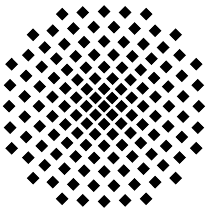

University of Stuttgart
Institute for Theory of Electrical Engineering
Prof. Dr. techn. Wolfgang M. Rucker

Master thesis

Web based Visualization

Nan Zhao

Betreuer:	Dr.-Ing. Matthias Jüttner
Beginn der Arbeit:	01.02.2017
Abgabe der Ausarbeitung:	24.07.2017

Erklärung

Hiermit erkläre ich,

- dass ich die vorliegende Arbeit selbstständig verfasst habe,
- dass ich keine anderen als die angegebenen Quellen und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe,
- dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens ist,
- dass ich die Arbeit noch nicht veröffentlicht habe,
- dass das elektronische Exemplar mit diesem Exemplar übereinstimmt.

Stuttgart, den 24.07.2017

Abstract

Inhaltsverzeichnis

1 Introduction

1.1 Motivation

Data visualization is an important method to demonstrate the results of numerical simulations. With graphics measures. the results can be presented through intuitive form and convey the information more effective, especially in the ares of teaching, product or research presentations, an easily available and lightweight visualization solution is expected[**?**]. Nowadays, with the development of web technologies and modern browsers, a cross-platform, web-based technologies WebGL(Web Graphics Library) make it possible.
Generally, people render 3D graphics by using particular APIs, e.g. DirectX, OpenGL. These APIs define a set of functions which can be called by the client program[**?**]. The solutions which build on these APIs need specific platforms, e.g. DirectX runs on Microsoft products, or specific configuration environment and plug-ins to deploy. These applications are also too heavy to run on mobile devices and their availability and compatibility are limited. The new web standard HTML5 brings up the new spark for web-based application. In general, HTML5 mainly involves the Hypertext Markup Language(HTML), the Cascading Style Sheets(CSS) and Javascript technologies, it introduces new features and new elements to enhance the support for multimedia and graphic, on the one hand the new specification reduces the dependencies on plug-ins, such as Flash, Silverlight, JavaFX, on the other hand it provides native support for graphics rendering with canvas element and JavaScript.
WebGL specification is official introduced into the HTML5 specification in 2014 as a web standard. It allows GPU acceleration and is supported by common browsers, such as Microsoft IE, Google Chrome, Apple Safari, Mozilla Firefox.
WebGL makes it possible for building an easy accessible, lightweight, cross-platform mobile visualization system. The initial idea was came up in the paper "Web Based 3D Visualization for COMSOL Multiphysics®"[**?**].
In this paper some further improvements are worked out and new features are added.

1.2 Outline of this thesis

This section outlines the chapters of this thesis.

Chapter 2: Visualization System

Chapter 3: Web client

Chapter 4: Rendering using WebGL

Chapter 5: Virtual Reality

2 Visualization System

In this chapter the system architecture and corresponding components will be introduced. This chapter emphasizes on depicting the whole picture of the solution, some specific technical detail will be represented in the later chapters.

2.1 System architecture

The visualization system is to display the COMSOL's plots in the web application. To accomplish the goal, the system architecture in fig. 1 is needed.

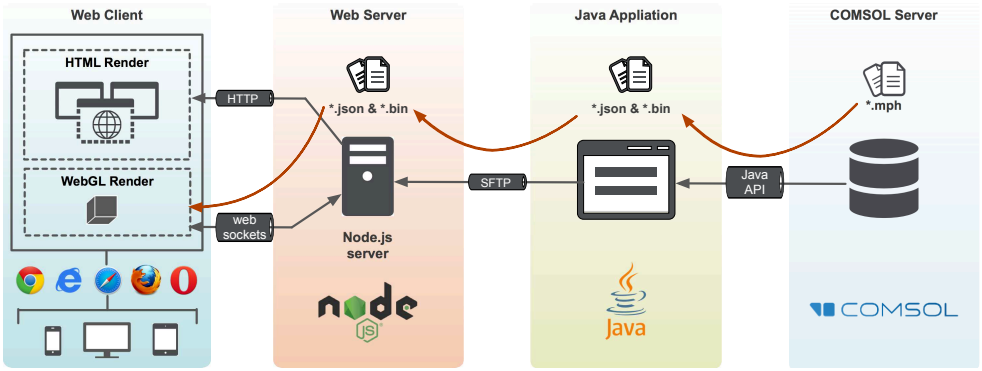

Abbildung 2.1: System architecture

To render the graphics in the web client, the source data need to be extracted and converted at first. COMSOL provides the COMSOL Java API(CJAPI) to handle with its mph file. To use the CJAPI, a Java application is needed. The Java application can access into the model object and extract the meta information and binary data of plots, then export the required data separately into JavaScript Object Notation(JSON) file and binary file. Via a Secure File Transfer Protocol(SFTP), the files are committed to the web server. As the back-end, web server provides the web application and all the necessary data for visualization. The web server is built by Node.js, which is based on Google's JavaScript

run-time environment. Its event-driven architecture and asynchronous I/O capability bring about good performance while loading large volume data. To meet the needs of dynamic reload of plots' data, a bidirectional connection protocol WebSocket is used between web server and web client. Since that web client runs on various browsers and devices, the compatibility must be considered. The web client is responsive design for both mobile and desktop devices and a universal UI(User Interface) is built with HTML5 standard and display the visualization results. Besides, user interaction is also supported for post-processing with the rendered graphics. For more user experience, the stereo display can be enable with 3D screen, and Virtual Reality(VR) experience could also be provided with VR headsets.

