Utrecht University for Applied Sciences

Digital Communication & Media

Faculty for Communication & Journalism

WebGL Product Configurators

From Technical and Market Viability to Alpha

Author:
RWJ Peelen

Supervisor:
Dhr. Kees Winkel

Jan. 2016

1 Abstract

Peppr is a company that is specialized in building photo-realistic visualizations. Late 2014, Peppr built a product configurator for SlimFitted, a company that builds tailored shirts. They wanted their customers to be able to design their own shirts.

Current product configurators for the web are built by splitting up the product into different layers. Every layer consists of a pack of images. Which, in Peppr's case, were 25 colors, 2 perspectives, 7 collars, 6 sleeves and 3 base shirts. This left Peppr with a sum of 6300 different layers (and thus, images) and when the client wants to add another color, they would have to build another set of 252 images. This is a timely and costly venture.

Peppr concluded the usual way of doing these types of projects is suboptimal and started looking for an alternative. That is where this thesis comes to play. March 2011 was the first release of WebGL (https://en.wikipedia.org/wiki/WebGL), an implementation of OpenGL technology for the web. Because WebGL renders directly from the video processor, it opens up the web to a whole new way of using 3d. The actual adoption rate has always been low because only the latest browsers would integrate the technology. Anno 2015 though, the playing field has changed. With more-and-more browsers supporting this new type of technology, the timing might be perfect to bring it to the masses.

In this thesis I will try to find if a WebGL based product configurator, is both viable in terms of technical and marketing aspects.

Contents

2 Theoretical Framework

2.1 Introduction

A couple of subjects need to be touched to set a proper perspective. Starting with the underlying technology that might make these kind of projects succeed; WebGL and OpenGL. After that there are numerous market specific issues we need to address, starting with the current state of product configurators, Gartners Hypecycle (and wether or not he has spoken about such hypes) and finally, the method we will be using to analyse all these domains, on which we shall base our conclusion.

2.2 OpenGL & WebGL

" WebGL is an application programming interface (API) for advanced 3d graphics on the web. It is based on OpenGL ES 2.0, and provides similar rendering functionality, but in an HTML and JavaScript context. The rendering surface that is used for WebGL is the HTML5 canvas element, which was originally introduced by Apple in the WebKit open-source browser engine. The reason for introducing the HTML5 canvas was to be able to render 2D graphics in applications such as Dashboard widgets and in the Safari browser on the Apple Mac OS X operating system. Based on the canvas element, Vladimir Vukicevic at Mozilla started experimenting with 3d graphics for the canvas element. He called the initial prototype canvas 3d. In 2009 the Khronos Group started a new WebGL working group, which now consists of several major browser vendors, including Apple, Google, Mozilla, and Opera. The Khronos Group is a non-profit industry consortium that creates open standards and royalty-free APIs. It was founded in January 200 and is behind a number of other APIs and technologies such as OpenGL ES for 3d graphics for embedded devices, OpenCL for por parallel programming, OpenVG for low-level acceleration of vector graphics and OpenMAX for accelerated multimedia components. Since 2006 the Khronos Group has aso controlled and promoted OpenGL, which is a 3d graphics API for desktops. The final WebGL 1.0 specification was frozen in March 2011, and WebGL support is implemented in several browsers, including Google Chrome, Mozilla Firefox, and (at the time of this writing) in the development releases of Safari and Opera. " [1]

At the time of Andreas' writing, WebGL was very much in its early development stage. Anno 2015 though, WebGL has found its way into mainstream browsers with Internet Explorer and Android support lacking for versions earlier than its absolute latest [2]. It currently has a total adoption rate of nearly 81%. The most interesting part of using WebGL is the "low-level acceleration". Which means that on the stack of available hardware and software to render graphics, WebGL and OpenGL are on the lower ones (or closer to hardware level). This makes using WebGL and OpenGL insanely fast when compared to software / browser acceleration, which is very high-level. The main bottleneck for using said low-level acceleration, is that when there is no low-level graphics processing unit (GPU) available, the computing needs to be done with high-level software, which is ultimately directed (through several levels of architecture) to the CPU of the device and some tasks are great for CPU calculations, however, some are not.

