UNIVERSITY OF TARTU

Institute of Computer Science

Software Engineering Curriculum

Suido Valli

Visualization of simulations of a robot operated car park system

Master’s Thesis (30 ECTS)

Supervisor(s): Dirk Oliver Theis

Tartu 2016

Visualization of simulations of a robot operated car park system

Abstract:

Due to the increasing number of vehicles on the road, traffic problems are bound to exist.This happens as the current car park facilities developed are unable to cope with the influxof vehicles on the road. To overcome this problem, robot operated car park system is developed.This solution would use already built parking lots and parking houses, but will allowsaid parking facilities to accommodate more vehicles as the size of the parking spot will bereduced and most of the roads leading to the parking spots can be eliminated. Other solutionsthat exist will need specially built parking facilities, which makes them more costly to implement.The aim of this thesis is to come up with the best solution for visualizing the algorithmused in the robot operated car park systems. As the algorithm behind it is complex, itis difficult to grasp it without proper visual presentation. The solutions found in the thesiswill be used in making a web application that will let the users interact with the system toget high-level overview of the algorithm’s inner workings

Keywords:

Interfacing, web application, simulation, visualization

**CERCS:**

P170, Computer science, numerical analysis, systems, control

Lühikokkuvõte:

Selle lõputöö tulem on robotiseeritud autoparklate süsteemi simulatsioone visualiseeriv veebiaplikatsioon. Meie ümber on üha rohkem autosid ning metropolides rohkem inimesi. Autode üleküllus viib parkimiskohtade puuduseni. Selle üheks lahenduseks sobiks seniste parklate ümbermuutmine robotiseeritud parkimissüsteemideks – see hoiaks kokku nii maaala kui ka raha, mis täiesti uute parkimiskonstruktsioonide ehitamise jaoks vaja läheks. Selle töö tulemuseks on veebirakendus mis visualiseerib sellise parkimissüsteemi taga oleva algoritmi tööd arusaadavalt ja meeldivalt

Võtmesõnad:

Integratsioon, veebirakendus, simulatsioon, visualiseerimine

**CERCS:**

P170, Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Table of Contents

[1 Introduction 4](#_Toc448427525)

[2 Related work 5](#_Toc448427526)

[2.1 Choice of application environment 5](#_Toc448427527)

[2.2 State-of-art web technologies 5](#_Toc448427528)

[JavaScript Libraries 6](#_Toc448427529)

[2.3 Data object transmitting standards 7](#_Toc448427530)

[2.4 Server side programming languages and frameworks 8](#_Toc448427531)

[2.5 Interfacing with C++ libraries 9](#_Toc448427532)

[3 Background 10](#_Toc448427533)

[4 Overview of the application architecture 11](#_Toc448427534)

[5 Implementing the visualization 12](#_Toc448427535)

[6 Conclusions 13](#_Toc448427536)

[7 References 14](#_Toc448427537)

[Appendix 15](#_Toc448427538)

[I. Glossary 15](#_Toc448427539)

[II. License 16](#_Toc448427540)

# Introduction

In this thesis a new web application is introduced to visualize the simulation of robot operated car park system of which algorithm is developed by Dirk Oliver Theis at University of Tartu. The aim is to come up with the best solution for visualizing the algorithm that makes the algorithm easy to grasp. Furthermore, the visualization has to fill the requirements set by the algorithm. The solutions found in the thesis are used in making the web application that will show the work of the algorithm.

The main obstacles to overcome this thesis is to find the right tools for making a visually appealing, interactive simulation of visualization of robot car parking systems. Furthermore, the right tools will have to be chosen to interface with the algorithm library and with the user. The tools are required to work efficiently together and support visualization process. In addition, the thesis tries to answer to the question how to visualize these kind of algorithms the best for interactivity and understandability. This thesis covers the research of finding the most efficient and understandable way to visualize the simulation of robotized car parking system.

In addition, it covers the extensive practical work during the development of the web application. This includes the client part of the application, part of server that interfaces with the client and finally, part of server that interfaces with algorithm library. As the library is stateless, the application also has to hold the state of the parking lot in its memory.

The findings of the thesis are demonstrated in developing the aforementioned visualization application.

