

UNIVERSITY OF CALIFORNIA
Santa Barbara

Re-framing the World Wide Web

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Media Arts & Technology

by

August Black

Committee in Charge:

Professor Marko Peljhan, Chair

Professor Rita Raley, Ph. D.

Professor Curtis Roads, Ph. D.

September 2011

UMI Number: 3481948

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent on the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3481948

Copyright 2011 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

The Dissertation of
August Black is approved:

Professor Rita Raley, Ph. D.

Professor Curtis Roads, Ph. D.

Professor Marko Peljhan, Committee Chairperson

September 2011

Re-framing the World Wide Web

Copyright © 2011

by

August Black

To my parents and brother.

Acknowledgements

Writing a dissertation is hard work requiring a large amount of intellectual, emotional, and financial stimulation and support. The realization of this dissertation, including the development of myself as a person, has been generously supported in this manner by a number of individuals and institutions.

Among the institutions that have supported me in my endeavors, I thank the Media Arts & Technology graduate program, the Art and Film and Media Studies department, the NSF IGERT doctoral program, the Interdisciplinary Humanities Center at UCSB, the Philip & Aida Siff Educational Foundation, and the BioImage Research Lab at UCSB.

I would like to thank, in no particular order, a number of individuals from UCSB who have been *there* for me. I thank Wesley Hoke Smith for sharing 5 years of living and breathing space with me. I thank Alex Norman for putting the “T” in up and for the new tooth. I thank Bob Sturm for the days we spent surfing. Thanks go to Bo Bell for your continued friendship and positive attitude. To Pablo Colapinto for being a matching piece in the puzzle. I’d like to thank Angus Forbes for his hoola-hooping and music sessions. To my other colleagues at MAT: Greg Shear, Lance Putnam, Mike Winters, Rama Hoetzlein, Charlie Roberts, Graham Wakefield, and Karl Yerkes.

I would also like to thank a number of friends and colleagues outside the University. Bob Adrian X has been more influential in the thoughts and development of this thesis than can be stated here. Heidi Grundmann first initiated me to radio art and larger ideas about media, art, and technology. I thank my fellow artists at alien productions, with whom I have lived, played, and worked for many years: Andrea Sodomka, Martin Breindl, and Norbert Math. More alienation! Steve Bradley, Sophea Lerner, Elizabeth Zimmermann, Federico Bonelli, Cecile Landman, Michael Mastrotatoro, Sabine Maier, Elfriede zeichen system, Asia Sumyk, Justine Gerenstein, and Kirsten Sanft. I'd like to thank Patty Desmond for the past 2 years of co-habitation and friendship. I thank Humberto Fossi for being a real *buey* . I thank Anna Scherz for her indirect but significant contributions to this dissertation.

I'd like to thank Rupert Huber and Markus Seidl for years of artistic collaboration and friendship. I thank Tom Sherman for being my teacher and for being a personal human. I'd like to thank Denis "Jaromil" Roio for introducing me to the pleasures of hacking when we embarked on programming the Multi-User Streaming Environment together. I thank Laura Devendorf for the world of recursion she has given me, without which this dissertation would probably still be writing itself.

I would also like to thank all those that have given me advice and help on IRC #nerd-art, #vala, #gtk+, #gstreamer, #libpeas, #dataflow, especially ax, juergbi, Lethalman, jryan, mhr3, and many others that I have forgotten.

I'd like to thank the advisors I've had along the way. Dr. B.S. Manjunath supported me for over 2 years in the BioImage lab. George Legrady initially brought me to UCSB. My committee members, Dr. Curtis Roads and Dr. Rita Raley, have provided steady interest, stimulation and support.

I'd especially like to thank my committee chair, Prof. Marko Peljhan, who has included me in his many adventures, not only this one.

Without my parents and brother - Bob, Gail, and Omar Black - I would not be who I am.

Curriculum Vitæ

August Black

Background

Born, 1975, in Baltimore, Maryland, USA. August Black is an artist, researcher, and developer of new tools and instruments. His research is based in the general overlap of culture, craft, and code. His artistic work is a mix of radio, video/film, installation and software, and has been shown internationally at venues such as the Ars Electronica Festival (Linz, Austria), Künstlerhaus (Graz, Austria), the pixelache festival (Helsinki Finland), the ICC (Tokyo, Japan), the Transmediale festival (Berlin, Germany), the Freud Museum (St. Petersburg, Russia), and the Píksel festival (Bergen, Norway). His research interests include the politics and production of open media formats (free radio, film/video, free software) and distributed communication structures for the web.

Education

- | | |
|------|--|
| 2011 | PhD in Media Arts & Technology, University of California, Santa Barbara. |
| 2005 | Master of Science in Media Arts & Technology, University of California, Santa Barbara. |
| 1997 | Bachelor of Fine Arts, Syracuse University. |

Awards

- | | |
|-----------|--|
| 2011 | UCSB Dean's Advancement Award |
| 2010-11 | Philip & Aida Siff Foundation Graduate Fellowship |
| 2009-2010 | UCSB Interdisciplinary Humanities Center pre-doctoral fellowship |
| 2003-2005 | National Science Foundation Fellowship in Interactive Media (IGERT) |
| 2000-2003 | Multiple grants from the Austrian Council for the Arts (Bundeskanzleramt) for independent projects |
| 1996 | Clements Scholarship, Syracuse University |
| 1993-1997 | Syracuse University Merit Scholarship for Visual Art |
| 1993-1997 | Syracuse University Chancellors Scholarship |
| 1993-1997 | Maryland Artist Equity Scholarship |

1993 Maryland Governor's Citation for the Arts

Experience

W2011 Ecology Department, UCSC – Sta. Cruz, CA – Research associate on CAMEO NSF. An art-science collaborative project building a database of food web interactions.

F2010 Art Department, UCSB – Sta. Barbara, CA – Teaching Associate for Art 122, advanced digital art.

W2010 Film and Media Studies, UCSB - Sta. Barbara, CA - Teaching assistant for Film Theory 192A.

2008-9 Self-Employed - San Miguel de Allende, Mexico - Artist & Software Developer

S2008, W08, F07 Film and Media Studies, UCSB - Sta. Barbara, CA - Teaching assistant for Film Analysis.

S2008, W08, F07 Art Department, UCSB - Sta. Barbara, CA - Teaching assistant Digital Art Studio

2005-2007 BioImage Research Lab, UCSB - Sta. Barbara, CA - Artist in Residence/Research Associate. Research in the areas of volume rendering, multimedia database applications and high resolution tiled display systems.

W2005 Media Arts & Technology - Sta. Barbara, CA - Teaching assistant for Visual Design Through Algorithms.

2000-2003 Self-Employed - Vienna, Austria - Artist, Graphic Designer, Web Developer, & Translator

1998-2000 Ars Electronica Futurelab - Linz, Austria - Artist/Researcher in Residence

1996-98 & 2001-3 Austrian National Radio - Vienna, Austria - Developer & Production Assistant for Kunstradio,

Skills

Foreign Languages: Fluent in German. Advanced Spanish.

Computer Languages: Expert in C, C++, Vala, Ecmascript, HTML, XML, Python, SQL, PHP. Proficient with Perl, Java, LaTeX, Unix shell.

Artistic: Proficient in pencil, ink, paint and brush, as well as computer aided print and screen design. Experienced at designing and building of prototypical objects in various materials.

Lectures and Workshops

2011	LabSurLab festival - 4 day workshop on artistic practices of streaming media. Medellin, Colombia
2010	Isuma TV - workshop on free software video editing. Participant in Arctic Perspective Initiative. Igloolik, Nunavut, Canada.
2009	Centro Multimedia – free software workshop. Centro Nacional de las Artes, Mexico City, Mexico.
2009	Red de los Medios Libres – workshop on free software for video. Mexico City, Mexico.
2009	Kyoto Art University - Artist lecture. Kyoto, Japan.
2009	Biblioteca Publica – Lecture on free software and green technologies. San Miguel de Allende, Mexico.
2008	Transmediale Festival – Artist presentation. Berlin, Germany.
2006	Kiasma Museum - Artist presentation of userradio. Helsinki, Finland.
2004	Kunsthall – Artist presentation. Bergen, Norway.
2003	Sibelius Academy – 4 day workshop on artistic radio production. Helsinki, Finland.
2002	Kunstuniversitaet Linz – Artist lecture. Linz, Austria.
1997	Akadamie der Kunst – Artist lecture. Vienna, Austria.

Selected Publications

August Black:“UserRadio (2002)” In *Transmission Arts: Artists & Airwaves*, New York 2011, PAJ, ISBN 978-1-55554-151-4

August Black:“An Anatomy of Radio” In *Re-Inventing Radio: Aspects of Radio as Art*, Frankfurt/Main 2008, Revolver, ISBN 978-3-86588-453-4

August Black:“Blindsight is 20/20” In *Re-Inventing Radio: Aspects of Radio as Art*, Frankfurt/Main 2008, Revolver, ISBN 978-3-86588-453-4

August Black:“Userradio” In *Proceedings of the 12th annual ACM international conference on Multimedia* , New York, NY, 2004, ACM, ISBN:1-58113-893-8

Exhibitions

- 2008-9 Der Gedankenprojektor - ICC Tokyo, Japan. kunsthaus Vienna, Austria. Ars Electronica Festival Linz, Austria. Künstlerhaus Graz, Austria.
- 2004 Propagandada - group exhibition at The AIM*Space Gallery, Los Angeles, CA.
- 2003 The Hard Drive Orchestra - Freud Museum in St.Petersburg, Russia.
- 2002 Simple World - Marek room at SchloßWolkersdorf, Austria
- 2000 Gateways - with alien productions Siebenbrunnenplatz, Vienna, Austria
- 1999 Paper Moon - as part of the Synworld. Museumsquartier, Vienna, Austria
- 1998 Entry Point - for Kunst in der Stadt II. with Markus Seidl, Bregenz, Austria

Performances

- 2005 "Slackford and Sun" - Live performance at the Aaniradio festival in Helsinki, Finland.
- 2004 DataDada - Live performance at the piksel festival in Bergen, Norway.
- 2001 48 - Live performance at the Network Community Congress in Graz, Austria.
- 2001 Can you See the Light? - Live Television performance as part of V-stream streaming media festival.
- 2001 kopenhagen kanon - with berlinertheorie at the Den Anden Opera in Copenhagen, Denmark.
- 2001 Alien City - with alien productions at Hamburger musikfest.
- 2000 Ain't that a Shame - with berlinertheorie at the Berliner Sophien-Saal, Berlin.
- 2000 memoria - with berlinertheorie. pro musica nova festival, Bremen.

Participation in Communication Projects

- 2008 Radio Oltranzista - week-long temporary FM radio station at the Weerwoord festival. De Balie, Amsterdam, Netherlands.
- 2007 Radio Oltranzista - week-long temporary FM radio station at Melkweg. Amsterdam, Netherlands.

