chapter 3 gives implementation details of the the WebGL environment used to capture the status quo; Chapter 4 presents the evaluation as well as limitations to this study; and Chapter 5 gives a summary of the results and presents future research opportunities.

2 Background

In this chapter, terms and technologies which will be used to answer the research questions will be introduced, namely WebGL, Augmented Reality, and their evaluation criteria.

2.1 WebGL

WebGL is a cross-browser and cross-platform compatible software library and API that extends JavaScript to allow it to natively generate websites with hardware-accelerated 3D content. The first version of WebGL was released in 2011 [16].

2.1.1 Design

WebGL uses the standard HTML5 <canvas> element and exposes its functionality to the web-developer through the Document Object Model (DOM) interfaces [16]. Thus, other web content can be combined with it automatically, and can be layered on top, around, or underneath the 3D content, through any language that supports DOM (such as JavaScript).

WebGL is based on OpenGL ES 2.0 and uses its shading language, *GLSL*. Although it shares semantics with desktop OpenGL 2.0, in order to enhance portability of desktop code to mobile devices, some differences do exist, such as the need for power-of-two textures or lacking support of 3D textures [17].

2.1.2 Desktop and Mobile WebGL

Although support for WebGL is available on the desktop computers within all major browsers (FireFox 4+, Safari 5.1+, Chrome 9+, Opera 12+), with the exception of Microsoft’s Internet Explorer [4], browsers on mobile devices supporting WebGL are still very rare.

With Firefox and Opera Mobile 12 [18], WebGL compatible browsers are available on Android. Google Chrome is available as a beta version for Android 4.0+, but does not support WebGL yet [19].

Apple has added WebGL capabilities to iOS with iOS 4.2 [20], [21]. However, officially WebGL is only available to be used on Apples iAd platform [20], [22]. With an hack discovered by Nathan de Vries [20], WebGL can also be enabled in a UIWebView. Using this way it is possible to build a custom WebGL viewer for iOS. However, since this method of enabling WebGL on iOS requires the use of non-public APIs, it cannot be made available to end users due to Apple’s App Store Review Guidelines [23]. The built-in Safari browser on iOS does not support WebGL yet and due to Apples App Store Review Guidelines, third party browsers are required to use the WebKit rendering engine provided by Apple [23]. So until Apple officially supports WebGL in Safari on iOS, there is no way available for the end user to run WebGL content on iOS.

As the desktop version of Internet Explorer does not support WebGL [4] and Microsoft

considers WebGL "harmful" [24], we probably wont see WebGL support on Windows Phone 7 in the near future.

Also RIM's tablet, the BlackBerry PlayBook, supports WebGL in web applications [25].

2.1.3 Security Issues

Although the support of WebGL seems to be widespread, there are also critical voices, with Microsoft probably being their most prominent spokesperson. Microsoft believes "that WebGL will likely become an ongoing source of hard-to-fix vulnerabilities" [24], because WebGL allows direct access to the computers hardware from the web. WebGL security therefore relies on graphic card drivers and other third party components to mitigate the risks [24].

Context, an information security consultancy, which recommends disabling WebGL in the browser, published and demonstrated two possible attack scenarios initiated from a malicious website [26], [27]. They were not only able to perform a successful Denial of Service attack, which lead to the crashing of operating systems and freezes of desktops, but they were also able to gather confidential information by stealing the content of the graphic memory with which they were able to reconstruct screenshots of the desktop. As stated by Context, these issues are inherent to the WebGL specification and can't be resolved without major changes in WebGL's architecture. Although there are countermeasures in development, which could resolve these issues, at the moment WebGL allows malicious programs access to the graphics hardware and software. [26], [27]

2.1.4 Status Quo

Regarding usage, Khronos Group does not supply official numbers of internet users or web pages with WebGL support. There are unofficial trackers implemented in several webpages registering browser support for WebGL, e.g., WebGLStats. According to this tracker network, about 51.1 % of desktop users are using a WebGL enabled browser, whereas on mobile devices only 2.3 % are using WebGL enabled browser.

2.1.5 Evaluation criteria

In the literature, WebGL implementations on mobile devices have been technically evaluated using various methods and range from checking support of official WebGL desktop browser examples to typed array conformance tests to performance tests [4].

2.2 Augmented Reality

Augmented Reality (AR) can be defined as combining real and computer-generated digital information into the users view of the physical and interactive real world in such a way that they appear as one environment, thus providing a bridge between digital information and the physical world" [5], [10], [28], [29], [11].

New technologies have always had an influence on human information behavior, and will again with possible scenarios because of AR. For example, it may be possible to provide recommendation agents, which have been shown to reduce a consumers information overload and search complexity [30], directly in-store. Acquiring product information in in-store settings has often been linked to consumer decision making and information processing [31], [30], [32]. Besides cognitive factors, the users affection may be impacted as well; social aspects of mobile image recognition - attaching digital storytelling to physical products - have been shown to have an impact on a users affection [33], e.g. trust, engage consumers to communicate and receive information about products [31].

In a survey about the expectations and usage of augmented reality applications [5], people expected them to enhance their lives by providing them with up-to-date information about, and proactive functions to act upon context-relevant data which was not readily available before. Furthermore, users expected them to offer "stimulating and pleasant experiences, such as playfulness, inspiration, liveliness, captivation, and surprise". Applications dealing with practical problems were regarded higher than applications for pure entertainment.

The main negative aspects were information flood, user's loss of autonomy, and a possible

switch from real to virtual experiences and information [5].

Augmented reality is occasionally used interchangeably with virtual reality because of their similarities, but they have an essential key difference. According to Yi Wu, a senior research scientist in interaction and experience research at Intel Labs, the difference is that augmented reality never substitutes your view of the real physical world, it merely adds virtual information on top of the real world [34]. Virtual reality, however, has no real physical objects in view.

2.2.1 History

Although the notion of augmented reality has existed for about half a century, the term was not coined before 1990 by a researcher at aircraft manufacturer Boeing, Thomas Caudell [35]. The definition has been slightly modified over time, but Caudell applied the term to a head-mounted digital display that guided workers through assembling electrical wires in aircrafts [36].