# Overview

## Similar solutions

# Application requirements

## Non-functional requirements

## Functional requirements

# System architecture

Technologies used are state-of-art and work for the best result. Web application approach was chosen as in recent years, the browser support for HTML5 technology stack standards has improved drastically, which makes web application developed for HTML5 truly crossbrowser and cross-device experience. The other advantage is the ability to access the application from anywhere.

Phaser was chosen as a JavaScript library supporting the visualization part of the development as it has the right support for the requirements of the application. This includes scalability in screen sizes and JSON support. JSON is used as data object transmitting standard as it is quicker to parse and transmit, but also because of its integration in JavaScript language.

Java is a solid language that will be used in the application over other languages as it provides security, strong typing, good documentation and big community. It is also the language that the author is most familiar with and knows the MVC structure of.

JNA was used over JNI for C++ method interfacing as it does not need any development on C++ side, which the author cannot do without asking permission of the supervisor.

## Architecture overview

## Server side platform selection

Nowadays server side programming can be made by many different languages that have different approaches on the web server design patterns. These language include Java, ASP.NET, Python, PHP, Ruby on Rails et cetera. For this project three languages were taken under review that could fill the requirements set by the client side application and the comfortableness of the author in the languages. These languages are PHP, Java and ASP.NET, which builds on C#.

Java is a general-purpose programming languages. The approach in Java is to let developers “write once, run anywhere” [7]. Java is used in a huge variety of use cases, including desktop applications, mobile applications and web applications, client and server side.

PHP is a server-side scripting language which is designed for web development. It can also be used as a general-purpose programming language. PHP is still one of the popular choices of web developers, known for huge community, big variety of frameworks and good documentation. ASP.NET is Microsoft’s open-source web application framework designed to produce dynamic Web pages.

ASP.NET uses C# programming language. A comparative study [8] compares the three aforementioned web technologies using MVC (Model-View-Controller) design pattern on all of the cases to build a web application. In Java that means using JSP web pages (\*.jsp) as views, the controllers are developed using Java servlets and the models are developed using Enterprise Java Beans and Java Persistence API. In ASP.NET, views were developed using Active Server Pages. The controllers are in C# code and views can be developed by either razor or aspx. As PHP does not have its own MVC design pattern, PHP system was implemented using CakePHP framework.

The study concluded that Java processes Login HTTP requests faster than two other systems. The study also shows that Java is the quickest to process GET method requests, but ASP.NET is the quickest in processing POST method requests. All in all, Java and PHP were found to be cheaper to implement compared to ASP.NET. The downside of Java is the difficulty of using it compared to CakePHP and ASP.NET.

Taking into account the findings in the study, Java MVC framework is used in the development of server side application for robot car parking system visualization. As most of the requests will be GET requests from client side to server, Java is the fastest to process them, which adds to the speed of the whole application

## Data object transmitting standards

The traffic between client and server in the discussed web application in this thesis will, in majority of times, be by asynchronous calls. AJAX [5]standard will be used to accomplish that. There raises a dilemma, which kind of data object transmitting standard to use in this process. The ones under review are the two most popular, XML and JSON.

XML (The Extensible Markup Language) is considered the ‘holy grail’ of computing for its universal data representation format. The priorities when designing the languages were simplicity and human readability. It is primarily used for Remote Procedure Calls. XML does not have any pre-defined tag sets – everything can be configured by user. An example of an object in XML is provided in figure 1.

<person>

<firstname>Suido</firstname>

<lastname>Valli</lastname>

</person>

**Figure 1.** Example of XML object.

JSON is designed to be human readable and easily parsed and used by computers. JSON is directly supported inside JavaScript. JSON is estimated to parse up to hundred times faster than XML. The disadvantages over XML include lack of namespace support, lack of input validation and extensibility drawbacks. An example of an object in JSON is provided in figure 2.

{

“firstname”: “Suido”,

“lastname”: “Valli”

}

**Figure 2.** Example of JSON object.

From the study [6] it can be concluded that sending data in JSON encoding is in general faster than sending it in XML encoding. On the other hand, the transmission times of XML are lower when fever objects are transmitted. As for the fact that sending and parsing JSON data is generally faster than doing the same with XML data and that JSON is natively supported in JavaScript, JSON will be used in the thesis project for transmitting data objects between client application and server.