- 2005 Fadaiat - With Marko Peljhan & RX-TX, Tarifa, Spain & Tangier, Morocco.
- 2005 Aaniradio - week-long experimental radio on a temporary FM frequency in Helsinki, Finland.
- 2001 a retrospective of real events - with Manfred Sölner. live contribution to the Acoustic Space lab in Irbene, Latvia.
- 1997 Recycling the Future I-IV - as part of the production team at ORF Kunstradio. Documenta X in Kassel, Germany. Ars Electronica Festival in Linz, Austria. Radio Lada in Rimini, Italy. ORF Funkhaus in Vienna, Austria.
- 1996 Rivers & Bridges – global telecommunication project produced at ORF Kunstradio, National Radio Station of Austria.

Abstract

Re-framing the World Wide Web

August Black

The research presented in this dissertation studies and describes how technical standards, protocols, and application programming interfaces (APIs) shape the aesthetic, functional, and affective nature of our most dominant mode of on-line communication, the World Wide Web (WWW). I examine the politically charged and contentious battle over browser market share and how this drives the seemingly *open* development of technical standards and the implementation of new features. I present a new and alternative browser prototype and communication framework called the Underweb that provides partial solutions to the problem space of the WWW. Parallel to the non-linear development dynamic of the amorphous electronic infrastructure of the WWW, the Underweb provides a more user-elegant set of technologies that gives developers and users the ability to not only read, but also to write, edit and publish in this system without third-party involvement.

Contents

Acknowledgments	v
Curriculum Vitæ	viii
Abstract	xiii
List of Figures	xvi
1 Introduction	1
1.1 What is the World Wide Web, thing and metaphor	11
2 Background	17
2.1 Internet	19
2.2 Free Software	24
2.3 Browser Homology	32
2.4 The Evolution of HTML	45
3 Problem Statement	61
3.1 There are no alternatives	65
3.2 Academy to commerce	66
3.3 Centralization	74
3.4 Standardization and generativity	81
3.5 Black-box technologies	89
3.6 Monolingual	91
3.7 Audio/Video playback	94
3.8 Read, Write, Publish	102
3.9 Literacy	108
3.10 Unknown and multiple formats	113

3.11 Rectangles	122
4 Methodology	129
5 The Underweb	132
5.1 Technical Development of the Underweb	133
5.2 The Underweb Framework	140
5.3 Libmentiras	149
5.3.1 Graphics	151
5.3.2 Text	153
5.3.3 Interactivity and mouse events	157
5.3.4 Animation and timing	158
5.3.5 Images	160
5.3.6 Custom drawing	164
5.3.7 Widgets	164
5.3.8 Video	167
5.3.9 Audio	170
5.3.10 Editing	173
5.3.11 Networking	175
6 Discussion	179
6.1 Reinventing the wheel?	182
6.2 Too late?	184
6.3 Future Research	185
6.4 Significance & Limitations	189
7 Conclusion	192
Bibliography	200

List of Figures

2.1	packet header encapsulation.	19
2.2	packet header encapsulation 2.	21
2.3	tcp-ip hourglass figure.	23
2.4	Timeline of browser genealogy	36
2.5	WWW poster	38
2.6	HTML markup code example	47
2.7	HTML example in Firefox	47
2.8	HMTL markup in Lynx	47
2.9	Styled HTML element in HTML 1.0 and CSS	54
3.1	Network Overlay.	76
3.2	Digital photo montage	119
3.3	Graphical example	124
3.4	SVG source.	127
5.1	The Underweb browser.	141
5.2	The Underweb GTK+ example.	142
5.3	The Underweb WebKit example.	147
5.4	Libmentiras graphic examples.	152
5.5	Libmentiras curved line examples.	153
5.6	Libmentiras markup and text wrapping example.	154
5.7	Libmentiras text wrap and rotation.	156
5.8	Libmentiras mouse events.	158
5.9	Libmentiras animation.	159
5.10	Libmentiras surface and matrix.	162
5.11	Libmentiras images.	163
5.12	Libmentiras custom drawing.	165
5.13	Libmentiras widgets.	166

5.14 Libmentiras page transparency.	167
5.15 Libmentiras video playback.	168
5.16 Libmentiras video encoding.	169
5.17 Libmentiras audio	171
5.18 Libmentiras Puredata	172
5.19 Libmentiras editing	174
5.20 Libmentiras TCP server	176
5.21 Libmentiras TCP client	177
5.22 Libmentiras HTTP server	178

Chapter 1

Introduction

The Web as I envisioned it, we have not seen it yet. The future is still so much bigger than the past.
Tim Berners-Lee

This dissertation offers a new framework, both theoretical and practical, for an alternative and more elegant World Wide Web (WWW). It consists of this written document and a computer application. The written document provides an explication of the current development trajectory of the WWW and an argument for the construction of a completely new communications infrastructure based on free software and its development methodologies instead of the existing ad-hoc industry-driven standards model. The application framework that I have developed aims to provide a practical and usable prototype that successfully demonstrates how the WWW could be different than what it is today. What I propose is a re-structuring, or what I subjectively call a re-framing, of the intent, purpose, and development methodology of the WWW infrastructure.

The absolute main driving force of this dissertation, including both the written thesis and software components, is simply to discuss and analyze the varied and multi-dimensional problem space of the World Wide Web (WWW) from a socio-technological perspective. The emphasis is on how the software components of the WWW create a space of communication, how that space has changed, and how it guides, informs, and to some extent constrains social behaviour under the conflicting forces of existing neo-liberal paradigms - the very paradigms that create and sustain the technical possibility of a WWW. It is a very large problem with many non-linear¹ interwoven idiosyncrasies, and to which I offer partial technical and theoretical solutions in the form of a new browser prototype that I call the Underweb.

Unlike most social theory that focuses only on the affective reflection technology has on social bearings, I focus on the convoluted technological determinants - the networks, the protocols, the software platforms and formal logics - that give shape to the outward expressive capabilities of the WWW including the influence it has on user behaviour. I am looking at the micro-structural level up (outward), instead of from the macro-structural level down (inward). From this perspective, I

¹The term non-linear has many technical and non-technical meanings. I use it loosely in this dissertation to highlight and describe effects that don't directly trace back to causes. In some ways it means unintended consequences such as is demonstrated by the cobra effect where solutions to a problem offer new problems:https://secure.wikimedia.org/wikipedia/en/wiki/Cobra_effect. In other cases, it refers to the general chaotic tendencies of interconnected systems.

look to draw correlations between underlying technical form and outward-laying affect. It is indeed a speculative argument as the irrational economic behaviour of the software industry, including the many successful developments in free software, eludes classical economic theory, and much to the confusion and dismay of many. Like many practices in art, the research here is speculation on the present and future.

To critical theorists, my written analysis may all seem too technical and too focused on the banal and minute details of technical interconnectedness. To technologists that study and create the technical objects of discussion, my proposal and prototype here may appear to be too general and/or abstract. I aim at the middle ground: to synthesize macro and micro perspectives.

My perspective is also techno-deterministic at the core. I find both direct and indirect relationships between technologies and human capabilities - without the possibilities that the automobile provides, there is no need or possibility for suburbs; without easy access to publishing on the WWW, there are no possibilities for wikileaks and the Arab Spring that just occurred in 2011. My investment in this perspective comes, however, with some doubt as to how far any observer can draw the relationship between given technologies and user behaviour - the same publishing technologies that provided a catalyst for the Egyptian spring are also the same technologies used to make surveillance on populations online.

Technologies do not function in a vacuum. They are made and maintained by individuals and collectives with their own set of logics and ideologies. For this dissertation, the logics and ideologies that are embedded within our technologies, but yet remain unarticulated are of concern.

The admittedly broad theme of the dissertation opens up at the place where technologies give soft determinant form to everyday life. I am motivated to do this project by a sense of the overwhelming commercial influence that has grown in the way our communications technologies have advanced. The WWW experience has evolved from an open almost-anarchic construct ruled by direct user participation into more centralized (and centralizing) tendencies. In the early days, the WWW consisted mostly of individual users who put up web pages on their own web servers. As more commercial investment in the WWW grew, many firms developed frameworks to provide services such as email (Gmail), messaging (Facebook, AIM, etc.), content publishing (Youtube, blogs), micro-blogging (twitter), and search (Google) at a cost. The cost to the user is that they must give up their personal data (in the case of Gmail and Facebook, this is often very personal and sometimes incriminating data) to the company at hand. With this data, private companies can build a powerful wealth of informational resources. On the surface, they can target users with pin-point advertising fit to their online behaviour and content. Beneath the surface and behind the scenes, a much stronger, but invisible and

barely describable, set of actuarial and analytical methods of cultural persuasion take place.

In this sense, I consider technologies in general to be as neutral as architectures or languages. A portion of the dissertation is dedicated to the problem space of the WWW that includes outlining and describing the major determining technological factors which remain to a large degree unnoticed by both the general public as well as to critical media theory.

Furthermore, the overarching metaphors of rhizomatic web-like decentralization have fueled both legitimate and illegitimate excitement for new communications technologies. The purpose of the Underweb software project is to refuel this excitement and shed light on possible and as yet unseen solutions to deliver (again) more decentralization to the net. To that end, I believe this development offers both cultural and technical knowledge in a very concrete and usable form of software.

I call my application prototype the Underweb because, unlike current browsers and their up-and-coming HTML5 standard, it exposes the underlying APIs to the developer and WWW participant instead of wrapping them in a constrained and mono-linguistic framework. The Underweb software that I have built is meant to provide only a minimal layer of infrastructure upon which the developer is given five main things that are not in HTML5 or the WWW system in general:

- greater flexibility in programming multimedia content
- greater extensibility of the browser
- the tools to read, write and publish online
- plausible methods for the non-technically-literate to write and publish, including graphical editing
- a methodology for future development of the browser itself

The Underweb is programmable in many languages (C, Vala, Python, Javascript) with many bindings to the most popular free software multimedia and networking libraries. These are the same libraries that are used by current popular browsers, but are concealed by their monolithic framework that forces development to remain within the confines of HTML/CSS/Javascript and the limited APIs that are proposed (but still underdeveloped) by the HTML5 standards. My Underweb framework demonstrates a more powerful and more user-elegant solution than what is already possible with HTML5.²

²I introduce the term *user-elegant* here as an alternative term to *user-friendly*. In the context of personal computing, friendliness is meant as a way to lure users into a financial, personal, and psychological engagement with computational technology. Where the technology was often perceived as overly-complicated and in some ways detrimental to users, friendliness was a clever, although collectively unintended, public relations strategy that emphasized the intelligence of computers and their ease of use. By way of side-effect, this PR campaign also emphasized the incapacity of users to learn and adapt to new methods. Under user-friendliness, the computer is close the user's peer level, and a user requires next to no education or tech-smarts to use it. Elegance, on the other hand, emphasizes the intelligence of users. It emphasizes solutions and interfaces for computational technologies that are simple, but not insulting of a user's ability. New technologies may, in some cases, require a learning curve, but offer a much more powerful and *empowering* way of interacting with machines.