One of the first devices using elements of augmented reality was a machine called Sensorama, built by Morton Helig [37]. This machine was designed to give the users a cinematic experience to more than just the visual. The demo film was a cycle ride through Brooklyn where you were supposed to get the feeling you were the one on the ride (this was done by vibrating ones seat, blowing wind in ones face, etc). These extra effects might be considered elements of augmented reality as, in the sense of the word, it augmented the reality of the experience [38].

At present, many current AR software, such as apps showing points-of-interest (POI) or product related informations, are considered representatives of augmented reality applications by users. A lot of these AR applications, however, are technically *pseudo-AR*, as not all applications align information properly on top of the view of the real world, or the augmented information is not truly 3D [35].

In April 2012 Google presented their vision of a future technology, called Project Glass [39]. This futuristic vision includes an augmented reality device in the form factor of glasses, which should cover the functionality of current smartphones (video conferencing, chatting, navigation, messaging, scheduling meetings etc.), but through constant wearing are possible to augment a users vision continuously. Despite the positive first impressions, there have already emerged some critics targeting mainly the possibility for excessive advertising and generally overwhelming users with information and distraction as well as concerns regarding the users privacy [40], [41].

2.2.2 Categories

AR applications can be categorized in two classes, *AR browsers* and *image recognition-based AR applications* [35].

AR browsers are applications that augment the real world - seen through the device's camera and other sensors - with digital information, usually from the web sources, such as graphic, links and other objects, similar to mashups [42]. Examples include touristic applications showing POI in the direction the device is oriented at, e.g., Layar (layar.com). This type of AR is usually called a magic lens in the literature [35].

Image recognition-based AR applications provide augmented information to everyday objects in the user's view based on visual recognition, which acts as a trigger for the digital information that is then presented, e.g., showing product information like prices and reviews [**?**] or search results [**?**]. Visual recognition can happen by the identification of markers like QR codes or by the identification of the objects themselves.

Given the two classes, the technical requirements for AR applications which can be deduced are: access to the devices video camera, image registration, positioning, orientation, and 2D and/or 3D image overlay.

2.2.3 Native apps and web applications

So far, augmented reality applications are almost exclusively *native* apps, i.e., applications written in the systems native programming language (e.g., Objective-C for iOS or Java for Android). The main reason is that up to recently it was not possible for the ordinary web

browser to access necessary system resources, such as the integrated video camera or the geolocation.

Lately, projects which target to resolve this issue begin to form; these are still in very early development phase, though. For example, WebRTC (WebRTC) is a new standard that enables real-time communication (RTC), i.e., the utilization of the video camera and microphone using Javascript and HTML5 for a range of applications such as video chat or voice calls. WebRTC is still in pre-alpha phase, and so far only one smartphone web browser supports this technology (Opera 12).

Augmented reality web apps are beginning to emerge, though, e.g., Argon (Argon).

2.2.4 WebGL usage

Real augmented reality applications using WebGL are rare, but tech demos begin to appear, given that access to a devices video camera is now possible, e.g., HTML5WebRTC.

Still in the beta phase, WebGL Earth webGLEarth is an open source software for visualizing maps similar to Google Earth. It relies purely on HTML5 canvas, WebGL and JavaScript, which is supported by most modern browsers. Nokia has also created a WebGL based software for visualizing maps nokiaMap. As an extra feature they have enabled stereoscopic 3D which, if you have 3D glasses, gives you the opportunity to view cities in 3D.

WebGL seems to spark the interest of game developers, too. Already now, there are games with the aesthetic value seen on many games distributed as a native application on mobile devices [43]. Seeing this trend, Opera Software wants to encourage new developers to this new paradigm [44].

2.2.5 Evaluation criteria

For AR evolution and adoption, user-oriented issues are critical [5]. AR research, however, still lacks evaluation methods" [45], and metrics are still very abstract [35]. Obstacles for evaluating the usability of mixed reality systems are manifold, and include a common testing platforms and benchmarks" [46].

Many researchers evaluate only early tech demos (cf. [47]); some evaluation oriented towards user experience based on HCI research exist [46], and include subjective ratings, e.g., user surveys with open questions.

2.3 UI and UX Research

User experience (UX) can be defined as a person's perceptions and responses that result from the use or anticipated use of a product, system or service [48]. In the literature, both instrumental and non-instrumental aspects are important, i.e., pragmatic elements like utility and subjective elements like pleasure and aesthetics [49].

Research on human-computer interaction (HCI) has been traditionally rooted in cognitive psychology, engineering, and computer sciences. Besides these fields, research on emotional factors of design is growing [50], which marks a shifts from *usability* to *user experience* analysis.

Academia evaluates user experience oftentimes by looking at the emotional state of the user [51], e.g., by letting users fill out pen-and-paper questionnaires during the course of the usage or by capturing user-made video diaries [52], [53].

3 Implementation

Given the two classes of augmented reality applications, tests for the following requirements had to be made: access to the devices video camera, image registration, positioning, orientation, and 2D and/or 3D image overlay. Furthermore, WebGL as a means for 3D-augmentation had to be tested.

To test the different requirements, we implemented a room in which you can look around and interact with several objects, like displaying additional information or starting a movie sequence. Geopositioning and orientation using the devices sensors are integrated as well as image overlays.

This section is structured as follows: we first describe the application which was implemented to use WebGL on iOS. Afterwards, we go on into detail elaborating which elements,

and their properties, populate the room. At the end of the section, we explain how we fetch the location of the user and how movement and viewing is done.

3.1 iOS WebGLViewer

As mentioned in section 2.1.2, there is no official WebGL support in a browser on iOS available at the time of writing. Therefore, a simple iOS application, called WebGLViewer, was implemented based on the method and instruction published by Nathan de Vries [20]. WebGLViewer consists of a UIWebView, a reload button and an address bar. The UIWebView was modified to show WebGL content (compare section 2.1.2 and [20]).

