INF FAKULTÄT FÜR INFORMATIK

VIS
Lehrstuhl für Visualisierung

Abschlussbericht zum wissenschaftlichen Individualprojekt

Entwicklung eines multi-modalen HTML5-basierten Kollaborationswerkzeugs für medizinische 3D-Visualisierungen

Autor:
Matthias Graf

Betreut durch:
Steven Birr
Prof. Bernhard Preim

13. August 2013

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Computergestützte Lernwerkzeuge gewinnen in der medizinischen Ausbildung an Bedeutung. Insbesondere in der Anatomie kommen interaktive 3D-Visualisierungen zum Einsatz, um die klassische Ausbildung zu komplementieren. Im Rahmen des sogenannten integrierten Lernens (blended learning), dass die Vorzüge traditioneller Präsenzveranstaltungen mit denen von E-Learning zu verbinden versucht, ist es dabei wünschenswert zwischen Nutzern eines Lernportals Interaktion zu ermöglichen. Lernende können sich so untereinander und mit erfahrenen Medizinern austauschen, auch wenn sie sich nicht zur gleichen Zeit am gleichen Ort befinden. Dazu werden Kollaborationstechniken benötigt.

1.2 Problemstellung

Das Ziel des Projekts war die Entwicklung eines multi-modalen webbasierten Kollaborationswerkzeugs für medizinische 3D-Visualisierungen auf Basis des LiverAnatomy-Explorers [Birr et al., 2012a]. Im ersten, theoretischen Teil der Arbeit werden dabei zunächst Kollaborationskonzepte für den medizinischen Anwendungsfall mithilfe einer Literaturrecherche untersucht. Im zweiten Teil werden zur Lösung der Problemstellung 3 konkrete Mittel vorgestellt:

- Live-Synchronisation der 3D Szene
- Chat Funktion
- Konferenzsitzung mit Webkamera- und Mikrofoneinbindung

Alle Funktionen sollten dabei den folgenden Anforderungen genügen:

- echtzeitfähig
- beliebige Teilnehmerzahl erlaubt
- plattformübergreifend (durch die Nutzung neuester HTML5 Technologien)
- nahtlos eingeflochten in die bestehende Anwendung
- leicht bedienbar

Ausgehend von diesen Anforderungen ist eine prototypische Entwicklung erfolgt, um die Ideen zu erproben. Die Ergebnisse werden im Abschnitt Umsetzung vorgestellt. Die Ausführungen schließen mit einer Diskussion und Ausblick.

2 Vorüberlegungen

In diesem Abschnitt wird zunächst das Anwendungsszenario für die beschriebene Problemstellung vorgestellt. Daraus leiten sich für die Lösung wichtige Rahmenbedingungen ab. Es schließt eine Literaturrecherche daran an, in der wichtige Erkenntnisse aus früheren Arbeiten zum Thema zusammengetragen werden.

2.1 Anwendungsszenario

Rechnergestützte Gruppenarbeit (Computer Supported Collaborative Work) ist der Name des die Vorüberlegungen umfassenden interdisziplinären Forschungsgebiets. Kollaborationstechniken werden darin typischerweise anhand von 2 Achsen differenziert: Ort (fern oder ortsgleich) und Zeit (synchron oder asynchron) [Baecker, 1995, p. 741]. Für die gegebene Problemstellung geht es um ortsferne, synchrone Interaktionsmethoden. Konkret bedeutet das, dass mindestens 2 Menschen, die von unterschiedlichen Orten auf das Computersystem zugreifen, zur gleichen Zeit darüber zusammenarbeiten möchten. Ein mögliches Anwendungsszenario ist Onlinementoring. Ein unerfahrener Assistenzarzt annotiert interaktive 3D-Modelle in einem Websystem zur Planung eines chirurgischen Eingriffs (wie bei dem LiverAnatomyExplorer [Birr et al., 2012b]) und erhält dabei Unterstützung in Form von Hinweisen und Korrekturen von einem medizinischen Experten. Im folgenden Abschnitt werden einige der Kollaborationstechniken für solche und ähnliche Anwendungsszenarien vorgestellt, die in der wissenschaftlichen Literatur vorgeschlagen wurden.