Where possible, I offer a comparison of relevant functionalities in the proposed HTML5 standard and my Underweb framework. These, however, should not be seen as definitive because each framework or set of APIs is not only in a constant state of progression, but could also be implemented into the other. This is simply the very nature of software itself, especially of free and open source software. This is further underscored by the fact that the software infrastructure of the WWW, including any conceivable alternative, is not a singular concept, but many different softwares including servers, clients, and multiple browsers that are developed and distributed by various vendors. Furthermore, the purpose of the software I write cannot be specified as I am building a communication ecosystem made of software that is built to run other softwares. It is for these reasons that I think it makes more sense to discuss probable trajectories and plausible underlying paradigms rather than discussing singular implementations and their temporary solutions.³

The written thesis and the software project of this dissertation are linked under the following hypothesis:

Change: Unlike the middle layers of the IP stack, the WWW can and will change.

100 years from now, it will not be the same format that it is today, just as the WWW is incompatible with older iterations of itself.

³In this sense, software is as flexible and elastic as ideology.

Hypertext vs Application: The initial underlying impetus of the WWW was hypertext. The focus has now shifted to bring more and more functionality into the browser through additional APIs. As this happens, the browser becomes very much like a desktop. However, it adds an additional soft layer between the individual user and his or her desktop. It also becomes a soft layer between the user and other users online. The focus must now shift to the abstract exchange and linkage of all known and as yet unknown kinds of technically formatted digital information, not just hypertext.

Centralization: Centralization of the WWW into third-party server systems such as Google and Facebook is socially dangerous because it places too much informational and political capital into the hands of private holders. It is also technically unnecessary if users have easy-to-use tools and the technical literacy to publish and maintain their own content. The unspoken main focus of the new HTML5 protocols is to make it technically easier for commercial companies to acquire and store data on users.

Commercialisation: Unlike the development of the Internet itself, which was created by radical academic engineers with the clear intent (including the education and financial support) to build a royalty free and universal digital communication platform, the WWW was left to free-market commercial

industries. Little to no academic research is involved in creating the WWW protocols, APIs and browsers. This dissertation argues for the necessity to bring such development back to academia (or at least out of the commercial industries). It also offers an initial prototype.

No blackbox technologies: Plugins exist to extend the functionality of the WWW. The most important of these, such as Flash and Silverlight, are blackbox technologies that conceal their inner workings from the user and developer. These have no part in a public space of universal communication.

Public wealth: The free access to the tools for information retrieval is already considered a public necessity and universal right. All WWW users navigate the infosphere without having paid for a browser. It is unlikely that anyone will ever want to pay for a browser. All pay-to-play commercial WWW alternatives and predecessors, of which there were only a few, have failed. Of the 5 major browsers (Explorer, Firefox, Chrome, Safari, and Opera) only 2 are closed source: Explorer and Opera. Chrome, Safari and Firefox, including the various derivatives, are free and open source software (FLOSS). Taken together, this means that according to statistics of the W3C as of 2011

between 40-70 percent of all WWW users browse the web with a free software browser.⁴ The percentage of FLOSS browser usage is ever increasing.

Read,write & publish: The free access to the retrieval and consumption of information must be equally matched by free access to tools for writing and publishing in this system. Any and all software that contributes to or constructs an ideal public and universal system of communication must be free software and must provide tools to not only read the system formats, but also write and publish. This is currently lacking in the WWW.

The software is the standard: The traditional standardization mechanisms of the WWW are unnecessary in a FLOSS environment. Creation of new software APIs and infrastructure through the proven methods of free software development (direct action through weak cooperation) is an improvement over the current methods of pseudo-standardization via direction by industry. Up until HTML5, all standards for the WWW have been implemented retroactively. HTML5 has been in progress since 2004, yet we are only seeing partial implementations of the APIs now. The development mentality of HTML5 leaves no open way to implement other APIs that are deemed insignificant or unworthy of support. The protocols, APIs and chosen for-

⁴See http://www.w3schools.com/browsers/browsers_stats.asp Internet Explorer is steadily declining while Chrome and Firefox steadily increase their user base

mats for the WWW are all created on the assumption that there should be a fair playing ground for competing (for-profit) entities to implement browser technologies. The ideology for this is focused on industry standards to which all third parties should comply. Following the development strategy of FLOSS which allows for new technologies to be implemented in a transparent fashion by any and all, and later standardized retroactively, these so-called preemptive standards are too brittle, too mono-linguistic, and logistically unnecessary.

Literacy: The WWW system and format as it was and as it is now is too complicated for normal computer-literate users to write and publish in the system. Users currently depend on centralized commercial services to publish online. There is both a lack of literacy among users and a lack of technical capabilities in the system. A more technically elegant system would provide both basic methods of writing and publishing as well as highly sophisticated and complicated methods.

1.1 What is the World Wide Web, thing and metaphor

It would be incomplete to start a discussion and articulation of the problems of the WWW without actually having an idea of what it is. The WWW is both

a thing that people use in their daily lives - although somewhat indefinable - and metaphor. The most important aspect to consider is that contrary to conventional understanding the WWW is not the Internet.

The Internet is generally classified as a network of networks. It links various disparate computer networks into one functional and congruent entity, allowing many kinds of devices and people from around the globe to intercommunicate. It achieves this by way of a flexible and efficient set of protocols known as the Internet Protocol Suite. The Internet Protocol Suite was designed mostly by paid research engineers who were altruistically interested in the idea of a royalty-free non-proprietary global communication infrastructure. It has taken many years for the Internet Protocol Suite to evolve into the stable form that we use today.

The WWW, like email and File Transfer Protocol (FTP), is a set of protocols and applications that run on top of the Internet. Building and innovating on a large and extensive tradition of hypertext research, Tim Berners-Lee conceived of the WWW as an academic project for the purpose of information management. It subsequently grew by the various users and participants who put up their own web servers and pages. Generally speaking, the WWW consists of a myriad of server and client applications communicating only via the HyperText Transfer Protocol (HTTP). The client - also known as a browser - requests content from the server and the server sends this content to the browser. The WWW content is

formatted mostly using a combination of HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and Javascript.⁵ The evolution of the transfer protocols (HTTP) and file format (HTML/CSS/Javascript) has taken place over many years. While HTTP has changed quietly and slowly over the years, the HTML/CSS/Javascript format has only changed through much dispute as browser vendors vied to gain an advantage in market power by growing a larger user base. This conflict is often referred to as the *Browser Wars*.

As a metaphor, the World Wide Web is more than just a seemingly neutral and topological arrangement of servers and clients. It has become the clairvoyant and cryptic face of global telecommunications infrastructure. It is the functional, symbolic, and discursive valve through which we trade the functional, symbolic, and discursive objects of our communication.

The web metaphor itself is animated and fluctuating, bringing duplicitous visions of liberating connectivity and shackled entrapment. The question of the web still remains: are we⁶ the spider or are we the fly in this arrangement? Are users at the navigators helm, able to control an informational landscape, picking and choosing from it as they want and need? Or, do informational technologies exhibit autonomous behavior and have influence on its users? From the

⁵The international standard for Javascript is sometimes known as EcmaScript. Both Javascript and Adobe's Actionscript are examples of EcmaScript derivatives.

⁶..as individuals or as a collective

amazingly vast cultural literature on the subject, sometimes described as “vapor theory”[29][66][42], we can see the oscillatory nature of this metaphor. The WWW is defined as both real and magic, utopian and dystopia, fulfilling and disappointing, freeing and controlling, deterministic and neutrally anarchic. There is, however, another more superficial⁷, and perhaps equally debilitating, metaphor - the frame.⁸

As an alternative metaphor, the frame allows us to look at the surface and portal provided by the web from formal, compositional, and content-oriented per-

⁷I mean this in the literal sense of surface, not shallow; although, we may discover that one directly points to the other; flat and prostrate.

⁸A frame is a specific, yet ambiguous, kind of non-dimensional flat-space. Bicycles, beds, eyeglasses, pictures, and doors all have frames. In architecture, a frame is a structural support system. In art, a frame is often used to hang artworks, such as painting and photography, on a wall. This signifies to the viewers what is *in* the art and what is not. In computer speak, a frame could be a network packet frame or an audio frame. It could be the wireframe view of a 3D rendering.

A frame is also a kind of window on the future; a frame of view. As history unfolds in the present and overlaps with some future trajectory, a frame of reference from things in the past shapes a frame of mind for future discovery and enlightenment.

A frame can be a framework or frameset. These are strangely not plural aggregates of multiple frames, but an overarching frame built from specific non-frame components. There are conceptual frameworks, legal frameworks, and software frameworks to name just a few. A software framework dictates not only the APIs that are used but also the overall flow of control in the program. These frameworks may have default behaviour, suggested or mandated operations, more-than-suggested best practices, as well as built in extensibility.

Time frames also have special properties. Eisenstein’s and Kuleshov’s montage theory demonstrates how emotional and psychological ideas can be framed on screen by the successive collision of film segments, one after the other. Edward Branigan further explains at least 15 other ways frames have significance in film theory, where frames of time are shown within frames on screen.[16]

A frameup is special kind of abstract frame. It pins the guilt of a crime on an innocent bystander or victim.

All in all it is a word that simultaneously denotes both encapsulation and support. When I say *Re-Framing the WWW*, it is my hope that the reader will consider the many radial meanings of this word.

spectives. It allows us to understand the WWW as a window on an informational landscape. Absurdly, it also situates the discussion within an equally vast, but surrogate groundwork. As the web browser increasingly becomes the most important window on designed and de-signified information, it also becomes a frame on the conscious world of communicative beings. Whoever or whatever controls this metaphoric construct, including all the encapsulated and interlinked forms by which it is concealed - if there is such an identifiable entity - governs behaviour. This windowing device decides what, how, and in which manner content can be exchanged among individuals on computer networks. It makes things visible and perceptible on the surface, but also renders unknown the underlying layers by way of protocol and procedural programming. As the gateway to interlinked computer-mediated consciousness, it is the ultimate stage for info-marketing and neurological theater at the epicenter of social intercourse. At this prosaic level of the frame, it becomes less relevant if the underlying structures are web-like, rhizomatic, or routed through a centralist system. The frame is all that we see and understand. It is the portal and aperture. It is the lens and facade. It is the reflective mirror of our own collectivity. It is also the enclosing construct - perhaps somewhat like an Elizabethan Collar - that delivers our attention to others sitting at the other side(s) of this portal. It serves as a threshold between infor-

mational and networked societies. Furthermore, as a critical device, the frame is both inclusive and exclusive, elastic and dimensional.