If the device is shaken while running WebGLViewer the application shows information about the battery consumption. The current battery level, the level when the loading of the current page is finished, as well as the delta of these two values is shown as percentage. In addition, the time period the website is loaded as well as the battery consumption per minute are shown. Unfortunately the battery level is only updated in 5 % steps. To get a more accurate result, the battery information pop-up will automatically be shown when the battery level changes.

The application *WebGLViewer* was used to view and evaluate WebGL content on iOS.

3.2 Room

Our application, *Room.html*, is build with the goal of demonstrating the possibilities and limitations of WebGL on mobile devices for the purpose of this paper. The web application in question, the Room.html, is available on oslo.paulsteinhilber.de/room.html.

Figure 1 shows a screenshot of room.html running in Chrome on a desktop computer, while playing the video. Figure 2 shows a screenshot of room.html running in the WebGLViewer on an iPhone.

The environment consists of a room with windows and a door (made of planes with textures) as well as several static and dynamic objects, i.e., chairs, tables, a blackboard, and a cube. Interaction is possible with some of them

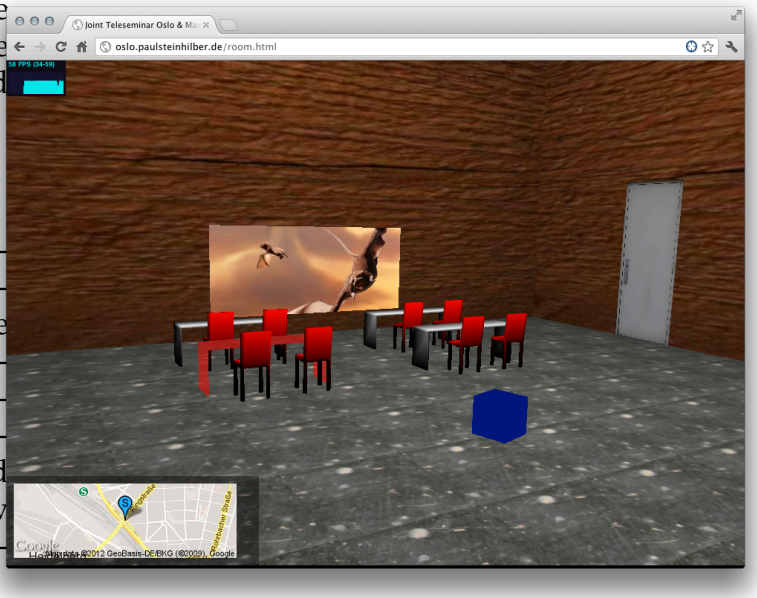

Fig. 1: Room.html with playing video in Chrome on a desktop computer

and they perform certain actions when clicked on or touched, i.e., the cube begins to rotate, the blackboard plays a video and a table displays further information.

The room is built using 6 planes put together to simulate a cube. Each plane has 4 vertices creating a single polygon. The blackboard is also of the simple sort. It is just a single plane and a single polygon. To make the windows and the door we added three additional planes with a texture, thus making the total number of polygons 9.

3.2.1 Objects

Objects in WebGL can both be build from scratch using basic shapes or can be imported from 3D modeling software. For our room we decided to use the tools provided by Three.js, a JavaScript 3D-library [54], to create the primitives (basic geometric shapes such as a plane or a cube) and use Blender [55], an open source 3D application, for objects of greater complexity (such as the chairs and tables).

The blackboard is a simple plane consisting of 4 vertices which together create a polygon. As a texture for the blackboard we added a video element. The user can touch to interact with the blackboard which plays or pauses the

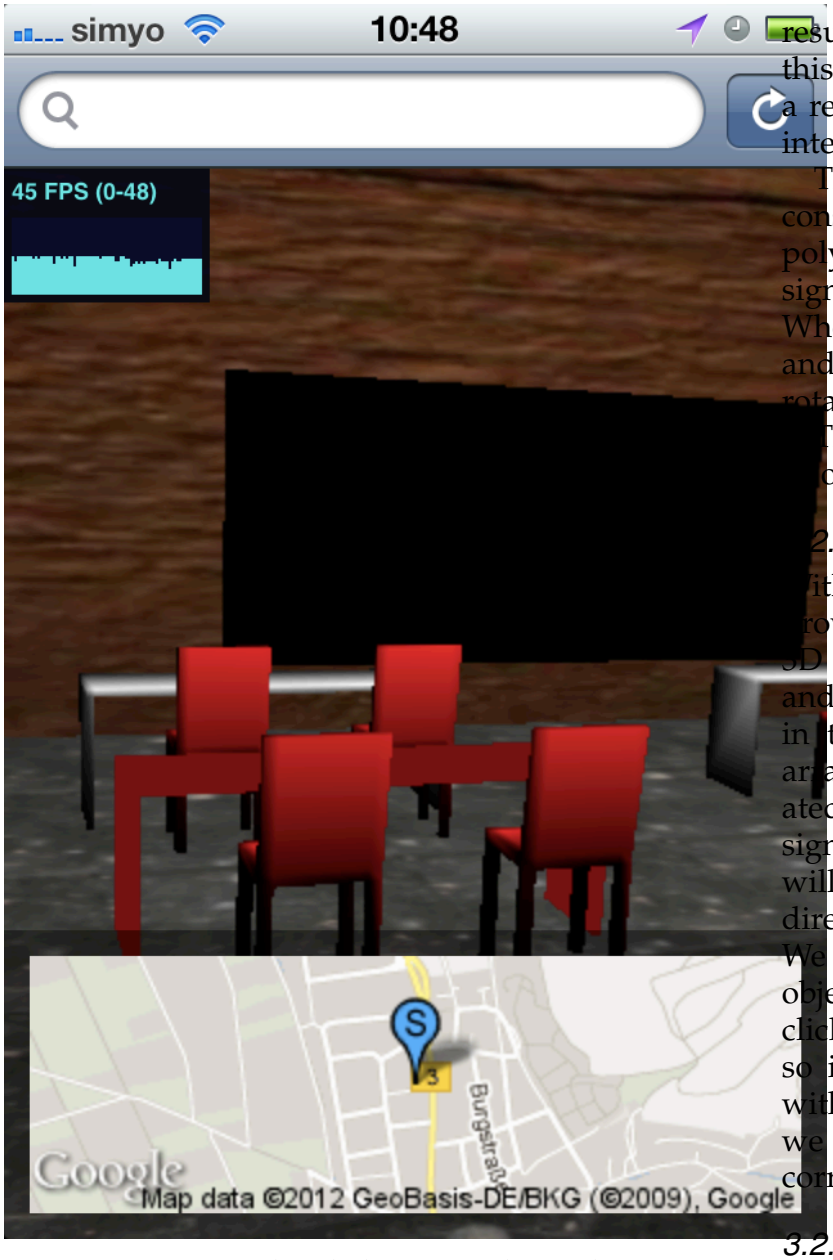

Fig. 2: Room.html shown in the WebGLViewer on an iPhone

video.

