WWW-Graphics-Module for Mixed-Reality-System

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

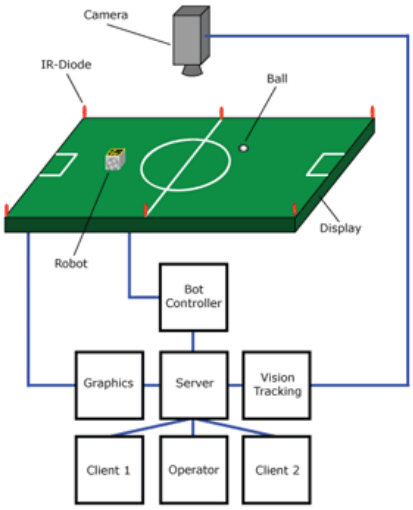

Fig. 1. **Mixed-Reality system:** The simulated soccer field is controled by a game server that uses several other modules for access to the robots, displaying the field or interacting with human operator or artifical intelligences

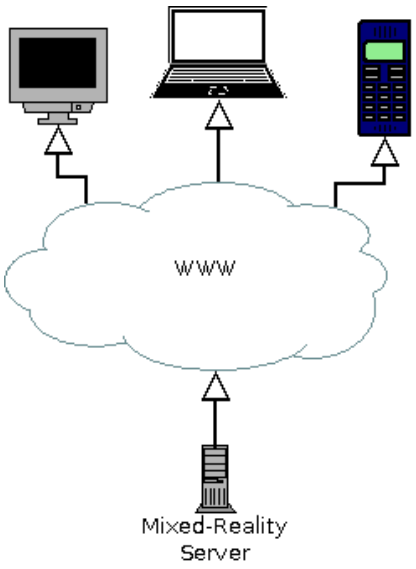

Fig. 2. **Project goal:** In the 1st stage we just wanted to bring the mixed-reality-games easily and reliably to end-users on all kinds of devices.

***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 for robots. The system used for playing in competitions all over the world [1] uses a horizontal screen on which a soccer field with field lines, goals and a ball is simulated. On this screen real robots with quadratic dimensions for each side of nearly 2,5cm move. A game server uses a vision system above the screen for tracking the robots and generating a world-model that includes positions of all robots as well as simulated objects like goals or important positions on the field. By using an infrared transmitter the server sends commands to the robots(Figure 1).

For each robot an artificial intelligence can connect to the game server to control the movement and actions of the robot. The game-server sends the world-model to this artificial intelligence, so that it can react to changes on the field.

The Mixed-Reality system is module based and uses different modules to track and control the robots or to display the field on the screen.

During a competition teams of artificial intelligences connect to the game server using a local network to control their designated robots and play matches against each other in realtime [2], while observers can watch the game live from the sidelines.

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 2) so that it is possible to observe matches or even play against each other without being present at the Mixed-Reality systems location. All that is needed is a device with a HTML5 compliant Browser and an internet-connection with a bit-rate of at least 10-kbit/s downstream.

B. Design (Wie sieht es aus?)

As in Figure 3 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, that checks the recieved JSON data for drawable objects. The objects are drawn using the 2D context of a HTML5 canvas [3].

The client offers forms for login an connecting to a new game, wich is only nessesary to connect to a new game. Watching a game is open to a non-resitricted audience.

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 [6] with most of the needed configuration and essentials

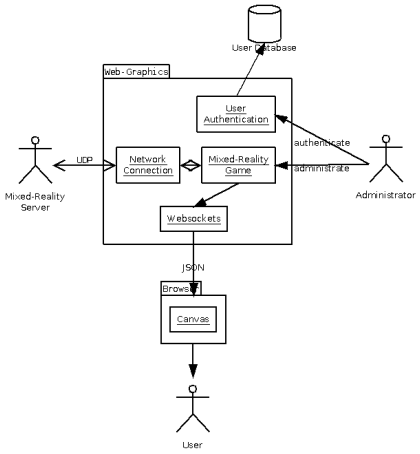

Fig. 3. **Project design:** As seen the concept has two parts: the server for connecting to the mixed-reality system and reciving and sending the gamedata to the clients and the client who interprets the data and displays it in a human readable and enjoyable format.

implemented with the middle-ware express [7], such as session-management, URL-routing and serving of static files. The Website is constructed through templates with the template-engine jade [8] and also served through express.
The user-persistence is handled by an sqlite-database with sqlite3 [9], because of its convenience to setup and redesign in a agile software development environment [10] and the ease of conversion to a better setup when the project reaches maturity.
The communication besides http is managed by websocket.io [11] [12], an extension to express.
To decode the incoming XML xml2js [13] is used and to build an easy to manage and debug system extensive logging is done with log4js [14] an conversion of the Java logging package log4j2 [15].
All the modules written for the server use the functional inheritance/creation pattern [16] to create private and secure objects. Every time an object is needed only once, it is created as a singleton with closure [17] 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 [17].