Using the frame as a constructive delineation between soft and fluid computational forms, the research focus for this dissertation is placed between the outer frame of the web browser and the inner frame of the desktop operating system. In essence, this dissertations attempts to define the negative space of the world wide web - all the potential functionality that is to some extent implicitly expected of the WWW as a universal communication platform, but that does not YET exist in web formats. As the WWW changes from a singular file format (HTML) on top of a communications infrastructure (the Internet) into an application environment (operating system) integrated within a communication infrastructure (a new Internet with cell phones, gps, and off-the-shelf ubiquitous wireless technologies), are there better suitable methods to activate this space?

This surrogate space that I here imagine and prototype can be seen as a parallel environment tightly enmeshed in the already existing and evolving WWW. As such, my aims are not to compete directly with the existing protocols and ways of doing things on the web. Instead, my aim is to build confluent experiments in code that examine what possible aesthetic, discursive, and vernacular spaces could be cultivated other than what already exists.

Chapter 2

Background

In this background section I briefly describe what I feel are relevant histories, guiding principles, and current plausible trajectories of the WWW. Because it is impossible to isolate the situation of the WWW from the complex historical intermingling of computing technologies, software development, liberal markets, and liberal democracy, I will also need to briefly outline the movement of and connection between socio-political, aesthetic, and technical agendas (where and when they exist). I will discuss the principles and modality of packet-switched networks, computer operating systems, free software, and the evolution of browsing and the HyperText Markup Language (HTML). A running theme is also set on the kinds of openness and freedom that are defined in protocological networks.

I follow four main trajectories:¹

¹A fifth underlying trajectory that I do not discuss is the development of the capabilities of computers, including their display and interaction devices such as the keyboard, mouse, gps, and multi-touch.

- the development of the internet
- the development of free software
- the development of hypertext
- the development of browsers and standards

The four meet at a very interesting intersection from which I draw a number of important conclusions. First, there is a general shift away from the text oriented WWW into video, multimedia, and general purpose computing. Historically speaking, there has also been a general shift from centralized and hierarchical media structures such as television and radio to distributed and decentralized media such as the early WWW. Now, the centralizing forces in Web 2.0 are bringing the WWW back to hierarchical structures. Additionally, because of the layered design of the Internet protocol stack, the middle layer of the protocol (the IP layer) is next-to-impossible to change. However, the design is made such that top-most and bottom-most layers can come and go. See Sect. 2.1 below. This provides for the possibility and plausibility of a new WWW. Additionally, because of the way browsers are used and how their value behaves in the market, they will always be free of cost to consumers. They can no longer be considered *products* or *property* in the traditional sense. Also, we can attribute the popularity and scalability of the WWW to three main reasons: it consisted of a source-viewable text format,

was provided without cost and royalty-free, and lacked centralizing back-linking. Following this, the network logic of free software in a civic informational environment also suggests a few things: that all future browsers will also provide their source code to users, and that to a certain degree, a more imperfect, loose, and open design is favorable over well-engineered or stringent ones. Additionally, the development of the WWW shows that most so-called standards in the WWW were implemented *post hoc*, after implementation and accepted use. This leads one to question the idea of standards in the WWW in general.

2.1 Internet

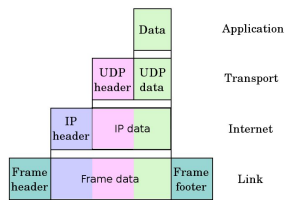


Figure 2.1: packet header encapsulation.

A brief analysis of the main components of the Internet will prove insightful in studying the choices made in constructing a scalable alternative architecture for the WWW and other applications. The developments I discuss here demonstrate two things. First, the design of the Internet Protocol is good, but not perfect.

We should not assume that any one protocol design or design strategy is best. Secondly and most importantly, the layering of the protocol is built in such a

manner that the middle layer becomes difficult to change. However, it provides a flexibility that makes it possible to change the applications that run on top of internet, such as the WWW. I now turn to the Internet communication protocol which is based on three main guiding principles: packet switching, encapsulation and layering of protocols, and end to end design.[18]

Packet switching and circuit switching are the two main modes of forming a network infrastructure.[38] Circuit switching creates a link in the network by reserving resources along a network path between two nodes and dedicating bandwidth for them. As long as there is a connection, the bandwidth remains the same (even if the bandwidth is not being used.) On the contrary, packet switching is a method by which a message is broken up into packets (or datagrams) and then sent out into the network without dedicating a fixed path between the two nodes and without reservation of bandwidth. If one route between the two nodes is congested or blocked, packets can be re-routed to circumvent. Packet switching can thus make no guarantee to deliver packets in a timely manner as net congestion and packet routing may cause losses and redirection of packets. Generally, we can think of the telephone as a circuit switching network and the Internet as a packet switching network.² Packet switching allows the net to be faster, distributed,

²Today, both the telephone and Internet are a mixture of packet and circuit switching. ATM is a circuit based. Skype and other Internet telephony are, of course, packet based.

scalable, more flexible, and more manageable. It also causes messages between two nodes to be segmented into short pieces for transmission.³

Packet switching is only important to the argument of this dissertation because it necessitates the encapsulation and layering of protocols and sets up the main Internet designing principle known as the *end to end* principle.

Since packet-switched networks need to segment a message into smaller parts that can make their way from one node to another, potentially through various routing schemes and over various physical network types, they must have information attached to them so that the destination is addressable at the various hubs along the way in the network. Essentially, this extra information comes in the form of a header

(and footer) at each hop and provides a means of abstracting the various protocols and services that might exist between a node on one network and a node on another. The header is extra memory space allotted on the front (or tail in the case of

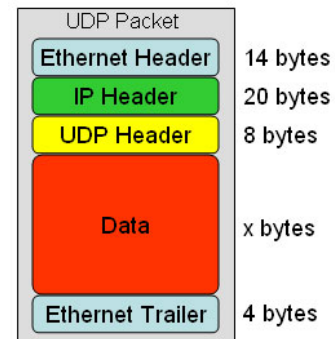


Figure 2.2: packet header encapsulation 2.

³Additionally, unlike circuit networks such as cable television, but more like ethereal broadcast networks, packet switched networks make it difficult to charge for *content* that may come from any source. France’s early tele-service network, Minitel, was able to charge for content by the minute in the 1980s over a packet-switch network because the terminals that ran on the system were *dumb* and could only connect to the central service by (a then public) France Telecom. Today, Minitel has been superseded by the web.

footer) of the data with information necessary for the intermediary gateways and routers between hosts and networks. As a packet goes through various networks it accumulates and changes header information. Each type of passageway can only edit its section of the header. In this manner, we can speak of this encapsulation as a layered or tiered system of communication. The Internet Protocol Suite (commonly known as TCP/IP) which defines this infrastructure of encapsulation and layering can be visualized as in Fig. 2.2 and Fig. 2.1.

This abstraction of services that forms the main infrastructure of the Internet is a derivation of the *end to end* principle of network design and distributed systems.[52] The *end to end* principle states that intelligence of a communications system should occur as close as possible, and wherever possible, at the end-points of the system. Computers that speak to each other are the intelligent end terminals. As one end node sends a signal or bitstream to another it passes through various infrastructure along the way such as routers, proxies, and gateways that act as translation units. The intermediary hubs and routers that form the network in between are simple dumb transceivers. They pass datagrams/packets from one kind of network to another, and don't know anything about what or why the end terminals are communicating. They only know how to encapsulate and hand the data over from one network system to another. Pushing the intelligence to the

end nodes in this way provides more flexibility and robustness to the system. This necessitates a layering of protocols.

These layers are often drawn as an hourglass to demonstrate their flexibility. As we can see in Fig. 2.3, the abstraction of protocol layers makes it possible to add and change components on the higher and lower layers of the stack. New protocols on the application layer, like HTTP and RTP (real-time transport protocol), can come into being while others, like Gopher and Archie, fade away. The very same thing goes for the lower layer as new physical

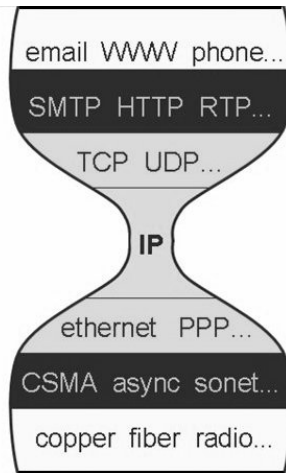


Figure 2.3: tcp-ip hourglass figure.

means for connectivity, such as fiber optic or new wireless encryption, have become possible. The middle layers of TCP (transmission control protocol) and IP (internet protocol)⁴ are almost impossible to change because practically all of the intermediate routers and gateways are now hardwired to run the middle layer protocols. To change this layer would mean a massive overhaul of the base internet infrastructure. What this boils down to is the following. While the middle IP and TCP protocols are difficult to change, it is not impossible to add or change pro-

⁴TCP and IP were at one time a single protocol, but are now split into two.

protocols at the upper and lower levels. In fact, in 2002, Katabi, Handley and Rohra developed a new protocol called XCP (explicit control protocol) that outperforms TCP as bandwidth use increases.[34] However, because it requires some features to be computed in the network as opposed to just the end-terminals, it has proven extremely difficult to implement. Another TCP compatible protocol, called PCP (phantom circuit protocol), also makes performance improvements on TCP, but has yet to be implemented.[2] For the purpose of this dissertation where focus is on the application and display layer (the browser and desktop frame), the only change that is necessary is in (mental, social, political, ideological, etc.) software that focuses all development on a single universal system of communication.

2.2 Free Software

I provide here a brief introduction to software, and more specifically, to Free Libre and Open Source Software (FLOSS). As a methodology of software development, tightly intertwined with cultural systems of production, FLOSS provides a vast context in which to place much of the research related to this dissertation. This context is defined by three main things. First, browser vendors not only provide their applications to users at no cost, but are also increasingly releasing their software to the public including the source code. Secondly, FLOSS meth-

ods of software production are increasingly becoming valid for both industry and private use. Finally, the public communication space that exists in the WWW necessitates a public software that is both *open* and *libre* .

In order to provide a background as to what this is and why it is important, it is first necessary to know what software is and how it can be open, free/libre, or closed.

Software can be abstractly and loosely defined as a set of instructions that perform some task. Today, software is roughly divisible into two separate parts: source code and byte code.⁵ The source code is the human-readable written text, usually in an English-based artificial language, that is then translated (i.e. compiled) into byte code that the computer can read and execute. Free and open source software is focused on protecting the availability of the readable source code in order to build shared digital resources and cooperative environments based on non-scarce digital resources and public wealth.