A single table contains 24 vertices which creates 22 polygons. Considering that our scene contains 4 tables, this means that the desks are responsible for 88 polygons. They all have a grey lambert material assigned with the exception of one table, which has a red basic material. This is to clearly show that it differs from the others and is interactive. When this table is touched a box appears explaining where such a table could be purchased.

A single chair consists of 56 vertices which results to 57 polygons. We have 8 chairs so this amounts to 456 polygons. All chairs have a red lambert material assigned, as no chair is interactive.

The cube is natively created in Three.js and consists of 8 vertices which amounts to 6 polygons. A basic red shader has been assigned similar to the other interactive objects. When the cube is touched it starts rotating and changes its color. Another touch stops the rotation and changes color one more time.

This gives us a grand total of 560 polygons in our application.

3.2.2 Interaction with Objects

With the *Ray* method/class the *Three.js* library provides a simple method for interacting with 3D objects [56]. By specifying a starting point and the direction of the *Ray*, any objects caught in the path of the ray will be added to an array. To identify these objects we have created an additional array holding all our "assigned" interactable objects. The starting point will always be the camera position and the direction is related to the mouse click or touch. We are only interested in the first interacted object (or else, in a larger project, we might be clicking on something in a separate room) and so it is the only object we bother comparing with our assigned interactable objects. When we have identified the object we perform the corresponding function.

3.2.3 Textures

Textures in the implementation were imported using *THREE.ImageUtils.loadTexture()*, which creates a material based on the texture and combines it with the object. Although compressed textures are technically possible and are important for large applications, this technique was not used in this tech demo.

3.2.4 Accessing Geolocation

To demonstrate that it is possible to access the physical position of a mobile device with JavaScript we show a map of the devices location at the bottom of the screen.

To get the devices location, the W3C Geolocation API [57] was used. This API is independent from the underlying method to get the

location. The location can be obtained from, for example GPS, or can be inferred from network signals such as IP address, RFID, WiFi and Bluetooth MAC addresses, and GSM/CDMA cell IDs [57].

3.2.5 Moving and Looking around in the Room

Although data from a gyroscope is accessible with JavaScript and could be therefore used in a WebGL application, we have only used data provided by a devices accelerometer, since not all of our test devices had a gyroscope included. The JavaScript *onDeviceMotion*-event is used to get the accelerometer data. On iOS, the accelerometer measures the sum of two acceleration vectors: gravity and user acceleration [58]. Using this data we were able to implement the feature to look up and down by moving the device. It is implemented in a way, that if the device is laying flat on a surface it shows the floor of the room. When holding the device vertically in your hand, you are looking at the blackboard.

The room application is reacting to touch events. You can swipe left or right to look to the left or the right, respectively.

If the application is run from a browser on a stationary machine or a laptop, it is possible to move and look around in the room with the keyboard (W A S D and arrow keys). Alternatively, by pressing the left mouse button and while holding it pressed, moving the mouse, you can look around the room. Implementation of keyboard movements was done via one of *Three.js* native movement classes since our endeavours to do so directly were renderd impossible by the lack of both documentation and complete implementation of the library http://mrdoob.github.com/three.js/docs/48/#FlyControls.

4 Evaluation

For the purpose of presenting the findings in our work in a structured and readily manner, we are going to devise a test matrix for our results. Since the WebGL technology is still relatively young and the support for it varies from browser to browser and from device to device, it would be very hard to devise common criteria for all the devices. Therefore, we are approaching the problem by presenting the results separately for each device in the form Device - OS - Browser. In that way we can outline our observations in a much structured way and get to stress the peculiarities of WebGL support on different devices.

Since our goal is to investigate WebGL on mobile devices, we choose one desktop computer as a reference. Which OS (Mac OS, Windows or Linux) and particular machine (Mac or PC, laptop or a stationary) is used is not that important, since all the modern computers should be powerful enough to render WebGL graphics. The important thing is to choose a browser that fully supports WebGL. We find that Google Chrome is a good choice for that. All the measurements and observations will be taken running the *Room.html*, as described in section 3.2, developed for the purpose of this paper.

Parameters we are observing are, for the most part, technical in nature. As the part of our application we are measuring Frames Per Second (FPS), parameter that gives us the number of times browser refreshes the screen per one second. In addition to the FPS we are also taking into consideration parameters like CPU usage and battery consumption, but as noted before the different parameters will be mentioned there it is applicable.

Furthermore, in a similar approach to Golubovic *et al.* [4], common functionality of WebGL is tested by executing the lessons of the WebGL tutorial from learningWebGL.com [59]. These lessons are based on a popular OpenGL tutorial (*NeHe*, nehe.gamedev.net/) and cover most of the common functions, and should be enough to put our target devices to the test. The results of these tests is presented in table 3.

In addition to technical parameters another interesting area of evaluation of WebGL is user experience. But since the most valid and exhaustive results in that field come from a large user surveys [5], we are not going to do any work in that area.