## Client side platform selection

There are quite a lot web technologies in use to make a graphic animation. Three of the biggest are Flash, Java and combination of HTML5, CSS3 and JavaScript.

Adobe Flash, formerly known as Macromedia Flash and Shockwave Flash, is a platform for creating rich Internet applications, usually used for development of Web-based games and interactive tools. Flash Player plugin is required for Flash content to work as it runs as a client-side sandboxed virtual machine [1]. Downside of Flash is the fact that it is proprietary technology.

Java is a powerful development platform. In Web browsers, Java runs in sandboxed virtual machine. For that fact, Java applets are theoretically platform-independent. The problem with Java is its need for the plugin and the fact that different applications may need different versions of the plugin. Furthermore, the initialization of the applet can be long. Also, there are serious security issues, due to what Java applets are not executed automatically in Web browsers anymore [2].

HTML5, CSS3 and JavaScript (from here on abbreviated to HTML5) provide open source alternative to Java and Flash. HTML5 is used provide static content, CSS3 to style the said content and JavaScript is used to make the content dynamic. As HTML5 is supported by all major browsers without plugin, the users do not have to be worried about security risks. One potential disadvantage of HTML5 would be the different interpretation of the standards by different browsers, but these could be easily overcome with some JavaScript libraries, for example jQuery. The main disadvantage of HTML5 is the relative ease to see the source code. As this thesis is in public domain, this fact is not accounted when choosing the web technologies. As mobile devices have much more interaction channels than traditional network channel, there are new ways for untrusted data to enter mobile devices. When this data is executed, there could be serious consequences. Solutions like PhoneGap are developed to fix the issue. [3]

As the main reason the web application approach for visualization was chosen was the application to run seamlessly on any number of devices, Java and Flash are ruled out mostly because of their plugin requirement, but also for the fact that they are morally old technologies and, as use of them decreases steadily, they might not be supported soon. The technology used for making the visualization application is HTML5 technology stack: HTML5, JavaScript and CSS3

### JavaScript Libraries

JavaScript as a language is very extensive and right now, the language that web programmers use. For visualization, one could start from scratch and invent the logic behind the animation him/herself. This approach is prone to errors and very time exhausting. As there are plenty of different graphics frameworks written on top of JavaScript, it would be wise to study the options and choose the most suitable one.

Ten different options will be studied and best suited for the purpose is chosen. [4]

CAAT (Canvas Advanced Animation Toolkit) is a 2D director based scene graph renderer. CAAT has a chance to have multiple instances as you can have unlimited number of directors for each web page. It is very diverse toolkit with unlimited number of timelines and actors. CAAT has abstracted input system and it does resource management and preloading. Disadvantages of CAAT is that it was last updated over two years ago and it does not have a community behind it.

CakeJS is a scene graph library written in JavaScript meant for HTML5 canvas tags. It has animation timelines and it supports mouse events. It does not support easy resizing of application in different screen sizes and aspect ratios. Furthermore, this project has been archived.

Canvas Engine is a library for easily creating games in HTML5 Canvas. It works on modern browsers and also smartphones. The scenes can be structured and preloaded. In addition, Canvas Engine has a model for handling server-side events to develop a multiplayer game. Although Canvas Engine is meant for building games, it is also suitable for the purpose of robot car parking system visualization and simulation. The main disadvantage is the smallness of community and the fact that Canvas Engine has not been updated for over a year.

ChesterGL (Chester Game Library) focuses on ease of use and performance. It supports a simple scene graph. It has time based actions, different shaders (for WebGL) and batched sprites. The disadvantage of ChesterGL is smallness of community and infrequent updates, lastly updated 2 years ago.

Cocos2d-JS is a game engine for web and native games. It has high performance, has modern JavaScript API, full platform without needing plugins. Furthermore, there is a capability to test and debug the developed application on the browsers before pushing them to target device. It has vast API including transitions, events, persistence et cetera. Cocos2d-JS is a good candidate for the JavaScript library to be used in the thesis practical work.

Construct 2 is designed for creating 2D applications rapidly. It is very high level framework. Construct 2 is not open source nor free as majority of other libraries discussed in this thesis. The major difference of Construct 2 is that it does not require any coding – the primary method of programming is through the event sheets. As Construct 2 is not free and there can be issues interfacing with server that cannot be issued inside the event sheets environment, it is not suitable for the purpose of the application developed in thesis.