FLOSS is the moniker for a techno-political movement that seeks to define and defend user rights in the development and use of software. FLOSS is any software where the distribution of the original source code is given to users along with the legal right to share, modify, improve, and redistribute that code. It defines free

⁵It is a bit more complicated than that as byte code is one compilation layer above machine code. Also, with interpreted scripting languages, virtual machines and inter-language code compilation allows even source code to be abstracted to a higher level.

software in terms of liberty, not price⁶, and is protected by a variety of licenses, the most important of which is the General Public License (GPL). Under the GPL and other compatible licenses, the redistribution of the code is permitted under one important condition: that the modified code is also available in source format under the same GPL license. This requirement is known as copyleft and earns its legal power from the use of copyright to accomplish the opposite of its usual purpose; that is, copyleft in the GPL guarantees the protection of user rights in generative iterations of software development.

FLOSS is also not the same thing as open source software. The two have technical and ideological differences and should not be confused, even though the idea of openness and freeness are inarguably separate but related. Each have multiple meanings and even more connotative and denotative interpretations. Openness is related to the restriction of movement and the state of passageways. Freeness, at least in the English language, generally has two connotative meanings. On one hand, it means without cost, or gratis. On the other hand, it means liberty. It is in the latter sense of the word that free software gets its name. In contrast to the ideology of liberty that drives the FLOSS movement, the open source movement is a business-oriented development methodology that focuses mostly on the utility

⁶Think *libre* as opposed to *gratis*. While there is enough ambiguity already in the term *libre*, the English language currently makes it worse by conflating cost (free commodity) with freedom (liberty).

and efficiency of software.[59] Eric Raymond and others coined the term⁷ in 1998 as a kind of marketing strategy for free software.[51] Taking the word *free* out of the name places more emphasis on the development strategy and makes the term more business-friendly at the expense of loosing its association with liberties. All free software is also open source. Open source software is not necessarily free software. Unlike other open source licenses, the GPL effectively ensures that a specific software remains in the public sector as an accessible community resource.⁸

The history of the FLOSS movement, however, starts before computer code was ever proprietary. In the 1950s and 60s operating systems and application software were widely distributed and maintained by communities of users from academia and industry in an open sharing environment. Source code was always given with hardware, often without a license. It wasn't until the late 70s and early 80s that software companies formed and turned a once open and sharing software exchange into a marketable commodity. In 1983, motivated by his frustration with a closed-source printing software, Richard Stallman formed the GNU Project that seeks to write a complete operating system free from constraints on the use

⁷According to <http://www.opensource.org/history> : “The 'open source' label was invented at a strategy session held on February 3rd, 1998 in Palo Alto, California. The people present included Todd Anderson, Chris Peterson (of the Foresight Institute), John "maddog" Hall and Larry Augustin (both of Linux International), Sam Ockman (of the Silicon Valley Linux User's Group), Michael Tiemann, and Eric Raymond.”

⁸It is important to note that FLOSS software is not just source code in the public domain (as it is in BSD) where a company could come close the source for use in their industry (as Apple did with the use of BSD in their OSX product). GPL means the source code is available plus copyleft restrictions.

of its source code. In 1991, roughly the same year as the birth of the WWW, Linus Torvalds released an open-source alternative to the Unix operating system called Linux. Together, GNU/Linux is a functioning FLOSS operating system with thousands of high-quality desktop and mainframe applications.⁹ Seen as one thing altogether, it forms the largest collaborative software project to date.

Having already been ported to virtually all hardware systems from large clustered cloud computing¹⁰ to cell phones¹¹ to embedded systems¹², GNU/Linux has proven itself to be a tested and validated method of software development for a variety of computing groups. It has proven effective not only for hobbyists and system administrators, but also for desktops¹³, Hollywood¹⁴, education¹⁵, government¹⁶, and crucial industrial operations.¹⁷

FLOSS's production of publicly available and usable software has also been validated as a sound economic strategy. A 2006 report by the European Commission

⁹I am writing this dissertation in a free software typesetting system called TeX created by Donald Knuth in the 1970s, roughly the same time as Richard Stallman's GNU movement and Berkeley's BSD operating system. This is particularly noteworthy in this context for a number of reasons. There was more than one instigator of the openness of software. TeX has been around for a long long time and commands a large community of developers and users. There are unfortunately few statistics on how many PhD dissertations were written in some variant of TeX since the late 1970s. I can only speculate that it is in the hundreds of thousands.

¹⁰See IBM's tutorial on how to build a high-performance linux cluster <http://www.ibm.com/developerworks/systems/library/es-linuxclusterintro/>

¹¹See Google's android phone for an example. <http://code.google.com/android/>

¹²<http://www.gumstix.com/>

¹³<http://www.ubuntu.com>

¹⁴<http://www.linuxmovies.org/>

¹⁵<http://laptop.org/en/>

¹⁶<http://arstechnica.com/open-source/news/2009/03/french-police-saves-millions-of-euros-by-adopting-ubuntu.ars>

¹⁷http://www.linux.org/info/linux_industry.html

on the economic impact of open source software found that in “almost all” cases long-term business costs associated with software could be reduced by switching from proprietary software to free and open source software[31, p.100] and save the software industry over 36% in software R&D investment[31, p.11]. In a more recent analysis from 2008, research firm Gartner believes that 80 percent of all commercial software applications will include open-source components by 2012¹⁸, stating that they “... provide significant opportunities for vendors and users to lower their total cost of ownership and increase returns on investment.”[30]

Free and open systems are not only technically functional and economically sound, but are also well integrated in social dynamics and dialogue. “Because free and open source software opens up the process of writing software in certain ways its [sic] also opens up the process of talking and thinking about it.”[27] There is no dialogue in closed systems. Closed-source systems provide black-boxed software objects where the instructional language, grammar, and syntax that run the system are hidden inside. True, you can use these closed systems, but you cannot discuss them as software objects, much less modify them to suit

¹⁸Most likely they are referring to the use of permissive open source licenses that have no copyleft clause such as the BSD license. Speaking of these permissive licences, an ars technica blog writer has this to say: “Some open-source implementations of commonly-used technologies are so widely adopted in commercial software applications that they have nearly become de facto industry standards. A few examples include the zlib data compression library, the OpenSSL secure sockets layer library, and the Boost C++ libraries. These open-source technologies can be found in mainstream commercial software applications developed by a wide range of well-known companies, like Adobe, Real Networks, McAfee, and many others.”[48]

individual, irregular, or atypical needs. These closed-source systems are built only to be used, and conversely to produce users. FLOSS, on the other hand, encourages - sometimes forces - a user to study and modify the coded structure of the software in its written human-readable form. In so doing, a convivial methodology of development is born based on communal interests and shared code.

This open, dialogic, and convivial manner of producing functional cultural artifacts has drawn from and sparked enthusiasm in diverse fields outside of computer systems engineering. In everything from free culture¹⁹ and creative commons²⁰ to open source democracy²¹ to open source money²² to open source groceries²³ and beer²⁴, we can see how cooperative initiatives are drawing from the success of the FLOSS movement to create collective wealth using shared strategies of openness. This collective sharing is exacerbated by the informational presence

¹⁹Free culture is generally seen as a set of movements with different agendas to promote the freedom to modify and distribute creative and culturally significant works such as music, videos, and literature. On a Nov. 2009 posting to the iDC mailing list reporting the first Free Culture Forum in Barcelona this year Micheal Bauwens defines it as "... the bottom up creation of multiple creative expressions by any member of the population, enabled through the global possibilities for interconnection that have been enabled by the widespread, though still very insufficient, availability of the internet and the tools for cultural production and distribution." <https://lists.thing.net/pipermail/idc/2009-November/004051.html>

²⁰Creative Commons are a set of licenses for granting users of creative works various degrees of permissions: <http://creativecommons.org/>

²¹<http://rebooting.personaldemocracy.com/about> and <http://www.gutenberg.org/files/10753/10753.txt>

²²<http://www.transaction.net/money/community/> and <http://openmoney.info/>

²³<http://www.openproduce.org/> and <http://opensourcefood.com/>

²⁴<http://www.opensourcebeerproject.com/>, <http://www.freebeer.org/blog/> are two examples of beer based projects that carry the rhetoric of free software. Tom Marioni's *Free Beer* conceptual artwork, however, precedes this on a number of levels.

and communicative network logic of the WWW. All of this demonstrates that the inclusive mechanisms of FLOSS have proven to work well enough to build stable, extendible, working, and (ab)usable software applications in all areas of computing.

The FLOSS methodology is of course not the only way to develop and release software. The system of FLOSS is also not without problems. One of the major problems is its lack of financial justice for developers. FLOSS as a whole has only a few viable business solutions, all of which are industry related. Most of them are based on servicing support to users. Despite the cultural significance of software, free software still lacks any real and direct social support system. Unlike the system of art funding in Canada, Europe, and Australia; unlike the systems of funding science in the United States; and unlike the funding of schools, roads and bridges by virtually every society, the development of free software does not receive any significant amount of publicly-sourced funding.

Still, despite the lack of direct funding, FLOSS has thrived in many places where most thought it couldn't. Most significantly, FLOSS has thrived in the browser market. It is also here where FLOSS makes the most sense - in a universal public communication environment where an open basis of communication is prerequisite for collective acceptance and communal exchange. Of the five major browsers on the market today in 2011 (Internet Explorer, Mozilla Firefox, Safari,

Chrome, and Opera), all of them are provided without cost. Three of them are not only open source software, but FLOSS, and make up somewhere between 41 and 73 percent of the market share.²⁵

2.3 Browser Homology

The story of the web, which for this dissertation is essentially the convoluted story of networked file browsing, is vast and multi-dimensional. Intertwined with the development of human-computer interfaces, network development, graphical displays, file formats, compression, and the Domain Name System (DNS), the browser has many antecedents, relatives, and operative parts. I illustrate a few of the historical developments here and draw a few conclusions. First, the WWW browser grew out of previous centralized hypertext environments that were in many ways superior to the initial WWW. Secondly, the WWW performed better than its predecessors in the real world because it consisted of a source-viewable text format, was provided without cost and royalty-free, and lacked centralizing back-linking. Thirdly, the concept of a WWW browser was not embodied by any particular software.

²⁵Percentages taken from 3 websites in June 2011. http://www.statowl.com/web_browser_market_share.php http://www.w3schools.com/browsers/browsers_stats.asp
<http://www.w3counter.com/globalstats.php> <http://www.netmarketshare.com/browser-market-share.aspx?spider=1&qprid=0>

On a historical level, it would be incomplete not to mention Vannevar Bush, Norbert Wiener, J.C.R Licklider, Douglas Englebart, and Ted Nelson for their respective work on the Memex, cybernetics, man-computer symbiosis, the oN-Line System, and Xanadu. Each served as antecedents to the WWW in various capacities.²⁶ Although these early developments only presented ideas and prototypes and did not produce a globally interlinked communications space, we can start to gauge from them the coming shape and scope of the WWW. More could certainly be said about the utopian and imaginary futures these technological developments have created.²⁷ I will, however, only mention a few of the developments briefly and then move on to a more recent genealogy of the web.