4.1 Energy Consumption

To measure the battery usage, the WebGL-Viewer application for iOS, as described in section 3.1, was used. The device was fully charged. The accordant content was opened in the WebGLViewer application. The device was then unplugged and the content reloaded. The time until the battery level decreased to 90 % as well as the battery consumption per minute was measured and calculated by the WebGLViewer. Using Apple Instruments, a part of Apples Developer Tools [60], we measured the *CPU Activity* (*Total Activity*, *Foreground App Activity* and *Graphics*) as well as the relative *Energy Usage* on a scale from 0 to 20. All tests have been performed on an Apple iPhone 4 with iOS 5.1 installed using the WebGLViewer. We have compared the battery usage and CPU activity of the implemented *Room.html*, the *Quake 3 WebGL Demo* by Brandon Jones [61] and google.com as reference. The results are shown in table 1.

Comparing the two WebGL applications to a normal website like google.com shows a huge difference in CPU activity. The overall CPU activity is below 10 %, with a value below 1 % for Foreground App Activity as well as *Graphics Activity*, while using google.com, whereas the overall CPU activity is over 70 %, with about 60 % for *Foreground App Activity* and up to 25 % for *Graphics Activity*, while using a WebGL application.

If we look at the battery consumption there is also a significant difference visible. The battery drains twice as fast while using a WebGL enabled website compared to google.com. The difference regarding the *Relative Energy Usage* is not as big, however, we don't know how Apple calculates this value. google.com still needs less energy compared to the WebGL applications. Also the room.html has a higher *Relative Energy Usage* compared to the Quake 3 WebGL Demo, which is consistent with our measurement of the battery usage, as well as with the time until the battery level decreases to 90 %.

4.2 Frames per second

To test the performance of WebGL on different platforms we have measured and compared

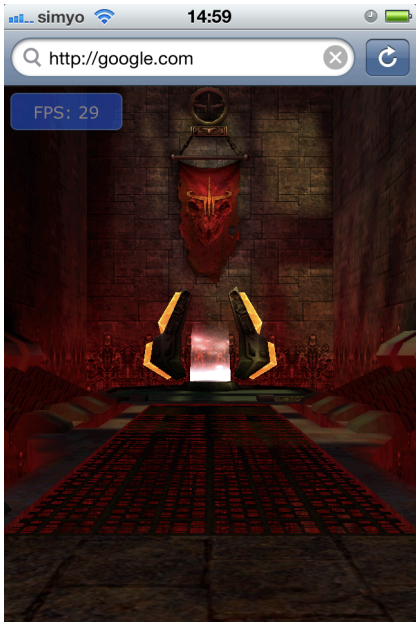

Fig. 3: Quake 3 WebGL Demo [61] shown in the WebGLViewer on an iPhone

frames per second (FPS) for different applications. FPS describes the frequency at which images are generated. The values below 10-12 FPS, are recognizable by the human eye as the separate images; for higher FPS, single images cannot be recognized and they blend together creating motion [62]. So a higher number of FPS creates a more fluid animation and is therefore considered better. Since the refresh rate of modern flat screens is 60 FPS there is usually no need to render with more than 60 FPS [63].

In table 2 we compared the FPS of the implemented *Room.html*, the *Quake 3 WebGL Demo* [61] (as shown in figure 3), and an example of a spinning cube, the *SpiritBox* from webkit.org [64], on different mobile devices as well as on one laptop computer (2010 MacBook) and stationary computer (2006 MacPro) as reference.

The lessons from learningWebGL.com [59] have been used as a benchmark to test the performance on different mobile device. The results are shown in table 3.

Opera has advertised the WebGL capabilities beginning with Opera Mobile 12 [18], but the performance of WebGL content on this browser is much worse than in Firefox. Although all lessons from table 3 are rendered correctly, the frame rate was always below 10 FPS, compared with 16 FPS in average in Firefox on the HTC

Test Case	Battery and Energy Consumption			CPU Activity		
	Time till 90 %	Battery Usage	Relative Energy Usage	Total	Foreground App	Graphics
room.html	36.03 min	0.278 $\frac{\%}{min}$	17	100 %	64 %	20 %
Quake 3	34.67 min	0.289 $\frac{\%}{min}$	16	70 %	60 %	8 %
google.com	78.93 min	0.127 $\frac{\%}{min}$	11	6 %	0.2 %	0.5 %

TABLE 1: Battery consumption and CPU activity of different WebGL applications

Device	Operating System	Browser	Launched	room.html	Quake 3	SpiritBox
Apple iPad 2	iOS 5.1	WebGLViewer	2011	61 FPS	61 FPS	60 FPS
HTC EVO 3D	Android 2.3.4	Firefox	2011	8 FPS	12 FPS	N/A [1]
Apple iPhone 4	iOS 5.1	WebGLViewer	2010	40 FPS	29 FPS	43 FPS
Apple iPod Touch (4th Gen.)	iOS 5.1	WebGLViewer	2010	21 FPS	25 FPS	43 FPS
HTC Desire	Android 2.2.2	Firefox	2010	2 FPS	N/A	8 FPS
Apple iPhone 3GS	iOS 5.1	WebGLViewer	2009	36 FPS	27 FPS	60 FPS
Reference Laptop [2]	Mac OS X 10.7.3	Google Chrome	2010	34 FPS	36 FPS	50 FPS
Reference Computer [3]	Mac OS X 10.7.3	Google Chrome	2006	58 FPS	59 FPS	85 FPS

TABLE 2: Frames per Second (FPS) of different WebGL applications on different devices
[1] *The FPS value is constantly alternating between values in the range from 10 FPS up to over 200 FPS, making it impossible to determine a realistic value.*
[2] *Reference Laptop: MacBook 2010, Mac OS X 10.7.3, 2.26 GHz Intel Core 2 Duo, 4 GB 1067 MHz DDR3 RAM*
[3] *Reference Computer: MacBook 2006, Mac OS X 10.7.3, 2x 2.0 GHz Dual-Core Intel Xeon, 6 GB 667 MHz DDR2 RAM*