2.2 Data processing

The core task of a visualization system is about data processing, the graphics data needs to be extracted from original file and converted correctly for the usage of web client. All algorithms and data structures of a model object is accessible via CJAPI.

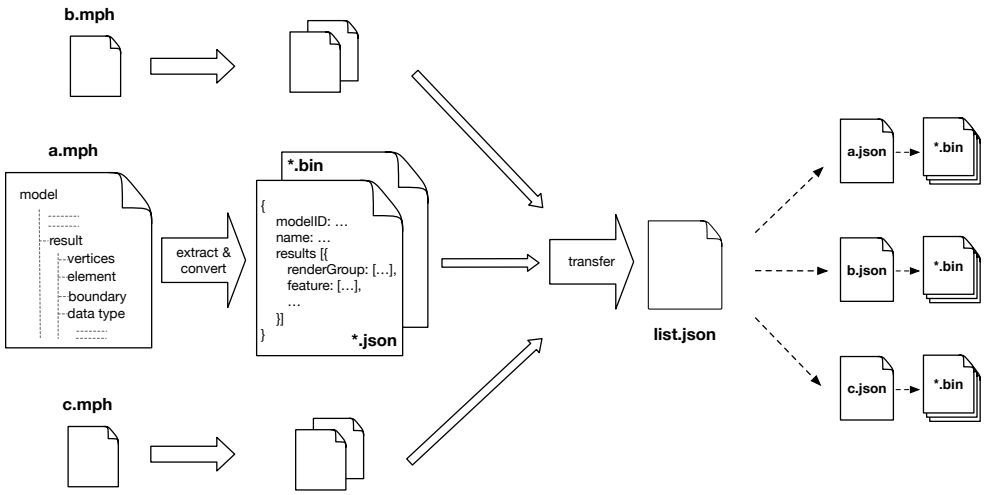

Abbildung 2.2: Data flow

3 Data Extraction and Conversion

3.1 Data-structure of COMSOL file

3.2 Data-format after conversion

4 Rendering using WebGL

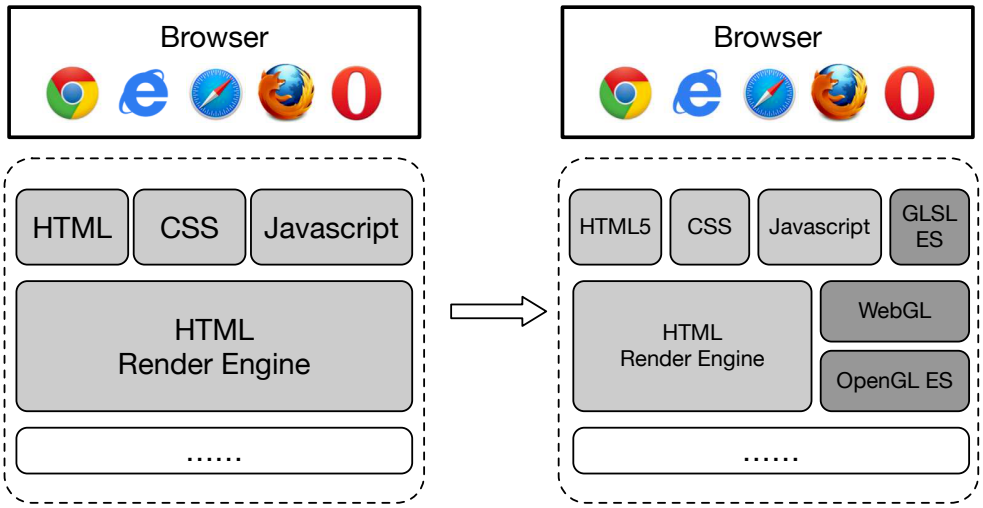

Abbildung 4.1: WebGL

4.1 WebGL graphics pipeline

4.2 2D translation, rotation, scale and matrix math

4.3 3D Orthographic, Perspective, Cameras

4.4 Rendering text using sprites

5 Virtual Reality

6 Conclusion

Bibliography

[1] Bauke, Heiko ; Mertens, Stephan: *Cluster Computing*. Springer, 2006. – ISBN 978–3–540–42299–0

[2] Bellifemine, Fabio ; Caire, Giovanni ; Greenwood, Dominic: *developing multi-agent systems with JADE*. John Wiley & Sons, 2007 (Wiley Series in Agent Technology). – ISBN 978–0–470–05747–6

[3] Göhner, Prof. Dr.-Ing. Dr. h. c. Peter: *Grundlagen der Softwaretechnik*. 2012

[4] Hoffmann, Dirk W.: *Software-Qualität*. 2. Springer Vieweg, 2013. – ISBN 978–3–642–35699–5

[5] Krüger, Guido ; Stark, Thomas: *Handbuch der Java-Programmierung*. 5. Addison-Wesley, 2009. – ISBN 978–3–8273–2815–1

Index of web address

[6] Projektseite DockingFrames. http://dock.javaforge.com/. Eingesehen am 12.11.2014

[7] Projektseite JFreeChart. http://www.jfree.org/jfreechart/. Eingesehen am 12.11.2014

[8] Projektseite Sigar. https://support.hyperic.com/display/SIGAR/Home. Eingesehen am 12.11.2014

[9] Icon der konsole. http://upload.wikimedia.org/wikipedia/commons/8/82/Konsole-icon.png. Eingesehen am 12.11.2014

[10] Icon des status-monitors. http://findicons.com/files/icons/2166/oxygen/128/utilities_system_monitor.png. Eingesehen am 12.11.2014

[11] Projektseite Comsol. http://www.comsol.com/comsol-multiphysics. Eingesehen am 12.11.2014