Particularly noteworthy in the late 1960s are Englebart's oN-Line System, Andy van Dam's HES²⁸/FRESS²⁹, Carnegie-Mellon's ZOG, and Andrew Lippman Aspen Movie Map. The oN-Line System that Engelbart developed at the Stanford

²⁶Any contemporary media art or web history textbook will have the full texts if not references to each project. Please refer to *Multimedia from Wagner to Virtual Reality*[47] and *The New Media Reader*[70] for examples

²⁷Richard Barbrook's *Imaginary Futures: From Thinking Machines to the Global Village* traces the technological and ideological frameworks of the *future* as produced and developed through contemporary history starting with the turn of the century, going through the Macy conferences, and then up through the Cold War, McLuhan, and the on-coming of the Internet. A large portion of the text is spent on defining the struggle for control over the imaginary future. In that struggle, many ideologies collide and coagulate. Cybernetics, technological determinism, cold war left and right, capitalism, big business, communism, direct democracy all form the geo-scattered imaginary future.[6] <http://www.imaginaryfutures.net/>. Also see Adam Curtis's *All Watched Over By Machines of Loving Grace* http://www.bbc.co.uk/blogs/adamcurtis/2011/05/all_watched_over_by_machines_o.html.

²⁸Hypertext Editing System

²⁹File Retrieval and Editing System

Research institute in 1968 gathered over 100,000 inter-referenced papers, reports, and memos into a clickable collaboration system.[1] It combines the use of the computer mouse (Engelbart's invention) with dynamically linked documents and a shared audio-video feed to form a responsive and shared info-media environment that was truly one of a kind.³⁰ It predates the WWW as well as augmented reality. Andy van Dam worked with Ted Nelson at Brown University to construct the HES (Hypertext Editing System) in 1967. It was a pioneering hypertext system that organized data into links and branching text. Its main emphasis was on menus, labels, and text formatting. It was then superseded by FRESS.³¹ FRESS extended HES in the 1980s by adding bidirectional linking and rudimentary transclusion - the ability to include a part of another text by reference. FRESS was used for instructional computing, type setting, word-processing, but also for teaching poetry.³² ZOG was developed in 1972 at Carnegie-Mellon University. Instead of pages, ZOG had frames with titles, descriptions, select menus, and a set of ZOG commands which would lead to other frames. Users would modify the frames using ZOG commands to build a collaborative knowledge base. A PERQ workstation implementation of ZOG was used on the nuclear-powered aircraft carrier USS Carl Vinson.³³ The Aspen Movie Map was an ambitious project in 1978 that

³⁰Please see <http://sloan.stanford.edu/MouseSite/1968Demo.html> for a description and video clips.

³¹FRESS comes from the German fressen, which means to eat like an animal.

³²<http://www.derose.net/steve/writings/whitepapers/fress.html>

³³http://ei.cs.vt.edu/book/chap1/htx_hist.html

sought to virtually map the entire town of Aspen, Colorado into a navigable hypermedia environment using a correlated database of street layout with film on laserdisc. Interaction was controlled through a dynamically-generated menu overlaid on top of the video image.³⁴ We see in these systems the formal qualities of current day browser usage - everything from clickable buttons, to editable wikis, to geo-referenced street viewing. Except for extra bells and whistles that come from high-performance computers, the only thing that sets these early prototypes apart from today is the concentration on utopian information exchange and an isolation from outside commercial agendas. Let's now turn to the direct start of today's hyper navigable system.

The direct lineage of software based browsing of distributed electronic media (mostly text), begins more or less in 1980 at CERN with Tim Berners-Lee's program Enquire-Within-Upon-Everything which allows links between arbitrary document nodes to be made. By that time, the idea of linked documents was becoming popular in academic crowds and one can find a number of varying terms and vocabularies for describing the linking methods between digital objects. 1987 saw the very first ACM Conference on Hypertext and Hypermedia. It was given again in 1989 and continues almost yearly since then. In the 1987 proceedings, you can already read about early web predecessors and the primordial ideas for

³⁴A video demonstration can be found here <http://www.youtube.com/watch?v=Hf6LkqgXPMU>

Chapter 2. Background

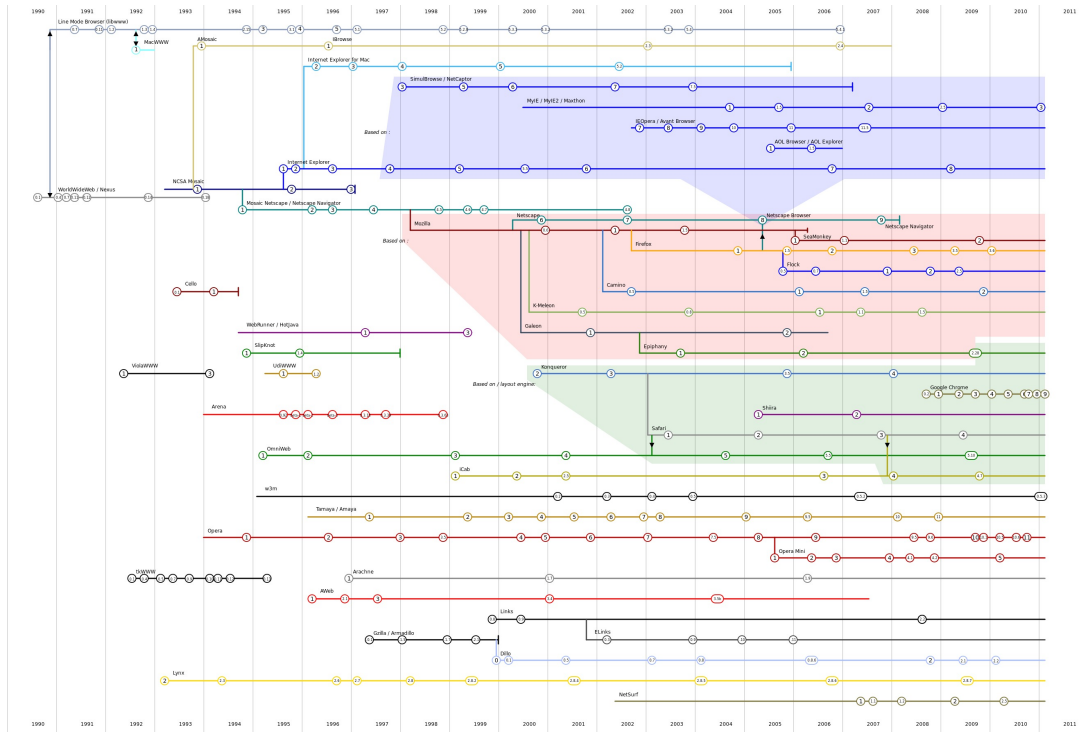


Figure 2.4: Timeline of browser genealogy. source: Wikimedia Commons[24]

things like wikis and online dictionaries and encyclopedias. Of course, a major theme in the proceedings is the use of hypertext for forked creative writing practices. As video and multimedia were not easily feasible in the 1980s on normal computers, much of the rhetoric revolves around text and information retrieval. Still, a predictive disposition and enthusiasm of the coming mixed-hypermedia was already alive and discernible.

More overlapping development arrives in the late 1980s. In 1987, Apple provided Bill Atkinson's HyperCard system with every Macintosh for no extra cost. In 1989, Ben Shneiderman and Greg Kearsley published *Hypertext Hands-on!*, a

commercially available electronic book with highlighted textual links[53]. In the same year that *Hypertext Hands-on!* was published, Berners-Lee drafted *Information Management: A Proposal* in which he describes better management methods with the use of hypertext at CERN.[10] His ideological version of hyperlinkage is contextualized here by the info-management problem of large evolving organizations. In 1990, a patent application was placed for an electronic book called *PageLink* that could download and navigate texts from other computers.[20] That same year, Tim Berners-Lee began working on the first network browser, simply called WorldWideWeb.³⁵ In 1991, WorldWideWeb was accepted only as a poster presentation at the Hypertext '91 Conference in San Antonio, Texas.[14] By the end of 1991, the web was off the ground and crawling.

There are a number of things that provided the right mixture of technology and freedom that allowed the WWW to take off like it did. Berners-Lee made a hypertext system just like those before him. Only he made a few very small, but significant, adjustments over his predecessors. His first innovation was to forgo a clean hypertext system with menus and proper back links. Instead of focusing on a well-structured and contained textual data with intricate linking mechanisms like its predecessors had done before, WorldWideWeb had only uni-directional links.

³⁵WorldWideWeb (without spaces) was the name of the browser created by Tim Berners-Lee. The World Wide Web (with spaces), on the other hand, is a generic name for the global communication space.



Figure 2.5: WWW poster at the 1991 Hypertext Conference in San Antonio, Texas. Source: <http://www.david-praterville.com/homework/hw01-html-history/01html-history.html> (Last accessed August 2011)

It was left to the users of the WWW to build their own navigational structures. His second innovation was to construct a protocol that allowed disparate hypertextual systems to speak to one another. This extended the reach of a hyperlinked document from an isolated computer to many over a simple pull-based and page-based protocol. This protocol would later become HTTP.[11] His third innovation was to provide his invention royalty free. Speaking about his ability to cash in on his development for the web, Tim Berners-Lee says the following.

It was simply that had the technology been proprietary, and in my total control, it would probably not have taken off. The decision to make the web an open system was necessary for it to be universal. You can't propose that something be a universal space and at the same time keep control of it.[13]

While he is speaking of the copyright and patenting of his invention (if we could believe that his invention was indeed isolated from other earlier systems), the same also applies to the internal format of his web architecture.

There is one other element that led to the popularity and growth of the web, although it was probably more of a side-effect than an intended plan - the WWW format was simple, text based, and open source. At the time of the WWW development, many computer screens lacked decent imaging technology. The resolution was small and the colors limited. Clear text was a natural way to pass information along in the Internet. A binary format would have offered some advantages and disadvantages over text, but Berners-Lee opted for a very simple text based markup format. The main disadvantage of a binary format would be the extra dependence on some sort of external program to make the files (as is used by Adobe to make the Flash format). This would have greatly limited the ability of users to write the WWW format. The original WWW format was also simple enough for many to write without having too much technical knowledge. Most importantly however, any page's source text could be viewed, studied, copied, and modified³⁶. The technical makeup of the format itself achieves the same socially networked effect as FLOSS, but without the license or the outspoken ideology.

³⁶See Jodi's <http://www.wwwwww.jodi.org/> and do *view source* on the page.