Lesson	Reference[1]	Desire	EVO 3D	iPhone 4	iPhone 3GS	iPad 2	iPod Touch
3 - Simple animations	59 FPS	13 FPS	16 FPS	57 FPS	57 FPS	58 FPS	40 FPS
4 - 3D animations	58 FPS	12 FPS	19 FPS	57 FPS	57 FPS	58 FPS	57 FPS
5 - Textures	59 FPS	13 FPS	16 FPS	40 FPS	57 FPS	58 FPS	40 FPS
6 - Texture filters and keyboard	59 FPS	12 FPS	18 FPS	40 FPS	57 FPS	58 FPS	40 FPS
7 - Basic Lighting	59 FPS	12 FPS	17 FPS	57 FPS	57 FPS	58 FPS	57 FPS
8 - Transparency and blending	59 FPS	11 FPS	16 FPS	58 FPS	57 FPS	58 FPS	58 FPS
9 - Particles	59 FPS	8 FPS	17 FPS	39 FPS	31 FPS	60 FPS	40 FPS
10 - Loading a map	58 FPS	11 FPS	15 FPS	57 FPS	57 FPS	58 FPS	40 FPS
11 - Sphere and rotation	59 FPS	12 FPS	15 FPS	40 FPS	57 FPS	58 FPS	57 FPS
12 - Point lighting	58 FPS	11 FPS	15 FPS	40 FPS	57 FPS	58 FPS	40 FPS
13 - Per-fragment lighting	59 FPS	10 FPS	17 FPS	36 FPS	40 FPS	58 FPS	39 FPS
14 - Specular highlights and JSON model	58 FPS	9 FPS	15 FPS	35 FPS	39 FPS	58 FPS	36 FPS
15 - Specular maps	58 FPS	8 FPS	16 FPS	22 FPS [2]	24 FPS [2]	58 FPS [2]	22 FPS [2]
16 - Render to texture	58 FPS	6 FPS	16 FPS	22 FPS	22 FPS	58 FPS	22 FPS

TABLE 3: Frames per Second (FPS) using the lessons from the learning WebGL tutorial on different devices
[1] *Reference Computer: MacBook 2010, Mac OS X 10.7.3, 2.26 GHz Intel Core 2 Duo, 4 GB 1067 MHz DDR3 RAM*
[2] *Rendering results aren't looking as expected*

EVO 3D. The *SpiritBox* on the other hand is not rendered correctly (only a blue square is visible) and the room.html doesnt load at all.

The most obvious result is the big performance difference between the iOS and Android devices. Even the HTC EVO 3D, which was launched in 2011, was not able to deliver results comparable to the iPhone 3GS launched in 2009. On the Android devices the *Room.html* was unusable to interact with fluidly, whereas it was rendered smoothly on all iOS devices.

While the Android devices have always had lower frame rates compared to the iOS devices, they were able to render all lessons from table 3 correctly. iOS based devices, on the other hand, were not able to render lesson 15 correctly. Screenshots to compare the rendering of lesson 15 on Android and iOS are shown in figure 4

and figure 5, respectively.

Another interesting result is, that the iPhone 3GS delivers significantly higher frame rates in some tests compared to the iPhone 4. An possible explanation could be, that the iPhone 4s display resolution is significantly higher, thus requiring the calculation of four times more pixels.

Comparing the performance of our reference machine, the 2010 MacBook, with the iOS based mobile devices is a indicator that today's mobile devices are powerful enough to render 3D applications with an acceptable frame rate.

4.3 Limitations

Albeit most of the tests we have performed were successful, we have still encountered a few limitations during our test phase. We are listing them, further down the text.

4.3.1 Device Sensors Access

WebRTC, an upcoming standard for access to the devices video camera and microphone, is currently only available on Opera Mobile 12, a browser currently not supported on iOS. With the Android devices lacking in performance, we were unable to implement a WebRTC application. During our evaluation and testing on other devices, we realised that the accelerometer wasnt working as intended on Android using a HTC EVO 3D running the *Room.html* in Firefox. When the device is laying flat on a surface, the rooms floor is shown as expected. Holding it vertically in your hand, however, still shows the floor. The accelerometer is undoubtedly working, but it seems to be much less sensitive, than the one on iOS based devices. Therefore, the feature to look up and down with the accelerometer is deactivated, when non-iOS device is detected.

4.3.2 Key and Mouse Events

As expected, key and mouse events cannot be triggered in WebGL on mobile devices with touch screens, but replacing them with touch events is a working solution for this problem.

4.3.3 Video texture

Touching the blackboard should start playing a movie on it. It worked perfectly on desktop computers using Windows, Mac and Linux as operating systems and Chrome, Safari and Firefox as browsers. We were unable to get the video texture working correctly on the mobile devices, though.

Namely, neither audio nor video were reproduced on the mobile devices after touching the blackboard, with the exception of working audio on the iPad 2. Using an iPhone 4 a significant decrease of the frame rate could be observed after tapping the blackboard. The frame rate dropped from 47 FPS to just 20 FPS after touching the blackboard. While measuring the CPU activity, after tapping the blackboard, the *Foreground App Activity* dropped from about 60 % to 30 % while the *Audio Processing* increased from 0 % to about 10 %, with an overall CPU activity of 100 % on an iPhone 4.

The video played perfectly using just the HTML5 video tag on iOS using Safari or WebGLViewer, hence a wrong video encoding could be eliminated as source of the problem. Since the video texture works on all browsers and all desktop operating systems, and the videos audio is even played on the iPad 2, we are assuming that our implementation is correct.

Therefore we think, that video texture not working for us is actually a limitation of WebGL on mobile devices.

5 Conclusion

In this paper we developed an application with the purpose of testing the feasibility of augmented reality (specifically 3D-augmentation) on mobile devices using only a web browser.

The technical possibilities for accessing the devices sensors for interacting with the users context, crucial to develop AR applications, are now given, even though in an early phase. One of the most common properties associated with augmented reality is camera integration, and although it is technically possible to realize this using pre-alpha software, due to the limitations of the mobile devices we tested (as mentioned in section 4.3.1), we were not able

to test augmented reality in a browser using a camera. Other requirements for augmented reality applications on mobile devices were successfully shown.

As we can see from the results, there was a huge difference between iOS and Android with regards to performance, but both seemed to handle the task, though with certain limitations as discussed previously. The application renders on all our devices, and some basic user interaction is possible. Compared to a reference machine we saw that the rendering cycle expressed through the FPS is a bit lower on mobile devices, but still good (with exception of Apples iPad 2, whose frame rate is comparable to the one on our reference machine). Furthermore, we observed that battery consumption is relatively high while running WebGL applications compared to ordinary web-surfing.

