Todo list

Bei Abgabe: Anweisung nocite in Bachelorarbeit.tex entfernen

Bei Abgabe: In Bachelorarbeit.tex und Expose.tex Dokumentenoption overfullrule entfernen und die Option final eintragen

Bei Abgabe: In Packages.tex beim Package todonotes die Option disable eintragen, um Todos zu deaktivieren

Bei Abgabe: In Packages.tex beim TODO die größte Seitennummer eintragen

Wenn eine Version der Arbeit erstellt wird, die gedruckt werden soll in Packages.tex beim Package hyperref die Option urlcolor=blue entfernen

TECHNIK
HOCHSCHULE MAINZ
UNIVERSITY OF
APPLIED SCIENCES

Masterarbeit

Studiengang Geoinformatik M.Eng.

A standards-based, task-specifc telepresence application for contemporary dance

Hochschule Mainz

University of Applied Sciences

Fachbereich Technik

Vorgelegt von:	Anton Koch
	Felsstraße 1
	67744 Seelen
	Matrikel-Nr. 946750
Vorgelegt bei:	Prof. Pascal Neis
Eingereicht am:	14.02.2024

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit

> **A standards-based, task-specifc telepresence application for contemporary dance**

selbstständig und ohne fremde Hilfe angefertigt habe. Ich habe dabei nur die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt.

Zudem versichere ich, dass ich weder diese noch inhaltlich verwandte Arbeiten als Prüfungsleistung in anderen Fächern eingereicht habe oder einreichen werde.

Mainz, den 14.02.2024 Anton Koch

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit.

Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Contents

List of Figures

List of abbreviations

3D 3-dimensional

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

CSS Cascading Style Sheets

HTML HyperText Markup Language

HTTP Hypertext Transmission Protocol

IETF Internet Engineering Task Force

JSX JavaScript Extensible Markup Language (XML)

JS JavaScript

PWA Progressive Web Application

RFC Request For Comments

RTC Real-time Communication

SPA Single Page Application

TS TypeScript

UI User Interface

W3C World Wide Web Consortium

XML Extensible Markup Language

XR Mixed Reality

1. Introduction

1.1. Background

In recent years, remote collaboration has become increasingly important due to the rise of broader digitalisation strategies and especially since the pandemic. As a result, teleconferencing and telepresence platforms have become more pervasive in a lot of work environments. These platforms allow people to work together remotely in real time, usually focusing on streaming video and audio, document sharing or collaborative whiteboarding. While this is fine for most use cases and desktop-based workplaces, it lacks certain immersive qualities required for practices such as contemporary dance, where people relate to a shared space and physical presence. This became very apparent in March 2020, when dancers were unable to rehearse and work together due to the lockdown. Despite this, there were approaches using Zoom to stream and record collaborative rehearsals or dance classes, but these were confined to a screen-centric interface and limited to audio and video.

While commercial conferencing tools such as Zoom, Google Meet and Microsoft Teams dominate in popularity among conferencing applications (Brandl, 2023), there are both free and open source variants such as JitsiMeet and BigBlueButton. However, these all focus on the most basic form of screen-based conferencing mentioned above. While there are various domain-specific solutions for specialised applications, mainly in telemedicine (as well as general industry and the military), that support more immersive and task-

specific remote collaboration, these are either unaffordable or unavailable to the general public.

As support for web standards is driven by key industry players (Davis et al., 2023), and with it the availability of a wide range of basic functionality, as well as access to display and sensor technology for deploying applications on desktop and mobile devices, there is an increased potential for smaller and more specific applications to be built and deployed with relative ease. This opens up new possibilities for niche cases of remote collaboration, such as dance, where collaborative functionality needs to be extended from the traditional paradigm of remotely viewing video streams to the creation of shared virtual environments that facilitate a sense of 'being there' together, if only at an abstract level (Skarbez et al., 2017).