2.2 Verwandte Arbeiten

Bereits im Jahr 1996 stellten Zikos et al. ein medizinisches System für synchrone Kollaboration vor [Zikos et al., 1996]. Ein Telekonferenzsystem wurde mit einer geteilten virtuellen Arbeitsumgebung kombiniert, um Vermerke in medizinischen CT-Bildern zu setzen. Die Zeigerposition jedes Teilnehmers ist für jeden Teilnehmer in Form unterschiedlich eingefärbter Zeiger sichtbar. Auch Vermerke bekommen eine jedem Teilnehmer zugeordnete Farbe. Das ist möglich, weil sich die Sicht auf die Daten (hier dem Bild) zwischen den Teilnehmern nicht unterscheidet. Ein Chat unterstützt die Kommunikation. Ein Videokonferenzmodul wird als Erweiterung vorgeschlagen.

Darauf aufbauend entwickelten Chronaki et al. ein umfangreiches webbasiertes Kollaborationssystem für die lokale Gesundheitsfürsorge in Zypern [Chronaki et al., 1997]. Es integriert geteilte Arbeitsplätze, ein Vermerksystem, E-Mail, Chat, persönliche Webseiten, Foren, Diskussionslisten, Ordner für medizinische Fälle. Viele dieser Techniken sind vorrangig für die asynchrone Interaktion geeignet. Als nachteilig werden die geringe Plattformunabhängigkeit beschrieben, sowie die Abhängigkeit von Latenz und Bandbreite der Anbindung für die Geschwindigkeit und damit Echtzeitfähigkeit der Techniken insbesondere für synchrone Kollaboration.

Ähnliche Bestrebungen gingen von dem Mitte der 90er Jahre in Schweden entstandenem Swedish Oral Medicine Network (SOMNet) aus. Zur Zusammenarbeit geographisch verteilter Einrichtungen entstand das SOMWeb [1] [Falkman et al., 2005]. Über die Bereitstellung der bloßen Technologie zur Kollaboration hinaus werden in der Arbeit Kriterien untersucht, die zu gelungenen Onlinekonferenzen beitragen. Hierzu zählen Gesprächsformalien, Struktur, Länge, Inhalte und Ziele. Telefonkonferenzen werden als Methode zum Austausch eingesetzt. Es wird fallbasiert gearbeitet. Obwohl alle Konferenzteilnehmer Material zu jedem Fall hinzufügen können, obliegt es dem Sitzungsleiter

[1] `http://www.somweb.se`

Notizen und Empfehlungen für den Fall in das System einzutragen, die sich aus der Konferenz ergeben haben [Falkman et al., 2008]. Damit wird ein Rechtesystem für die Implementierung solcher Plattformen vorgeschlagen.

Die technische Komponente beleuchten Charles Marion und Julien Jomier in ihrem Artikel zur Echtzeitsynchronisation der Interaktion in 3D-Visualisierungen [Marion and Jomier, 2012]. Von verschiedenen Endgeräten aus kann so mit einer 3D-Szene interagiert werde, dass alle Anwender die gleiche Sicht teilen. Möglich ist das in dem von ihnen vorgestellten Visible Patient System [2]. Damit ist es Teilnehmern möglich zum Beispiel während eines Vortrags auf ihrem eigenen Gerät nachzuvollziehen, über welche Inhalte gerade gesprochen wird, wenn der Vortragende ihnen darin etwas zeigen möchte. Das System beruht auf den Standards WebGL [3] und WebSockets [4]. Damit überbrücken sie viele Nachteile voriger Implementierungen. Lokales Rendering erübrigt im Gegensatz zu serverseitiger Bildgeneration (z.B. [Jourdain et al., 2010]) Netzwerklaufzeiten und verringert die Datenlast im Netz. Auch die Nutzung von WebSockets bietet im Gegensatz zu auf AJAX basierenden Verfahren Latenzvorteile. Hohe Plattformunabhängigkeit und Portiertbarkeit wird durch Nutzung offener Webstandards erreicht. Das steht im Gegensatz zu vorigen Vorschlägen, zum Beispiel Collaviz [Dupont et al., 2010], das zum Rendering auf Java OpenGL setzt, wofür extra Plugins und Programme benötigt werden. Diskutierte Nachteile von WebGL jedoch sind der hohe Leistungsanspruch an den Client, wenn komplexe 3D Graphiken dargestellt werden. Es kommt außerdem hinzu, dass die Szenendaten zunächst übertragen werden müssen.