Our results, presented in this paper, indicate that WebGL at its current state can be used on mobile devices to augment 3D objects to a users view. The standards for AR applications in web browsers are not yet completed, and implementations are not stable yet, which makes a widespread use nearly impossible at this moment. However, if Moores law [66] applies to the progression of mobile devices as well as web standards, and mobile software continue to follow its current pace, we can be sure that augmented applications in web browsers could be ubiquitous on mobile devices in the near future.

INSTALLING WEBGLVIEWER

To install the WebGLViewer using the provided source code on an iOS device a membership in Apple's iOS Developer Program is needed. However compiling and running the WebGLViewer in the iOS Simulator is possible using XCode. As alternative, a registered developer can provide you a binary, which is build for your specific device. We used testflight.com to distribute such binaries. To request a binary you need to create an account on http://bit.ly/zWoZQJ and register your device as soon as you are accepted.

APPENDIX B

A screenshots of SpiritBox at 6.

A screenshots of ”Lesson 9” at 7.

REFERENCES

[1] A. Taivalsaari, T. Mikkonen, M. Anttonen, and A. Salminen, “The death of binary software: End user software moves to the web,” in *2011 Ninth International Conference on Creating, Connecting and Collaborating through Computing*, 2011, pp. 17–23.

[2] March 2012. [Online]. Available: https://docs.google.com

[3] March 2012. [Online]. Available: http://280slides.com/Editor/

[4] D. Golubovic, G. Miljkovic, S. Miucin, Z. Kaprocki, and V. Velisavljev, “Webgl implemenation in webkit based web browser on android platform,” in *Telecommunications Forum (TELFOR)*, 2011, pp. 1139–1142.

[5] T. Olsson and K. Väänänen-Vainio-Mattila, “Expected user experience with mobile augmented reality services. workshop of mobile augmented reality,” in *MobileHCI 2011*, 2011.

[6] W. L. D. Lui, D. Browne, L. Kleeman, T. Drummond, and W. H. Li, “Transformative reality: Augmented reality for visual prostheses,” in *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[7] F. Mannuß, J. Rubel, C. Wagner, F. Bingel, and A. Hinkenjann, “Augmenting magnet field lines for school experiments,” in *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[8] G. Liestol, “Learning through situated simulations: Exploring mobile augumented reality,” in *(ECAR Research Bulletin 1, 2011) Boulder, CO: EDUCAUSE Center for Applied Research, 2011, available from http://www.educause.edu/ecar*, 2011, pp. 1–14.

[9] A. Mulloni, H. Seichter, and D. Schmalstieg, “User experiences with augmented reality aided navigation on phones,” in *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[10] T. Höllerer and S. Feiner, “Mobile augmented reality,” in *Telegeoinformatics: Location-Based Computing and Services*. Taylor and Francis Books Ltd, 2004.

[11] P. Wellner, W. Mackay, and R. Gold, “Back to the real world,” *Communications of the ACM*, vol. 36, no. 7, pp. 24–26, 1993.

[12] March 2012. [Online]. Available: http://www.mobilephonedevelopment.com/archives/1149

[13] March 2012. [Online]. Available: http://blog.nielsen.com/nielsenwire/?p=29786

[14] S. Wellman, March 2012. [Online]. Available: http://www.informationweek.com/blog/229215818

[15] March 2012. [Online]. Available: http://blogs.adobe.com/flashplatform/2011/11/flash-to-focus-on-pc-browsing-and-mobile-apps-adobe-to-more-aggr
html

[16] March 2012. [Online]. Available: http://www.khronos.org/news/press/releases/khronos-releases-final-webgl-1.0-specification

[17] April 2012. [Online]. Available: http://www.khronos.org/webgl/wiki/WebGLandOpenGLDifferences

[18] 2012. [Online]. Available: http://my.opera.com/chooseopera/blog/2012/02/27/opera-mini-7-next-and-opera-mobile-12