The standard for Real-time Communication (RTC) in Browsers or WebRTC ('WebRTC: Real-Time Communication in Browsers', 2023) was first proposed by Google in 2011 and became an official World Wide Web Consortium (W3C) standard in 2021 (Couriol, 2021). It is already being used in a wide range of applications such as some of the conferencing tools mentioned above, media streaming servers such as Wowza or Ant, or real-time frameworks and servers such as Mediasoup, Janus or LiveKit. In its most basic form, WebRTC establishes peer-to-peer connections between different devices, allowing low-latency exchange of media streams as well as arbitrary messages on so-called "data channels".

1.2. Proposal

I propose a feasibility study for a reference implementation of a domain-specific telepresence platform based entirely on web standards and open source components. The platform allows streaming of sensor data in addition to audio and video. This means that remote collaborators can share a wide range of data from multiple sources, such as motion capture, wearables and more. As the data flows through the WebRTC data

channels alongside the basic video and audio streams, it can be analysed and rendered as required for a specific task on the client device. To demonstrate the platform's capabilities, a reference implementation will be created to simulate basic aspects of presence in a shared virtual environment (Skarbez et al., 2017). The software implementation, written entirely in JavaScript, should rely as much as possible on existing open source libraries and frameworks and add as little custom code as necessary. Only the minimal viable product will be built to observe its basic functionality, leaving issues of robustness, interface design, security and scalability outside the scope of this study. For dancers to communicate while moving in remote rehearsal spaces, a video feed is less important as they need to be free from looking at a screen or typing on a keyboard. A virtual or augmented reality headset is also ineffective because it hinders vision and movement in the physical space. Here, spatial sonification seems to be the optimal alternative to create a virtual environment without interfering with the dancers' ability to move freely. The sound generation code should be kept as minimal as possible.

For practical implementation testing, two remote rehearsal spaces of similar size will need to be set up with a motion capture system (TDB) that generates spatial data from the dancers' movements. Both rooms will be within the university network to provide a known and controllable quality of service. The two dancers will use the system in each location. Interaction will be via wireless in-ear sports headsets, which are commonly used for exercise and already provide a basic verbal communication method via WebRTC's audio channels.

To interact with the spatial dimension of a shared virtual environment consisting of both rehearsal rooms, sonification is used as a means of orientation in the virtual space. The Web Audio Application Programming Interface (API) provides functionalities for generating, mixing and positioning sound sources in spatial audio ('Web Audio API', 2021), which will be used to render each participant's microphone sound alongside their own distinct signature sound (one percussive and the other sustained) at their tracked position in space, modulated by two basic movement qualities, the contraction index and the quantity of movement, based on an existing proposal for qualitative movement analysis

(Volpe, 2003) and derived from the motion capture data. Each participant can only hear the other to avoid confusion. The spatial audio representation should provide a basic sense of position, while the sound characteristic should indicate the type of movement being performed. In addition, verbal cues can be exchanged.

A separate client implementation uses the same data to visualise both the dancers' motion capture data as basic 3-dimensional (3D) images and adding the combined sonification, which can be viewed by a third party using the Mixed Reality (XR) Device API for the web ('WebXR Device API', 2023), allowing viewers to evaluate aesthetic aspects of the combined interaction result. In addition, a recording of the session's streams can later be replayed and reviewed by the dancers themselves using this client implementation.

2. Concepts

2.1. Telepresence

2.2. Motion Capture

2.3. Movement Sonification

2.4. Application Design Paradigms

2.4.1. Single Page Applications

The concept of a Single Page Application (SPA) originated around the beginning of the 2000s with the terms "Inner-Browsing" (Galli et al., 2003) and Asynchronous JavaScript and XML (AJAX) (Garrett, 2005). It breaks with the traditional way of moving from one page to another in favour of asynchronously loading and replacing parts of the current page. This allows for a website to evoke the look and feel of a regular desktop application.

2.4.2. Progressive Web Applications

The term Progressive Web Application (PWA) was initially coined in 2015 by two Google employees in an online Article (Russell & Berriman, 2015). At its core it describes the process of a website "progressively" evolving into a true device application by adding offline functionality and blending with the operating system functionality. It is often built atop the concept of an SPA and can be perceived by the user as an application they own instead of just accessed at a remote location.

