WWW-Graphics-Module for Mixed-Reality-System

Hannes Eilers, *Master-student, Chairman student group NorthernStars, FH-Kiel,*
Eike Petersen, *Master-student, FH-Kiel*

***Abstract*—Warum? Was? Wie? (Max 200 W)**

***Index Terms*—mixed-reality, javascript, server, client, paper.**

I. Introduction

THE Mixed-Reality is a robocup soccer league that uses hardware and software simulation to develop artificial intelligences (AIs) for robots. The system used for playing in competitions all over the world uses a horizontal screen, that simulates a soccer field with field lines, goals and a ball. On the screen are real robots with quadratic dimensions for each side of nearly 25cm. A game server uses a vision system above the screen for tracking the robots and generating a worldmodel that includes positions of all robots as well as simulated objects like goals or important flags on the field. By using a infrared transmitter the server can send commands to the robots.
For each robot a seperate process can connect to the game server to control the speed of the right and left wheel or the robot. The gameserver sends the worldmodel to this process, so that it can react to changes on the field.
The Mixed-Reality system uses several modules to track and control the robots or to display the field on the screen.
During a competition one AI for each robot connects to the game server using a local network and team leaders next tof the soccer field can request timeouts to change the behaviour of a AI or replace the robots for kick offs.

A. *Project definition (Was machen wir?)*

The projects goal is to bring the local mixed-reality-games to an audience everywhere in the world (Figure 1). All that is needed is a device with a HTML-5 compliant Browser and an internet-connection with a bit-rate of at least 10-kbit/s downstream. The primarily targeted devices are pcs, tablets and smart-phones but not limited to those.
The games should be streamed in real-time from the mixed-reality game-server over a web-server to in-browser RICH-clients, where they will be displayed in the same fashion as the local mixed-reality-game graphics.

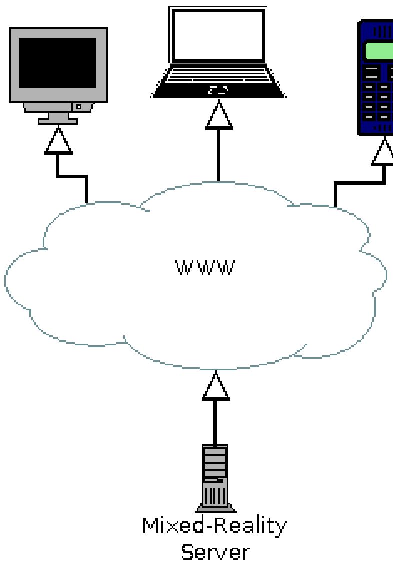

Fig. 1. **Project goal:** In the 1st stage we just wanted to bring the mixed-reality-games easily and reliably to end-users on all kinds of devices.

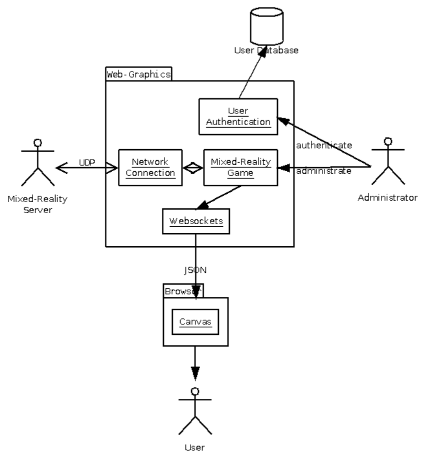

Fig. 2. **Project design:** As seen the concept has two parts: the server for connecting to the mixed-reality system and reciving and sending the gamedata to the clients and the client who interprets the data and displays it in a human readable and enjoyable format.

B. *Design (Wie sieht es aus?)*

As in Figure 2 shown, the web-server is capable to establish a connection to any Mixed-Reality game-server and game over UDP. This connection is interchangeable an internal game representation, that converts the incoming XML to JSON and verifies it, manages the listening clients and encapsulates needed administrative functions. The verified JSON is sent over websockets at a fixed interval to the clients, (Canvas und so? RICH-Client usw)

C. *Implementation(Wie ist es aufgebaut?)*

1) Server: The design for the server is implemented as a proof-of-concept on the server-framework Node.js [1] with most of the needed configuration and essentials implemented with the middle-ware express [2], such as session-management, URL-routing and serving of static files. The Website is constructed through templates with the template-engine jade [3] and also served through express.

The user-persistence is handled by an sqlite-database with sqlite3 [4], because of its convenience to setup and redesign in a agile software development environment [5] and the ease of conversion to a better setup when the project reaches maturity.
The communication besides http is managed by websocket.io [6] [7], an extension to express.
To decode the incoming XML xml2js [8] is used and to build an easy to manage and debug system extensive logging is done with log4js [9] an conversion of the Java logging package log4j2 [10].
All the modules written for the server use the functional inheritance/creation pattern [11] to create private and secure objects. Every time an object is needed only once, it is created as a singleton with closure [12] to avoid misuse and implementation errors. The communication between the mixed-reality-game representation and the connected websockets is implemented with a simplified observer pattern [12].

2) Client:

II. Conclusion

The project and the creation of the proof-of-concept have been a success, even while we where not able to bring mixed-reality to mobile users with smart-phones, because of the spotty HTML5 compliance, we achieved all others of our 1st stage goals.
Even in lieu of this more work is needed to develop the proof-of-concept further, to be easily and reliably used in the mixed-reality-system. Sorely needed is a rigorous easy and automatic testing of all components and their implementation, witch has been neglected because of time- and manpower-constraints. Also should we implement all of our 2nd stage goals, like replays of games, game-chat between spectators and a secure and robust account management with tie-ins to social media. Also when HTML5 will be fully implemented on all mobile devices, we can develop the mobile page.

Acknowledgment

The authors would like to thank the Fachhochschule Kiel for use of their location and rooms, the Hochschulgruppe NorthernStars, www.northern-stars.de, for supporting and helping us on the edges of the project and Prof. Manzke for letting us skirt his regulations, so that we could build something that we can use and develop further.
Our source-code lies at https://github.com/NorthernStars/MR-Web-Graphics and everyone who want to use or every develop it further is heartily encouraged to do so.

References

[1] Joyent, Inc, *Node.js*, http://nodejs.org/.
[2] expressjs.com, *express web application framework for node*, http://expressjs.com/.
[3] jade-lang.com, *jade Node Template Engine*, http://jade-lang.com/.
[4] MapBox, *node-sqlite3 - Asynchronous, non-blocking SQLite3 bindings for Node.js*, https://github.com/mapbox/node-sqlite3.
[5] K. Beck, *Extreme programming eXplained : embrace change*, 2nd ed. Addison-Wesley, 2005.
[6] G. Rauch, *websocket.io*, https://npmjs.org/package/websocket.io.
[7] G. Rauch, *SMASHING Node.js JavaScript Everywhere*, 1st ed. WILEY, 2012.
[8] Leonidas-from-XIV, *node-xml2js*, https://github.com/Leonidas-from-XIV/node-xml2js.
[9] nomiddlename, *log4js-node*, https://github.com/nomiddlename/log4js-node.
[10] Apache Software Foundation, *Apache Log4j 2*, http://logging.apache.org/log4j/2.x/index.html.
[11] D. Crockford, *JavaScript: The Good Parts*, 1st ed. O'Reilly, 2008.
[12] Data Object Factory, *JavaScript + jQuery Design Pattern Framework 2013*, Data Object Factory, LLC, 2013.