Between 1991 and 1993, momentum for the WWW increased and by 1993 there were already servers outside of Europe with a few different browsers in operation: Midas, Samba, Viola, and Erwise. Midas was an X-Windows browser for the Motif platform. Samba, also known as MacWWW, was a browser developed at CERN for the Mac platform. Unlike other browsers, it opened each link in a new window. Viola was released in 1992 and was the first web browser with inline graphics, scripting, tables, stylesheets, and plugins.³⁷ Berners-Lee's WorldWideWeb browser was unable to display graphics inline because of underlying problems with the NeXT system for which it was developed. In many ways, Viola was well ahead of its time with document embedding and style sheet capabilities, both of which didn't make it into popular browsers until years later. Erwise was also released in 1992 for Unix computers running X-Windows. It was the combined masters project of four students at the Helsinki University of Technology and along with Viola claims to be the first graphical browser. In 1993, NCSA released the first alpha version of the Mosaic browser which eventually becomes the commercially available Netscape Browser. That same year CERN agrees to allow anyone to use the web protocol and code royalty-free.[14]

It wasn't until 1995 that Microsoft Corporation released their Internet Explorer, which to no surprise would quickly become the most popular and widely

³⁷For pictures, see here: <http://www.viola.org/>

used browser on the web. Up until that point, Microsoft had viewed the web with curiosity and scepticism. Now, they were not only ready to enter the market, but to integrate their Internet Explorer directly into the windows operating system, and with no extra cost to the user. The file browser was the same application as the internet browser. This direct integration into the most widely used desktop operating system basically guaranteed its spot on the top of the browser market. Subsequently, in 1998, the United States Department of Justice and 20 U.S. states sued Microsoft for allegedly eliminating rival competition and creating a monopoly. Additionally, this integration aesthetically and conceptually folds the HCI³⁸ oriented file browser with clickable icons onto the WWW browser and molds them into a seamless computational window. The same frame is used to navigate the internally clean world of the desktop (file browser and selector) and the externally dirty world of the WWW. This seamlessness creates a power struggle that sets a number of things in motion, slowly forcing Netscape - the top browser at the time - to close its commercial operations. At the same time, it opens the door for a competitive free software browser to come about. In October 1998, the Mozilla foundation was formed based on the source code of the Netscape suite. The release of this source code would eventually lead to the development of

³⁸Human Computer Interaction

the Mozilla Firefox browser, the first FLOSS browser to gain a competitive edge in market share.

Two other interesting things happen in 1995; the invention of the wiki and Hyper-G. Ward Cunningham's WikiWikiWeb, the very first Wiki available, added direct and easy editing of a WWW page inside of a browser. Its main innovation was that it gave users a direct method of publishing content on the WWW. As a technology, a Wiki is a language overlay on top of HTML. It uses an even simpler syntax than the WWW markup language. Additionally, the Wiki provides a simple system for back-linking of pages with additional limited source tracking. It also allowed one to create links to pages that did not yet exist, thus encouraging these informational resources to be expanded at a later time. Many of today's hypertext systems - in the original sense of providing pathways through data and information for the sake of knowledge - are based on Wikis or Wiki-like entities. Even though Berners-Lee purportedly created HTML as an easy-to-use markup language, the popularity of the Wiki shows that it wasn't quite simple enough.

Hyper-G was a novel, but ultimately unsuccessful, attempt to compete with the web. It saw six major shortcomings in the web format: uni-directional links, no native search facilities, nonuniform interfaces through variable design, little support for the maintenance of large datasets which leads to data separation,

orientation towards consumption, and the non-scalability of user requests.[3]³⁹ Some of these, such as the consumer-oriented nature of the web and lacking of search facilities, are valid critique. Others, such as the uni-directional links and nonuniform interface, are better seen as advantageous features of the web. Forcing a double linkage or uniform interface would have put an incredible aesthetic and structural restriction on the web. Leaving those things out allowed the web to grow organically as it did. Furthermore, the contention between clean and dirty visions of a hypertext info environment becomes clear.

Starting with Bush's Memex, the very early years of development produced a much different conception of hypertext browsing than what we have in the WWW in 2011. With the exception of Nelson's Xanadu, early pioneers envisioned closed consistent systems for information management, much more akin to current database technology than hypertext environment of the WWW. The vision was for a clean universal medium of data, information, and knowledge exchange. The current version of hypermedia, however, is more open and anarchic in comparison. Today, the WWW includes erratic hyperlinks between sites on one set of computers and sites with their own (in)consistently linked pages. The architecture stipulates that any page can link to any other page without the guarantee

³⁹http://www.mprove.de/diplom/text/2.1.15_hyperm.html was accessed in December 2009. Recently found documents include: <http://www.chemie.fu-berlin.de/outerspace/doc/hyper-g-abs.html>

of a direct back link. The linking structures become the design choice of web developers and not the system. It is what made the WWW so flexible and scalable. At the same time, this *feature* of the WWW makes search engines desirable and necessary, in turn opening up another can of worms.

When hypertext is interlinked with databases with the intent of providing information in a uniform manner, as with Wikipedia or Google Scholar, we see something more like the original intent of hypermedia developers. However, when hypertext is interlinked with databases and with some other intent that is not necessarily information based, such as with YouTube or Facebook, we end up with a self-publishing environments that act like communicative electronic graffiti systems and take on non-uniform appearances and structures unlike what was initially conceived with hypertext. It still retains the non-linearity of the hypertext vision, but I think to a much stronger effect than originally desired and with a more varied and unpredictable outcome.⁴⁰

Although the main purpose may be described as such, browsers have always done more than just deliver content to users. In fact, it is probably more correct to say that browsers deliver users to content, and not the other way around. The way the WWW is used today greatly exceeds the primary concern of the original hypertext developers. Information management and retrieval is now more likely

⁴⁰There were, of course, early complaints about the WWW, that it was too vast and difficult to navigate.

defined as user management and retrieval. The importance of the browser assumes a new dynamic as it becomes a valuable tool for both.

2.4 The Evolution of HTML

In this section, I briefly analyze the historical development of the World Wide Web format - the combination of HTML, CSS, and Javascript. The format grew out of simple aspirations to exchange stylized and formatted textual data. From this evolution, a few things become clear about HTML and its development process. First, the HTML specification was almost always retrofitted to accommodate changes that browsers had already implemented. Additionally, the development of HTML and its partially implemented standards has taken place through a non-linear conversation between developers, users, specifiers, and vendors. Thirdly, not all specifications make it into browsers, and not all browser innovations make it into the specs. Also, while HTML was great for structural page-oriented and text-oriented content, now the new focus is on multimedia. This places attention on the Javascript API and not on HTML protocol. Furthermore, HTML is great for situations that require a separation of form/presentation and content but is bad for presentation oriented content, such as design, where form/presentation is content. Lastly, HTML5 is a brittle 125,000

word-and-growing beast that is overly complicated and, to date, still only partially implemented. I'll start with a brief history of this format.

The display format of the World Wide Web is the HyperText Markup Language (HTML) and was derived as a subset of the Standard Generalized Mark-up Language (SGML) around 1990 by Tim Berners-Lee. SGML is an ISO-standard technology for defining generalized markup languages for static text documents and data systems. SGML is in turn descended from IBM's Generalized Markup Language (GML) that Charles Goldfarb, Edward Mosher, and Raymond Lorie developed in the 1960s.⁴¹ I will describe here a few of the tags, entities, and attributes of HTML so that we can get a general picture of how HTML has evolved over the years.

Markup, as a general concept, is simply a description language for creating, or marking, various logical divisions, structure, and style in a text using text itself. So, for example, the following three figures show a segment of HTML markup that defines an unordered list of elements, each with a text item and a few styling elements. Fig.2.6 shows the code itself as text, while Fig.2.7 and Fig.2.8 show how they are rendered by a graphical and text browser, respectively.

In the markup code (Fig.2.6), textual elements are grouped together between the `` and `` tags. Each individual list element is marked between the

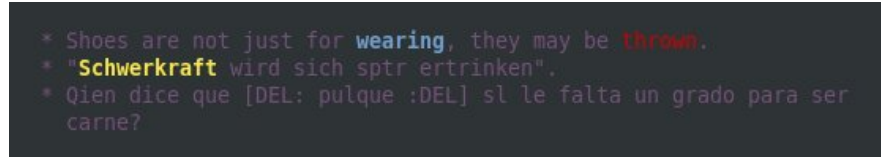
⁴¹<http://www.sgmlsource.com/history/roots.htm>

```
<ul>
<li>Shoes are not just for <i>wearing</i>,
they may be <b>thrown</b>.</li>
<li>&bdquo;<big>Schwerkraft</big> wird
sich sp&auml;ter ertrinken&ldquo;.</li>
<li>&iquest;Quien dice que pulque s&oacute;lo
le falta un grado para ser carne?</li>
</ul>
```

Figure 2.6: HTML markup code showing three items in a list.

- Shoes are not just for *wearing*, they may be **thrown**.
- „Schwerkraft wird sich später ertrinken“.
- ¿Quien dice que pulque sólo le falta un grado para ser carne?

Figure 2.7: HTML markup showing three items in a list rendered by Firefox.



```
* Shoes are not just for wearing, they may be thrown.
* "Schwerkraft wird sich sptr ertrinken".
* Qien dice que [DEL: pulque :DEL] sl le falta un grado para ser
carne?
```

Figure 2.8: HTML markup showing three items in a list rendered by the Lynx browser.

 and tags. Some text items are enclosed by and or <i> and </i> tags to render them as bold or italic respectively. Since HTML was originally only based on ASCII⁴² character encodings, characters that are foreign to the English language and outside of the scope of ASCII need special attention to render them to screen. Characters, such as the ä, ó, and ÿ, require extra formatting and markup. In Fig.2.7 and Fig.2.8, we see how the style and format

⁴²The American Standard Code for Information Interchange (ASCII) is a character-encoding scheme based on the ordering of the English alphabet.

for how markup is interpreted and rendered to screen is left to the browser. As such, the markup only serves as guideline and provides no guarantee for how a browser displays a page.

The first elements of the HTML protocol came, of course, with the first browser prototypes in 1990-1. Initially, tags mentioned in an informal CERN document included things like `<TITLE>`, `<ISINDEX>` `<NEXTID>`, `<XMP>`, `<LISTING>`, `<SECTION>` and `<PLAINTEXT>`.⁴³ Except for the `<TITLE>` tag, all are mostly unused or deprecated at this point in time. In 1992, another informal revision of HTML brings new tags such as `<P>`, `<H1>` through `<H6>`, `<HP1>` through `<HP6>`, `<DL>`, and `` to mark up paragraphs, headings, highlighted areas, definition lists (glossaries), and unordered lists respectively. Many of these, except for `<HP>` tags are still in use today.