2.4.3. Real-time Web Applications

A real-time web application enhances the user experience by relaying relevant changes on the server to the server as they happen. This can be a simple chat application or a more complex collaborative multi-user environment. While real-time updates can happen on any multi-page website, they can also be a beneficial feature of an SPA or a PWA. Instantaneous updates are commonly realised using WebSockets, a transmission protocol that was standardised as Request For Comments (RFC) 6455 by the Internet Engineering Task Force (IETF) in 2011 (Fette & Melnikov, 2011). It allows full-duplex communication between client and server, running on the same ports and in the same transport layer as the half-duplex Hypertext Transmission Protocol (HTTP) protocol, thus being compatible with existing web infrastructure. It allows for updates to be pushed to the client whenever a resource on the server changes.

2.5. Application Deployment

2.5.1. Containerisation

2.5.2. Orchestration

3. Tools

3.1. Web Development Languages

3.1.1. JavaScript

3.1.2. TypeScript

3.2. Native Application Development

3.2.1. Node JS

3.2.2. Python

3.2.3. C/C++

3.3. Frontend Frameworks and Libraries

There is a wide range of available JavaScript (JS) frameworks to build dynamic frontends for SPAs and PWAs. The three libraries currently dominating the landscape are *React*, developed by *Facebook* in 2013, and *Vue.js*, developed by Evan You in 2014. These libraries can be used in conjunction with frameworks to offer complete solutions providing

routing and state management. Another popular framework is *Angular*, which was originally released by *Google* in 2010, then re-released in 2016.

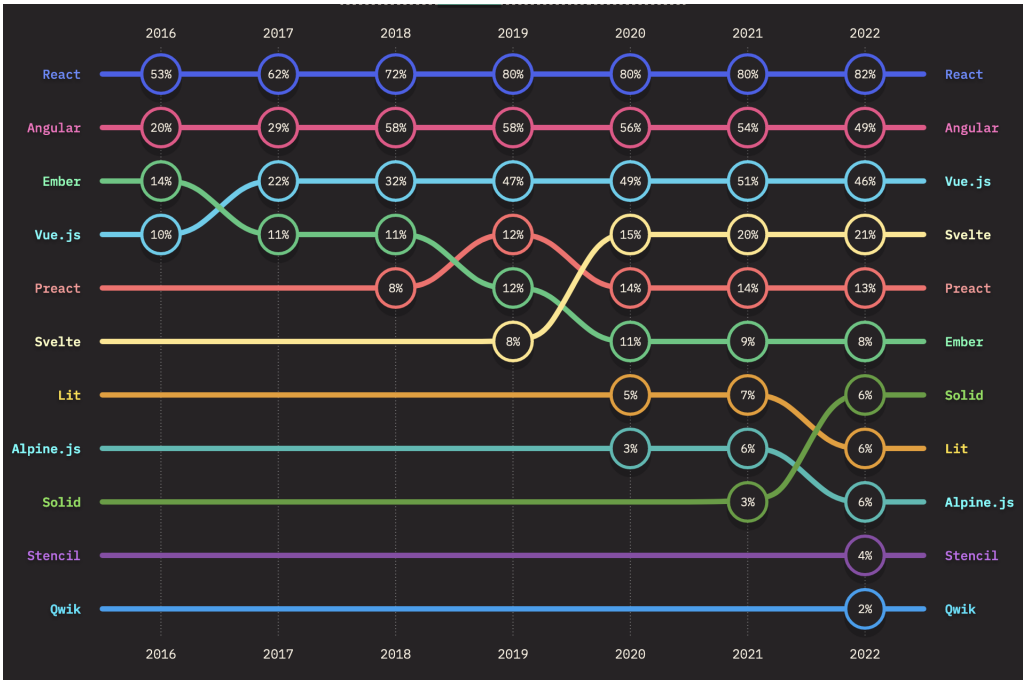

Figure 3.1.: State of JS: Most used frontend frameworks in 2022[1]