2) Client: The client side web templates using CSS3 [4] to specifiy the user interfaces design, generated from the HTML tags using the jade template engine. It includes a hidden overlaying login form, thats visibilty is accessable through an portable javascript function and used for login into webservers administration page.
The CSS3 design also resets the browser based drawing styles for common HTML5 tags to define a common design of the clients webpage on different browser versions. So a unique design for forms or list-like items can be guaranteed.
For drawing graphical objects from the recieved JSON data a 2D context of a HTML5 canvas obejct is used. Everytime new data recieved from the server the canvas is cleared and the new data is drawn on it. A set of javascript functions allows context independend drawing of complex objects like rectangles, circles, dots or robot markers. So a RICH client [**?**] functkionality for different mixed-reality szenarios and applications is offered. However currently only drawing of soccer szenarios is implemented.
For drawing simple objects like rectangles or circles the build in canvas context instructions [3] are used. For drawing the robots marker a more complex drawing is implemented, because the marker consists of a image representing its current position and team color and also a text showing information about the internal identification number of the robots and its name. So the client uses a reusable HTMl image object [5] that is drawn on the canvas. After that the drawn image is no longer accessable and the image objetc can be resused for drawing an image again. Using this the bclient doesn't have to wait before drawing until the image object is loaded through the browser. The text for the robots information is added to the webpage document as a html tag with unique id, inside a hidden supsection of the documents structure. So the text position an orientation can be set using CSS3. This is nessesary because drawing the text directly on the canvas needs to much performance for smooth animations of the robot markers.

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 Unisversity of Applied Sciences Kiel for use of their location and rooms, the robotics group 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 for the NorthernStars.

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] R. Gerndt, M. Bohnen, R. da Silva Guerra, M. Asada, *The Engineering of Mixed Reality Systems Human-Computer Interaction Series, Chapter 20: The RoboCup Mixed Reality League A Case Study*, 1st ed. Springer 2010

[2] NorthernStars, *Mixed-Reality*, http://northernstars-wiki.wikidot.com/robocup:mixed-reality

[3] W3C, *HTML Canvas 2D Context*, http://www.w3.org/TR/2dcontext/

[4] W3C, *Introduction to CSS3*, http://www.w3.org/TR/2001/WD-css3-roadmap-20010523/ rich-client Wikipedia Foundations, Inc., *Rich client platform*, https://en.wikipedia.org/wiki/Rich_client_platform

[5] W3C, *Objects, Images, and Applets*, http://www.w3.org/TR/REC-html40/struct/objects.html#h-13.1

[6] Joyent, Inc, *Node.js*, http://nodejs.org/.

[7] expressjs.com, *express web application framework for node*, http://expressjs.com/.

[8] jade-lang.com, *jade Node Template Engine*, http://jade-lang.com/.

[9] MapBox, *node-sqlite3 - Asynchronous, non-blocking SQLite3 bindings for Node.js*, https://github.com/mapbox/node-sqlite3.

[10] K. Beck, *Extreme programming eXplained : embrace change*, 2nd ed. Addison-Wesley, 2005.

[11] G. Rauch, *websocket.io*, https://npmjs.org/package/websocket.io.

[12] G. Rauch, *SMASHING Node.js JavaScript Everywhere*, 1st ed. WILEY, 2012.

[13] Leonidas-from-XIV, *node-xml2js*, https://github.com/Leonidas-from-XIV/node-xml2js.

[14] nomiddlename, *log4js-node*, https://github.com/nomiddlename/log4js-node.

[15] Apache Software Foundation, *Apache Log4j 2*, http://logging.apache.org/log4j/2.x/index.html.

[16] D. Crockford, *JavaScript: The Good Parts*, 1st ed. O'Reilly, 2008.

[17] Data Object Factory, *JavaScript + jQuery Design Pattern Framework 2013*, Data Object Factory, LLC, 2013.