Crafty is a flexible 2D framework for JavaScript games. It can either use Canvas or DOM, it has sprite map support and collision detection. It is also lightweight framework and open source. Furthermore, it is actively developed and has active community. All in all, Crafty is a considerable candidate, taking into account the requirements of the application developed.

EaselJS is a 2D graphics engine which supports objects nesting and has mouse interaction model. It uses familiar approach for developers which should make it easy to work with. EaselJS is frequently used in making HTML advertisements. ImpactJS is a 2D and isometric HTML5 graphics engine. Tit supports real time multi user application and implements a scene graph based architecture. It has a built in physics module.

ImpactJS is a considerable framework to use for the application, as it allows creating in addition to two-dimensional scenes isometric scenes, which can be beneficial for understandability of algorithm workings.

Phaser is a free 2D game framework that supports Canvas and WebGL rendering. It uses Pixi.js internally for rendering. Phaser can automatically switch between Canvas and WebGL rendering, according to the available technologies in the device. Phaser can use JSON and XML for asynchronous calls, it supports inputs from mouse, touch screen, keyboard. Furthermore, with Phaser has built-in Scale Manager which allows developer to scale the application to fit any size screen. Phaser is actively developed and has active community.

Although there are plenty of good candidates in the 10 reviewed libraries, the author chose Phaser as it has the most useful features needed for the development of parking system visualization built in. For example, it can use JSON or XML data without any further libraries or development on programmer part

# Application overview

## Application flow

## Application functions

# Validation

# Future opportunities

# Conclusions

# References

|  |  |
| --- | --- |
| [1] | N. Stewart ja S. Reimers, „Presentation and response timing accuracy in Adobe Flash and HTML5/JavaScript Web experiments,“ *Behavior Research Methods,* pp. 309-327, 2015. |
| [2] | N. Ganesh, *Static Analysis of Malicious Java Applets,* San Jose: San Jose State University, 2015. |
| [3] | P. Trivedy ja S. Harneja, „HTML5 – the new standard for Interactive Web,“ *International Journal of Research,* kd. 1, nr 10, pp. 39-42, 2014. |
| [4] | AIR FORCE RESEARCH LAB ROME NY INFORMATION DIRECTORATE, *Advanced Visualization and Interactive Display Rapid Innovation and Discovery Evaluation Research Program task 8: Survey of WEBGL Graphics Engines,* New York: AIR FORCE RESEARCH LAB ROME NY INFORMATION DIRECTORATE, 2014. |
| [5] | J. J. Garrett, *Ajax: A New Approach to Web Applications,* San Fransisco: Adaptive Path, 2005. |
| [6] | N. Nurseitov, M. Paulson, R. Reynolds ja C. Izurieta, *Comparison of JSON and XML Data Interchange Formats: A Case Study,* Montana: Montana State University, 2009. |
| [7] | N. Langley, „Write once, run anywhere?,“ May 2002. [Võrgumaterjal]. Available: http://www.computerweekly.com/feature/Write-once-run-anywhere. [Kasutatud 14 04 2016]. |
| [8] | D. A. Botwe ja J. G. Davis, „A Comparative Study of Web Development Technologies Using Open Source and Proprietary Software,“ *International Journal of Computer Science and Mobile Computing,* kd. IV, nr 2, pp. 154-165, 2015. |

Appendix

1. Glossary

|  |  |
| --- | --- |
| Caret  The bar (or other symbol) marking the active editing point. | Sisestusmärk  Märk, mis märgib teksti sisestamise asukohta. |
| Template  A gauge, pattern, or mold, commonly a thin plate or board, used as a guide to the form of the work to be executed. | Mall  Näidik, muster või valuvorm, mis esitab täitmisele võetava töö struktuuri. |

1. License

**Non-exclusive licence to reproduce thesis and make thesis public**

I, **Suido Valli**,

(*author’s name*)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

**Visualization of simulations of a robot operated car park system**,

*(title of thesis)*

supervised by Dirk Oliver Theis,

*(supervisor’s name)*

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **19.04.2016**