" Both CPU and GPU possess distinct architectural features. Modern multi-core CPUs use up to a few tens of cores, which are typically out-of-order, multi-instruction issue cores. Also, CPU cores run at high-frequency and use large sized caches to minimize the latency of a single-thread. Clearly, CPUs are suited for latency-critical applications. In contrast, GPUs use much larger number of cores, which are in-order cores that share their con-trol unit. Also, GPU cores use lower frequency,and smaller sized caches[Mittal 2014a].Thus, GPUs are suited for throughput-critical applications. " [4]

What is stated by Mittal and Vetter is the very basic, low-level difference between a CPU and GPU. The CPU is designed to handle any calculation thrown at it, while the GPU is designed to be more special purpose. The importance of the CPU vs GPU is that WebGL makes use of the low-level acceleration of the GPU and more importantly, most modern smartphones these days have a GPU, making a potential adoption rate for the configuration platform much higher as end-users might well be able to use them on the go as well.

2.3 Current State of Product Configurators

3 Introduction

3.1 Problem Statement

Here I'll try to state problems per domain as widely as possible.

3.1.1 Service domain

3.1.2 Technical domain

3.1.3 Organizational domain

3.1.4 Financial domain

3.2 Objectives

Here I'll try to deform those problem statements to actual researchable objectives.

3.2.1 Main Objective

Broad Research question

3.2.2 Service objectives

3.2.3 Technical objectives

3.2.4 Organizational objectives

3.2.5 Financial objectives

4 Plan of Action

4.1 Research Methodology

Short introduction as to why this section exists

4.2 Assignment of Methodology

Certain aspects of the objectives of the research need certain types of research. I'll assign them here while going into detail in the subsections below.

4.2.1 Service objectives

4.2.2 Technical objectives

4.2.3 Organizational objectives

4.2.4 Financial objectives

4.3 Experimental

Explanation of experimental research related to loading times / browser compatibility goes here

4.4 Literature

Explanation for empirical evidence based research goes here

4.5 Interviews

Here I'll try to find ways to interview larger companies on the matter and see if they would be interested in a pilot project.

4.6 Research Validation & Critical Notes

5 Research Findings

Here I'll state the findings on the research per domain objectives, with a research conclusion in the end.

5.1 Service objectives

5.2 Technical objectives

5.3 Organizational objectives

5.4 Financial objectives

5.5 Research Conclusion

This will contain a go / no go moment, in which shall be decided if the building section shall be finished within this research.

6 Building – Optional

6.1 Introduction to Single Page Web-Applications

6.1.1 Single Page Concept

6.1.2 Design Patterns

6.2 MEAN Stack

Introduction to the MEAN stack and per component an explanation to the use of it

6.2.1 MongoDB

6.2.2 ExpressIO

6.2.3 Angular

6.2.4 Node JS

6.3 WebGL Engine

6.3.1 ThreeJS

6.4 Requirements

Requirements for the actual app, which components are vital to the usefulness, in other words, what's the MVP

6.4.1 Functional

6.4.2 Technical

6.4.3 Data

6.4.4 MoSCoW

6.4.5 MVP

6.5 DataModel

6.5.1 Node based vs Table based

6.5.2 Data Requirements to Nodes

6.6 Wire framing

6.6.1 User Stories

6.6.2 Prototype

7 Conclusion

8 Further Recommendations

9 Attachments

10 References

[1] Andreas Anyuru, *Professional WebGL Programming: Developing 3D Graphics for the Web*, John Wiley and Sons Ltd, West Sussex, 2012.

[2] Alexis Deveria, *Can I use" provides up-to-date browser support tables for support of front-end web technologies on desktop and mobile web browsers.*, `caniuse.com`

[3] *This is the official OpenGL Website, where they've stated a lot of information on the history of OpenGL*, `https://www.opengl.org/about/`

[4] Sparsh Mittal and Jeffrey S. Vetter *A Survey of CPU-GPU Heterogeneous Computing Techniques*, Oak Ridge National Laboratory and Georgia Tech, 2015.