Viele der für die Lösung der Problemstellung wichtigen Aspekte wurden bis hierher erwähnt. Dazu gehören die Bereitstellung einer Vielzahl von möglichen Interaktionsmodalitäten (Video, Audio, Text), um die Kommunikation zwischen Teilnehmern zu erleichtern. Native Browserunterstützung ist wichtig, um im Vorfeld der eigentlichen Verwendung keine Barrieren aufzubauen (wie der Installation benötigter Software oder Plattformabhängigkeit). Auch für die Benutzerschnittstelle selbst ist eine intuitive Gestaltung wichtig. Die Echtzeitfähigkeit der Techniken muss beachtet werden. Aus der SOMNet Forschung wird weiterhin vorgeschlagen, die charakteristischen Merkmale der Kooperation auch im System abzubilden, beispielsweise in Form einer geeigneten Rechteverwaltung. Diesen Gedanken folgenden geht es im nächsten Kapitel um die Umsetzung des Prototypen.

3 Umsetzung

Der folgende Abschnitt geht im Detail auf die Synchronisation der 3D-Szeneninteraktion und der Vermerke ein. Im Anschluss wird kurz die Benutzerschnittstelle beschrieben und dann auf die zugrunde liegende Softwarearchitektur eingegangen.

[2] `http://visiblepatient.eu`
[3] `https://www.khronos.org/registry/webgl/specs/latest/`
[4] `http://www.w3.org/TR/websockets/`

3.1 Synchronisation

Im Rahmen des Projekts wurde der LiverAnatomyExplorer um Kollaborationswerkzeuge erweitert, um das vorgestellte Anwendungsszenario eines Onlinementoring zu ermöglichen. Die 3D-Szene steht dabei im Vordergrund. Sie stellt Fälle von Lebererkrankungen dar. Die Kamera kann frei in der Szene positioniert werden. Es können Vermerke in der Szene durch Marken und dazugehörige Kommentare gesetzt werden. Um die Kollaboration zwischen Nutzern zu ermöglichen war es deshalb essentiell, dass die mit dem Setzen von Vermerken verbundenen Interaktionen zwischen ihnen synchronisiert werden, denn so könnten Fallentscheidungen genau nachvollzogen werden. Dazu wurde im Rahmen des Projekts die bidirektionale Synchronisation zwischen allen Nutzern umgesetzt. Synchronisiert werden alle Aktionen, die die Kameraeinstellungen (Position, Richtung) verändern (Zoom, Rotation, Translation). Außerdem wird das Setzen und Bearbeiten von Szenenmarkern für Vermerke synchronisiert.