3.3.1. React

React (https://react.dev/), developed by *Facebook* and maintained by its successor *Meta*, has become the most widely used tool for building SPAs and is steadily leading the rankings for most used frontend frameworks both in the *StackOverflow* (Overflow, 2023) and the *State Of JS* (Greif & Burel, 2023b) polls. By definition it is not a framework, but a User Interface (UI) library that builds on other extensions to support state-management, routing and deployment functionality. Although it is not a framework itself, there are existing frameworks like *Next.js* (https://nextjs.org/) for the web and *ReactNative* (https://reactnative.dev/) for building mobile apps using native functionality. React makes use of JavaScript XML (JSX) which allows directly mixing inline HyperText Markup Language (HTML) with the JS or TypeScript (TS) code structure.

[1] Greif and Burel, 2023b

3.3.2. Vue.js

Vue.js (https://vuejs.org/) was developed by Evan You and is maintained by an international team of individuals. In the first years after its inception it had a rather marginal presence. This can be at least partially attributed to the fact that it originated in China and most of its supporting modules were localised in Chinese language. Over the years it grew in popularity and received much more international support, eventually overcoming the language barrier. Unlike *React*, it is billed as a "progressive framework" that provides very basic functionality for building reactive components, but also accommodates more complex use-cases (You, 2021). *Vue.js* builds on standard JS or TS, HTML and Cascading Style Sheets (CSS) to build components, recommending a simple template mechanism mixed with reactive substitutions. However, it also supports using JSX for specifying inline HTML within JS. As with *React*, there are extensions and frameworks like *Quasar* (https://quasar.dev/) and *Nuxt* (https://nuxt.com/) that enable even more sophisticated workflows application development and deployment.

3.3.3. Angular

Angular was initially released by *Google* in 2010 as *AngularJS* and officially discontinued in 2022 (https://angularjs.org/). A completely overhauled and currently used version 2 was then released in 2016 and is maintained by *Google*. It is different from *React* and *Vue.js* in that it is a complete framework that contains everything required to build and deploy an application and it explicitly recommends TS as a programming language. The framework also is less flexible in that it is opinionated and has its own set of best practices baked into the framework's structure.

3.4. Backend Libraries

3.4.1. Express

3.4.2. Koa

3.4.3. Fastify

3.5. Backend Frameworks

3.5.1. Nest JS

3.5.2. Meteor

3.5.3. Feathers

4. Methodology

4.1. Quantitative Analysis

4.1.1. Performance Testing

4.1.2. Time spent

4.2. Qualitative Analysis

4.2.1. Code Quality

4.2.2. Critical reflection

5. Implementation

5.1. Architecture

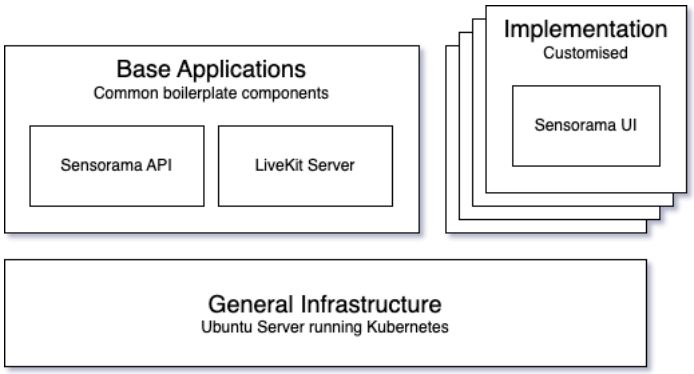

Figure 5.1.: The main components comprising the application architecture

5.1.1. Hardware

5.1.2. Software

5.2. Application infrastructure

While all the frameworks represented in 3.1 could be used to build an application as envisioned in this study, Vue is selected as the tool of choice, both due to the relatively high acceptance and the comparably steep learning curve. Additionally, it is already used in a number of applications designed by the author.