[19] 2012. [Online]. Available: https://developers.google.com/chrome/mobile/docs/faq
[20] November 2011. [Online]. Available: http://atnan.com/blog/2011/11/03/enabling-and-using-webgl-on-ios/
[21] November 2011. [Online]. Available: https://developer.apple.com/library/iad/documentation/UserExperience/Conceptual/iAd_Design_Guide/iAd_Design_Guide.pdf
[22] June 2011. [Online]. Available: https://www.khronos.org/webgl/public-mailing-list/archives/1106/msg00036.html
[23] 2012. [Online]. Available: https://developer.apple.com/appstore/resources/approval/guidelines.html
[24] June 2011. [Online]. Available: http://blogs.technet.com/b/srd/archive/2011/06/16/webgl-considered-harmful.aspx
[25] 2012. [Online]. Available: http://devblog.blackberry.com/2012/02/playbook-native-webgl-development/
[26] May 2011. [Online]. Available: http://www.contextis.com/resources/blog/webgl/
[27] June 2011. [Online]. Available: http://www.contextis.com/resources/blog/webgl2/
[28] E. Klopfer and K. Squire, "Environmental detectives - the development of an augmented reality platform for environmental simulations," *Educational Technology Research and Development*, vol. 56, no. 2, pp. 203–228, 2007.
[29] J. Vallino, "Interactive augmented reality," Ph.D. dissertation, University of Rochester, 1998.
[30] T. Kowatch and W. Maass, "In-store consumer behavior: how mobile recommendation agents influence usage intentions, product purchases, and store preferences," *Computers in Human Behavior*, vol. 26, no. 4, pp. 697–704, July 2010.
[31] S. Karpischek and F. Michahelles, "my2cents - digitizing consumer opinions and comments about retail products," in *Proc. of Internet of Things (IOT)*, 2010.
[32] B. Xiao and I. Benbasat, "E-commerce product recommendation engines: use, characteristics, and impact," *Management of Information Systems Quarterly*, vol. 31, no. 1, 2007.
[33] R. Barthel, A. Hudson-Smith, M. Jode, and B. Blundell, "Tales of things. the internet of 'old' things: collecting stories of objects, places and spaces," in *Proc. of the Urban Internet of Things*, 2010.
[34] 2012. [Online]. Available: http://curiosity.discovery.com/question/augmented-virtual-reality
[35] T. Olsson and M. Salo, "Online user survey on current mobile augmented reality applications," in *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, October 2011, pp. 75–84.
[36] [Online]. Available: http://www.wired.com/gadgetlab/2009/08/augmented-reality/
[37]
[38]
[39] 2012. [Online]. Available: https://g.co/projectglass
[40] 2012. [Online]. Available: http://abcnews.go.com/blogs/technology/2012/04/google-glasses-will-you-want-google-tracking-your-eyes/
[41] 2012. [Online]. Available: http://www.washingtonpost.com/blogs/arts-post/post/google-project-glass-cool-or-creepy/2012/04/05/gIQArkAQxS_blog.html
[42] D.Schmalstieg, T.Langlotz, and M.Billinghurst, "Augmented reality 2.0," in *Virtual Realities*. Springer, Vienna, 2011.
[43]
[44]
[45] M. G. et al., "Experience with an ar evaluation test bed: presence, performance, and physiological measurement," in *ISMAR 2010*, 2010, pp. 127–136.
[46] C. Bach and D. L. Scapin, "Obstacles and perspectives for evaluating mixed reality systems usability," in *Proceedings of the IUI-CADUI Workshop on Exploring the Design and Engineering of Mixed Reality Systems (MIXER)*, 2004.
[47] A. Dünster, R. Grasset, and M. Billinghurst, "A survey of evaluation techniques used in augmented reality studies," in *Proc. ACM SIGGRAPH 2008*, 2008.
[48] I. organization for standardization, "Iso fdis 9241-2010:2009. ergonomics of human system interaction - part 210: Human-centred design for interactive systems (formerly known as 13407)."
[49] M. Hassenzahl and N. Tractinsky, "User experience - a research agenda," *Behaviour and Information Technology*, vol. 25, no. 2, pp. 91–97, 2006.
[50] D. A. Norman, "Emotion and design: Attractive things work better," *Interactions*, vol. 9, no. 4, 2002.
[51] V. Roto, "Web browsing on mobile phones - characteristics of user experience," Ph.D. dissertation, Helsinki University of Technology, 2006.
[52] M. Csikszentmihalyi and R. Larson, "Validity and reliability of the experience-sampling method," *Journal of Nervous and Mental Diseases*, vol. 175, no. 9, pp. 526–536, September 1987.
[53] M. Isomursu, K. Kuutti, and S. Väinämö, "Experience clip: method for user participation and evaluation of mobile concepts," in *Proceedings of the eighth conference on Participatory design: Artful integration: interweaving media, materials and practices*, 2004, pp. 83–92.
[54] 2012. [Online]. Available: https://github.com/mrdoob/three.js/
[55]
[56] 2012. [Online]. Available: http://threejsdoc.appspot.com/doc/three.js/src.source/core/Ray.js.html
[57] 2012. [Online]. Available: http://dev.w3.org/geo/api/spec-source.html
[58] 2011. [Online]. Available: https://developer.apple.com/library/ios/#documentation/CoreMotion/Reference/CMDeviceMotion_Class/Reference/Reference.html#//apple_ref/occ/cl/CMDeviceMotion
[59] 2012. [Online]. Available: http://learningwebgl.com/blog/?page_id=1217
[60] 2012. [Online]. Available: https://developer.apple.com/technologies/tools/
[61] 2012. [Online]. Available: http://media.tojicode.com/q3bsp/
[62] P. Read and M.-P. Meyer, *Restoration of Motion Picture Film (Butterworth-Heinemann Series in Conservation and Museology)*. Butterworth-Heinemann, 2000.
[63] 2012. [Online]. Available: http://robert.ocallahan.org/2010/11/measuring-fps_26.html
[64] 2012. [Online]. Available: http://www.webkit.org/blog-files/webgl/SpiritBox.html
[65] April 2012. [Online]. Available: http://learningwebgl.com/blog/?p=1778
[66] R. R. Schaller, "Moore's law: past, present and future," *IEEE Spectrum*, vol. 34, no. 6, pp. 52–59, June 1997.
[67] April 2012. [Online]. Available: http://learningwebgl.com/blog/?p=1008

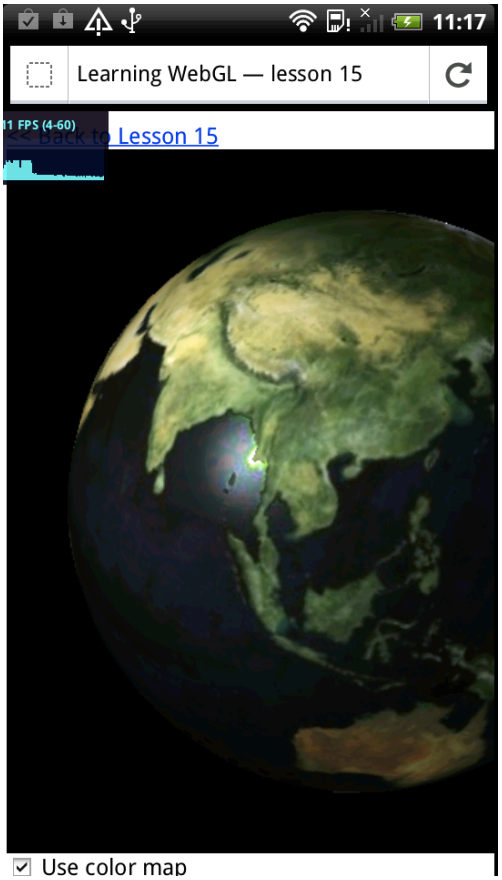

Fig. 4: Lesson 15 WebGL demo on Android (adapted from [65])

Fig. 5: Lesson 15 WebGL demo on iOS (adapted from [65])

Fig. 6: WebGL demo SpiritBox by [64]

Fig. 7: Lesson 9 WebGL demo (adapted from [67])