Technisch wird das durch die Übermittlung der dazugehörigen JavaScript Befehle einschließlich Parametern realisiert. Diese werden dann bei den Zuhörern ausgeführt (eval). Das ist eine einfache Lösung, aber sehr unsicher, da so durch einen modifizierten Client beliebige Befehle auf den fernen Maschinen ausgeführt werden können. Andererseits ist der Code im Kontext des Browser Tab in einen Sandkasten eingebunden. Über den aktuellen Tab hinaus besteht also kein Sicherheitsrisiko. Das Datenvolumen, dass zur Synchronisation so übertragen werden muss, ist minimal. Alternativ ist auch denkbar, bei jeder Interaktion den gesamten Zustand der Anwendung zu übertragen. Das wäre erheblich datenintensiver, hätte aber den Vorteil, dass Clients auch einer laufenden Sitzung beitreten könnten, ohne asynchron zu werden. In der aktuellen Umsetzung ist dagegen der gleiche Ausgangszustand beim Start der Interaktion Voraussetzung, denn es werden immer nur Zustandsveränderungen übertragen.

Der Mauszeiger selbst wird nicht übertragen. Es ist daher in einem gewissen Umfang möglich, parallel zu arbeiten, zum Beispiel beim Setzen, Editieren und Kommentieren von Vermerken. Auch die Anzeigeeinstellungen zur Szene werden übertragen, also welche Teile des Modells zur Darstellung markiert (oder ausgeblendet) wurden. Zur Onlinekommunikation wurde im Rahmen des Projekts ein Webvideokonferenzsystem und ein Chat implementiert und nahtlos in den LiverAnatomyExplorer integriert.

3.2 Benutzerschnittstelle

Die Kollaborationswerkzeuge sind in das bestehende Webinterface eingebunden. Die konzeptionelle Gesamtansicht ist in Abbildung 1 zu sehen. Es zeigt die Standardanordnung der Elemente im Browserfenster. Hinzugekommen sind die Fenster für die Videodarstellung und den Chat. Diese sind fixiert am unteren Bildschirmrand positioniert. Links das Video der eigenen Webkamera, in der Mitte der Chat und Rechts das Video aller anderen Teilnehmer. Nehmen mehr als 2 Teilnehmer an der Konferenz teil, erweitert sich das Fenster entsprechend. Das Videobild maximiert sich seitenverhältniserhaltend in das es umgebende Fenster. Alle Fenster sind in Größe und Position veränderbar. Außerdem können sie geschlossen werden. Das ist zum Beispiel dann sinnvoll, wenn die Videofunktion

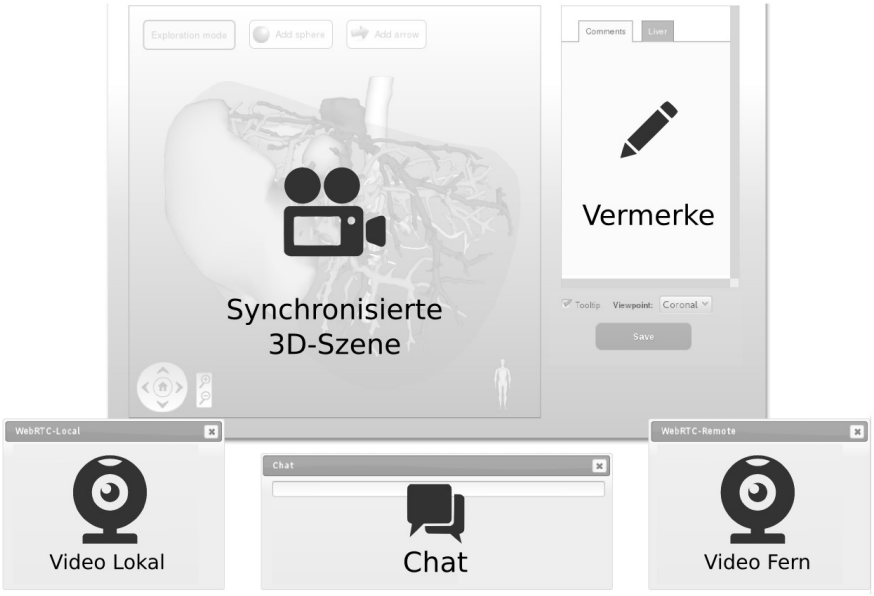

Abbildung 1: Konzeptionelle Gesamtansicht Nutzerschnittstelle

nicht verwendet wird, sondern nur telefoniert wird. Die Fenster wurden mit jQuery UI Dialogs [5] umgesetzt.