5.2.1. WebRTC Server

5.2.2. Database

5.3. Design Paradigms

5.3.1. Application Partitioning

5.3.2. Coding Style

5.3.3. Testing

5.4. Application Components

5.4.1. Web Frontend

5.4.2. API Backend

5.4.3. Native Utilities

6. Conclusion

6.1. Evaluation Results

6.2. Critical reflection

6.3. Outlook

Bibliography

Brandl, R. (2023, January 4). *The Most Popular Video Call Conferencing Platforms Worldwide*. Retrieved April 3, 2023, from https://www.emailtooltester.com/en/blog/video-conferencing-market-share/

Couriol, B. (2021, April 7). *10 Years after Inception, WebRTC Becomes an Official Web Standard*. Retrieved April 3, 2023, from https://www.infoq.com/news/2021/04/webrtc-official-web-standard/

Davis, J., Nguyen, T., & Simmons, J. (2023, February 1). *Interop 2023: Pushing interoperability forward*. Retrieved April 3, 2023, from https://webkit.org/blog/13706/interop-2023/

Fette, I., & Melnikov, A. (2011, November 1). *The websocket protocol*. Retrieved January 14, 2024, from https://datatracker.ietf.org/doc/html/rfc6455

Galli, M., Soares, R., & Oeschger, I. (2003, May 16). *Inner-browsing: Extending web browsing the navigation paradigm*. Retrieved January 11, 2024, from https://web.archive.org/web/20030810102320/http://devedge.netscape.com/viewsource/2003/inner-browsing/

Garrett, J. J. (2005, February 18). *Ajax: A new approach to web applications*. Retrieved January 11, 2024, from https://web.archive.org/web/20150910072359/http://adaptivepath.org/ideas/ajax-new-approach-web-applications/

GitHub. (2023, November 8). *Octoverse: The state of open source and rise of ai in 2023*. Retrieved January 14, 2024, from https://github.blog/2023-11-08-the-state-of-open-source-and-ai/

Greif, S., & Burel, E. (2023a, January 11). *The state of js: Most used backend frameworks in 2022*. Retrieved January 11, 2024, from https://2022.stateofjs.com/en-US/other-tools/#backend_frameworks

Greif, S., & Burel, E. (2023b, January 11). *The state of js: Most used frontend frameworks in 2022*. Retrieved January 11, 2024, from https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/

Overflow, S. (2023, June 13). *2023 stack overflow developer survey: Web frameworks and technologies*. Retrieved January 13, 2024, from https://survey.stackoverflow.co/2023/#section-most-popular-technologies-web-frameworks-and-technologies

Russell, A., & Berriman, F. (2015, October 8). *Progressive Web Apps: Escaping Tabs Without Losing Our Soul*. Retrieved January 11, 2024, from https://medium.com/@slightlylate/progressive-apps-escaping-tabs-without-losing-our-soul-3b93a8561955

Skarbez, R., Brooks Jr., F. P., & Whitton, M. C. (2017). A Survey of Presence and Related Concepts. *ACM Computing Surveys*, *50*(96), 1–39. https://doi.org/10.1145/3134301

Volpe, G. (2003, April 22). *Computational models of expressive gesture in multimedia systems*. Retrieved April 4, 2023, from https://theses.eurasip.org/theses/157/computational-models-of-expressive-gesture-in/download/

Web Audio API. (2021, June 17). Retrieved April 4, 2023, from https://www.w3.org/TR/webaudio/

WebRTC: Real-Time Communication in Browsers. (2023, March 6). Retrieved April 3, 2023, from https://www.w3.org/TR/webrtc/

WebXR Device API. (2023, March 3). Retrieved April 4, 2023, from https://www.w3.org/TR/webxr/

You, E. (2021, September 29). *Introduction: The progressive framework*. Retrieved January 14, 2024, from https://vuejs.org/guide/introduction.html#the-progressive-framework

Appendix Listing

A. Appendix