3.3 Architektur

Die Kommunikationsprotokolle die für die Kollaborationswerkzeuge zum Einsatz kommen, sind WebSockets für den Chat und die Szenensynchronisation sowie Web Real-time Communication (WebRTC) [6] für die Videokonferenztechnologie. Letzterer ist noch nicht abschließend definiert (working draft) und wird bisher nur in den Browsern Google Chrome und Mozilla Firefox implementiert (experimentell). Er definiert eine Schnittstelle zum Zugriff auf die Multimediahardware des Computers (Webkamera und Mikrofon) und deren Kommunikation im Netzwerk. Die eigentliche Datenübertragung (für Video und Audio) findet dabei direkt zwischen den Clients statt (peer to peer). Im Gegensatz dazu arbeiten WebSockets auf Server-Client-Basis. Chat und Szenensynchronisation werden also zentral über den Server abgewickelt. Deshalb muss auf dem Server neben dem eigentlichen HTTP-Webserver, der die Seite bereitstellt, auch noch ein WebSockets Server betrieben werden. Auch für WebRTC ist ein sogenannter Signalserver [7] nötig, um die Kommunikation zwischen sich verbindenden Clients zu arrangieren. Er verwaltet auch für gewöhnlich die verschiedenen Sitzungen, die parallel auf ihm laufen können. Im Falle dieses Prototyps wird davon jedoch der Einfachheit halber kein Gebrauch gemacht.

[5] `http://api.jqueryui.com/dialog/`
[6] `http://www.w3.org/TR/webrtc/`
[7] `https://github.com/andyet/signalmaster`

Es gibt zu jeder Zeit nur genau eine Sitzung und bisher keine Möglichkeit der Verwaltung dieser. Alle Teilnehmer sind gleichberechtigt. Signalserver und Websockets Server sind außerdem in einem Dienst kombiniert. Dieser Dienst ist eine nodeJS Anwendung, die auf der Socket.IO API [8] aufbaut, die wiederum WebSockets implementiert. Bei der Wahl der API wurde im Rahmen des Projekts außerdem das mod_pywebsocket [9] getestet. Einer der Vorteile ist die nahtlose Einbindung in den weit verbreiteten Apache HTTP Server. Auf der Seite der Clients müssen auch entsprechende JavaScript Bibliotheken eingebunden werden. Socket.IO stellt eine bereit und für die Nutzung von WebRTC wurde auf die SimpleWebRTC API [10] zurückgegriffen.

4 Diskussion

Im vorigen Abschnitt wurden die Werkzeuge beschrieben, mit denen die zu Beginn aufgestellte Problemstellung gelöst werden sollte. In diesem Abschnitt werden Stärken und Schwächen aufgezeigt und diskutiert.

Zunächst ist festzustellen, dass bisher keine umfangreichen, systematischen Tests der implementierten Kollaborationswerkzeuge erfolgt sind. Weder ihre Funktionsfähigkeit ist im Detail erprobt worden noch ihre Eignung für das Anwendungsszenario. Die folgenden Ausführungen basieren stattdessen auf einfachen Beobachtungen, nicht-rigorosen Tests und Einschätzung auf Basis des Wissens um die zugrunde liegende Softwarearchitektur.

Die in der Problemstellung vorgeschlagenen Kollaborationswerkzeuge sind im vollen Umfang im Prototypen umgesetzt worden: Live-Synchronisation der 3D Szene, die Chat-Funktion und die Konferenzsitzung mit Videokamera- und Mikrofoneinbindung. Alle sind mit Einschränkungen voll funktionsfähig. Einschränkungen ergeben sich zunächst aus der prototypischen Natur der Implementierung. Zum Beispiel ist der Sicherheitsaspekt der Szenensynchronisation nicht berücksichtigt worden. Außerdem ist die Synchronisation nur korrekt möglich, wenn alle Clients ausgehend vom gleichen Ausgangszustand interagieren. Tritt ein Client einer bereits laufenden Sitzung bei, bleibt seine Szene asynchron. Dieses Problem zu beheben bedarf einer weitreichenden Anpassung der internen Programmierschnittstelle. Ein weiterer wesentlicher Vorteil einer solchen Anpassung ist, dass damit selektive Synchronisation möglich wäre. Eine entsprechende Benutzerschnittstelle könnte es erlauben, zu wählen, welche Art von Interaktion synchronisiert wird - zum Beispiel nur Kamera oder nur Vermerke.

Es gibt derzeit auch keine Sitzungs- oder Rechteverwaltung. Denkbar ist hier eine Erweiterung des Prototypen, um mehrere Sitzungen zu ermöglichen, den Nutzer eine Sitzungsauswahl bereitzustellen und Rechte zu vergeben. Für das Szenario eines Vortrags könnte man das Recht zur Interaktion allein dem Vortragenden geben, während die Zuhörer sich entscheiden, ob sie ihre eigene Sicht an die des Vortragenden koppeln möchten (folgen) oder selbst steuern möchten (entkoppelt und asynchron).

Erste Tests der Videokonferenzfunktion haben ergeben, dass sie nur unter bestimmten

[8] `http://socket.io/`
[9] `http://code.google.com/p/pywebsocket/`
[10] `https://github.com/HenrikJoreteg/SimpleWebRTC`

Umständen funktioniert. In lokalen Netzwerken funktioniert sie. Allerdings ist die Übertragung des Videos bei einem Test über das Internet fehlgeschlagen. Vermutet wird, dass das SimpleWebRTC Framework die Firewalls nicht überwinden kann. Das schränkt die Funktionsfähigkeit der Kollaborationswerkzeuge derzeit am meisten ein. Außerdem ist eine erfolgreiche Verbindung bisher nur zwischen Firefox-Browsern möglich. Chrome käme zwar auch in Frage. Allerdings erfordert dieser Browser anders als Firefox für die Übertragung des Videosignals, dass die Verbindung verschlüsselt ist. Das ist derzeit nicht der Fall. Es ist allerdings zu erwarten, dass der WebRTC Standard in Zukunft von mehr Browsern implementiert wird und deren Interoperabilität verbessert wird. Es gibt derzeit keine bessere Alternative, die eine webbasierte, pluginfreie Umsetzung der Videokonferenzfunktion ermöglicht.

Weiterhin ist bei Tests aufgefallen, dass das Videosignal unter Umständen einfrieren kann, falls die Videofenster verschoben werden. Die Ursache ist ungeklärt.

Die Geschwindigkeit der Techniken ist stark von der Netzwerkqualität und der Leistung der Endgeräte abhängig. Chat und Szenensynchronisation sind im Vergleich zur Videokonferenzfunktion ressourcensparend. Letztere benötigt hohe Rechenleistungen zur Verarbeitung der Video- und Audioströme (insbesondere kodieren und dekodieren). Ist allerdings eine hohe Bandbreite, geringe Latenz und hohe Rechenleistung aller Endgeräte gegeben, sind alle Kollaborationswerkzeuge echtzeitfähig. Die Leistungsanforderungen steigen mit der Teilnehmerzahl linear.

Im Rückblick auf die Anforderungen ist damit zusammenfassend festzustellen, dass die Echtzeitfähigkeit und Plattformunabhängigkeit nur bedingt gegeben sind; eingeschränkt durch die Videokonferenzfunktion. Alle weiteren Anforderungen, eine beliebige Teilnehmerzahl, die nahtlose Einbindung in die bestehende Anwendung und die leichte Bedienbarkeit konnten in vollem Umfang erfüllt werden.

5 Zusammenfassung & Ausblick

In diesem Individualprojekt ist es gelungen, eine bestehende medizinische, webbasierte Lern- und Austauschplattform um multi-modale Kollaborationstechniken prototypisch zu erweitern, die Nutzern es auf einfache Weise ermöglicht, sich online zu einem Fall auszutauschen und dabei interaktiv in einer synchronisierten 3D-Visualisierung zu interagieren. Da das System online verfügbar ist und bis auf Einschränkungen bei der Videokonferenzfunktion plattformunabhängig und ohne die Installation von Plugins funktioniert, ist es möglich, von beliebigen Orten und Systemen aus miteinander zu kollaborieren.

Im zukünftigen Arbeiten zum Projekt sollte eine Nutzerstudie durchgeführt werden, um die Eignung des Prototypen für das Anwendungsszenario empirisch zu ermitteln. Eine genauere Analyse der Anforderungen ist dafür essentiell, um Kriterien für die Eignung festzustellen. Darüber hinaus scheint es ratsam zu sein, über die technischen Aspekte hinaus auch lerntheoretische und soziale Kriterien in die Betrachtungen einfließen zu lassen.

Das Projekt ist als freie Software (GPLv3 [11]) veröffentlicht [12].

Literatur

Ronald M Baecker. *Readings in Human-Computer Interaction: toward the year 2000*. Morgan Kaufmann Pub, 1995.

Steven Birr, Jeanette Mönch, Dirk Sommerfeld, and Bernhard Preim. A novel real-time web3d surgical teaching tool based on webgl. In *Bildverarbeitung für die Medizin 2012*, pages 404–409. Springer, 2012a.

Steven Birr, Jeanette Mönch, K.J. Oldhafer, Uta Preim, and Bernhard Preim. Der LiverAnatomyExplorer: Ein web- und fallbasiertes Trainingssystem für die Anatomieausbildung. In *CURAC*, pages 6–10, 2012b.

Catherine E Chronaki, Dimitrios G Katehakis, Xenophon C Zabulis, Manolis Tsiknakis, and Stelios C Orphanoudakis. Weboncoll: medical collaboration in regional healthcare networks. *Information Technology in Biomedicine, IEEE Transactions on*, 1(4):257–269, 1997.

Florent Dupont, Thierry Duval, Cédric Fleury, Julien Forest, Valérie Gouranton, Pierre Lando, Thibaut Laurent, Guillaume Lavoué, Alban Schmutz, et al. Collaborative scientific visualization: The collaviz framework. In *JVRC 2010 (2010 Joint Virtual Reality Conference of EuroVR-EGVE-VEC)*, 2010.

Goran Falkman, Olof Torgersson, Mats Jontell, and Marie Gustafsson. Somweb-towards an infrastructure for knowledge sharing in oral medicine. *Studies in health technology and informatics*, 116:527–532, 2005.

Göran Falkman, Marie Gustafsson, Mats Jontell, and Olof Torgersson. Somweb: a semantic web-based system for supporting collaboration of distributed medical communities of practice. *Journal of medical Internet research*, 10(3), 2008.

Sebastien Jourdain, Utkarsh Ayachit, and Berk Geveci. Paraviewweb, a web framework for 3d visualization and data processing. In *IADIS International Conference on Web Virtual Reality and Three-Dimensional Worlds*, volume 7, 2010.

Charles Marion and Julien Jomier. Real-time collaborative scientific webgl visualization with websocket. In *Proceedings of the 17th International Conference on 3D Web Technology*, pages 47–50. ACM, 2012.

Marios Zikos, Constantine Stephanidis, and Stelios C. Orphanoudakis. Comed: Cooperation in medicine. In *Proceedings of EuroPACS*, volume 96, pages 88–92. Citeseer, 1996.

[11] `http://www.gnu.org/licenses/gpl-3.0.html`
[12] `https://github.com/rbyte/LiverAnatomyExplorer/tree/synchronisation`