HTML 1.0 was first published by the The Internet Engineering Task Force (IETF) in 1993.⁴⁴ It defines the default character set used in the representation of an HTML document to be ISO Latin 1, or its 7-bit ASCII subset. It does, however, include a *charset* attribute in the root `<HTML>` tag to specify other character encoding schemes used to represent the document. It makes no mention though of Unicode characters sets.⁴⁵ It also defines a `<HEAD>` and `<BODY>`

⁴³<http://lists.w3.org/Archives/Public/www-talk/1991SepOct/0003.html> shows Tim Berners-Lee describing his tags to Dan Connolly

⁴⁴<http://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt>

⁴⁵Unicode's history dates back to 1986 <http://www.unicode.org/history/versionone.html>

tag to differentiate between viewable and concealed parts of the HTML document. The <HEAD> is used to define characteristics of a document, such as the title and meta information, that are not seen in the subsequent <BODY> element which encapsulates all display elements that can be seen in the browser.⁴⁶ HTML 1.0 also defines links and anchor tags, <A HREF> for linking two separate documents or two separate parts of a single document. This is the quintessential element of HTML and the WWW. This tag is the web's silky thread. Additionally, HTML 1.0 brings the stylistic elements of , <I>, <TT>, <U>, , , for bold, italic, typewriter text, underline, emphasis, and stronger emphasis respectively. The tag for inline images, , is also included in v1.0, but without specification on what image format will be supported. The very minimal HTML 1.0 was designed to be easy to write, was focused on textual information instead of graphical design, and was made in unison with the HyperText Transfer Protocol (HTTP) to deliver static web pages upon request from server to client (and not the other way around).

At the same time in 1993, Dave Raggett proposed HTML+ as an evolution of the HTML standard including structural elements such as tables, figures, input forms, indexing and mathematical formulas. The proposal was never implemented

⁴⁶This head-body separation is a standard among almost all file formats whether it is audio, video, or 3D vector formats. Because all computational communication is digital all file formats are written in ones and zeros. The computer is thus blind to what you might be telling it to read. A header is necessary to inform the computer how to read the subsequent info.

as a recommended specification, but was instead superseded by HTML 2.0 in 1995.⁴⁷

HTML 2.0 retroactively extended the language to include a number of tags and attributes that were then starting to become common in browsers. The `<LINK>` tag is defined that allows for linking of associated resources such as style sheets in the `<HEAD>` element.⁴⁸ A few idiomatic elements such as `<CITE>` and `<KBD>` are added for marking quotations and display user input (as in instructional manuals). These remain in the standard but are hardly commonplace. The line break and horizontal rule elements - `
` and `<HR>` - are also added to be able to respectively force spacing and draw a horizontal line between two elements lined up one above the other. We can also see the first mention of Unicode character sets (UCS) in the 2.0 spec. The biggest change in HTML 2.0 are the `<FORM>` and `<INPUT>` tags that allow formatted input (text, check boxes, radio buttons) from the user to be sent to the server.

HTML 2.0 remains fairly minimal, and still lacks a number of elements and features that were already seeping into browsers at the time. Both the `` and `<TABLE>` tags are not yet mentioned. Also, the spec does not yet include the `type=file` attribute of the `<INPUT>` tag for transferring files. Furthermore,

⁴⁷<http://www.ietf.org/rfc/rfc1866.txt>

⁴⁸It is strange that the `<LINK>` tag was not used for hyperlinks, but instead to include external scripts and documents inside the current document.

the tag still does not specify encoding formats, but does have this to say: “In practice, the media types of image resources are limited to a few raster graphic formats: typically ‘image/gif’, ‘image/jpeg’. In particular, ‘text/html’ resources are not intended to be used as image resources.”[12] While the W3C standards committee is eager to make specifications, the attitude is nonchalant and there were always limits to what they included, when, how, and for who.

There were many discussions on exactly how to implement non-text based material in the architecture. The tag for adding images is a good case in point. Other suggested methods were for a general <INCLUDE> tag for variable external media, a specific <ICON> tag that had already been implemented in the Midas browser, or for including images in the <A HREF> tag. Additionally, an <AUD> tag for audio inclusion was suggested but never implemented.⁴⁹ Berners-Lee was in favor of a generalized include format and against having a special tag for images. A final consensus was made on an image specific tag for raster graphic content, while at the same time non-standard <OBJECT> and <EMBED> tags were being implemented for more generalized content.

Developed in early 1996 by Dave Raggett and made an official W3C recommendation in 1997, the HTML 3.2 specification retroactively added widely deployed features such as tables, applets, text flow around images, and the font tag.

⁴⁹For a recent listing of this by Marc Pilgram (a google employee), see <http://diveintomark.org/archives/2009/11/02/why-do-we-have-an-img-element>.

HTML 3.2 also makes the first mention of color definitions. The <FRAME> and <IFRAME> tags were respectively developed and implemented by Netscape and Internet Explorer for the purpose of including external HTML content within an existing HTML document. Although only the <FRAME> element makes it into the 4.0 specification, the different aspects of the two tags are discussed by the W3C in 1997.⁵⁰ The <IFRAME> tag is never formally specified, but it does become standard among browsers.⁵¹ At the same time, the non-standard <BLINK> and <MARQUEE> tags are developed by independent browser vendors and heavily used by web developers to allow for blinking and scrolling texts.

The subsequent versions of HTML add support for more multimedia options, scripting languages, style sheets, better printing facilities, and documents that are more accessible to users with disabilities. However, at this time, HTML is forked between HTML4.1 and XHTML. HTML 4.1 is a continuation of HTML under SGML while XHTML is specified to be a stricter cleaner version based on XML. Here, the differences between these forks become increasingly pedantic⁵², and full support for the W3C recommendations for either of them is not really achieved by any browser.

⁵⁰<http://www.w3.org/TR/WD-frames-970331>

⁵¹W3C even mentions it here: http://www.w3schools.com/TAGS/tag_iframe.asp

⁵²See http://wiki.whatwg.org/wiki/HTML_vs._XHTML for an overview.

While all of this protocol development is in motion, style sheets⁵³ and scripting languages are being built and made to work in unison with HTML on the client side. These two text-based technologies, each with their own evolving syntax and grammar, can position, style, and functionally operate on the formatted content of HTML. Each provide aesthetic and functional capabilities, but also define default behaviour⁵⁴, restraints, and limitations. The Cascading Style Sheets (CSS) language of becomes an official style recommendation by the W3C consortium while Javascript still remains officially unspecified, but unofficially accepted.

The styling language of the web is called Cascading Style Sheets (CSS)⁵⁵, and it defines a language for setting the color, position, border, font, and alignment of the various HTML elements. Said to separate form from content, CSS can benefit large sites that have a uniform look and feel by allowing a style sheet to be linked in the <HEAD> of each document and thus allow the maintainer of the site to change elements on multiple pages from one controlling style declaration. Style sheets also give HTML a way to position its elements in an absolute manner. Before style sheets, HTML could specify widths of elements such as tables or headers, but it could not define their heights or position the elements in a specific

⁵³Pei-Yuan Wei's Viola browser boasted of style sheet support as early as 1993. <http://virtuelvis.com/archives/2005/01/css-history>

⁵⁴Default behaviour may not necessarily be a restraint, but does guide aesthetic and functional behaviour in the browser.

⁵⁵CSS was originally proposed by Håkon Wium Lie. His PhD thesis on the topic provides a good overview, history, and detailed insight into the ways and means of styling formatted text. <http://people.opera.com/howcome/2006/phd/>

location on screen. The document elements were simply stacked from top to bottom and left to right.⁵⁶

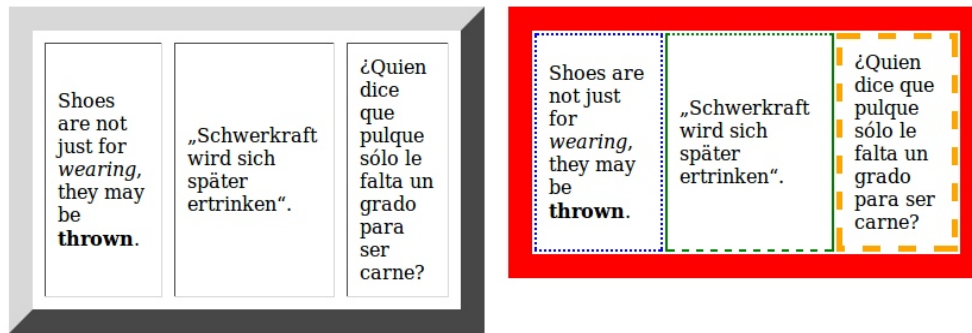


Figure 2.9: An HTML table element that is styled using HTML 1.0 tag attributes on the left and CSS on the right.

Fig. 2.9 shows two tables as rendered using HTML 1.0 markup and newer CSS styling. In HTML 1.0 to 3.2, one had the possibility to define cell padding, spacing and border widths on boxed elements. However, although not defined in any specification, the border is rendered with a colorless simulated beveled edge. The CSS-styled table has more, but still limited, options for dotted, dashed, and colored borders. With CSS, one can also define differing width and line styles for the left, right, top and bottom borders individually. This example is just one of many. Here, we can observe how the two format specifications define a clear aesthetic space of possibilities. In fact, it is almost arbitrary that the possible

⁵⁶This certainly exposes the cultural preference of a reading direction. In fact, there was no technical reason why pages could not be anchored on any of the four corners and rendered from bottom to top or right to left. However, for technical reasons, a single anchor at one of the corners for any given page is necessary for any single element. However, it is not necessary to make position anchors uniform for an entire page.

(default) borders would be solid, beveled, shadowed like the MacOS, or rendered with rainbow-colored naked women.⁵⁷

Near the mid 90s, scripting languages also started to be attached to browsers to allow some logical client-side programming of the web document. While the HTML 3.2 and 4.x definitions define a `<SCRIPT>` tag for including external scripting languages, none define which language to use. At first, there was Javascript (previously known as Livescript) on Netscape and VBScript on Internet Explorer. Since VBScript was limited to Microsoft platforms, it has slowly faded from use. Javascript has become the de facto standard, albeit in various non-standard syntactical forms on individual browsers. Each of these technologies adds yet another non-standard and semi-implemented language to the web developer's tool-chain, making it that much more difficult to learn how to make pages for the WWW. Additionally, with the ability to program (i.e. script) the static marked-up web pages with a functional programming language like Javascript, a radical shift in expectations and use of the web comes about; one that focuses more on application programming interfaces (API) than on the HTML protocol.

With the addition of a scripting language comes the possibilities of motion and procedurally generated content on the client side of the browser. The client browser is then capable of changing color, size, style, and position of elements

⁵⁷I imagine for one second what it would be like if the WWW was invented in more baroque times.

on screen without refreshing the page. Previous to this, the only possibility for interaction or on-the-fly generation of content was to program a server-side application that would load a new page in the user's browser each time she wanted a change in the page. With programmability, we can observe a larger functional space of possibilities unfold in the browser. No longer is the document static and page-based, but is now rendered dynamically. However, because the versions of Javascript included in these browsers have limited bindings to the page document, the browser is only able to perform simple logic and combinations of the color, size, style, and position as defined by HTML and CSS. If the bindings were to be extended to the operating system level of file, socket, and device access, we would already have the beginnings of an all-purpose media scripting language inside the browser. This does not, however, happen. Javascript cannot yet open a telecommunication socket with the server, it cannot draw directly to pixels on screen, it cannot write files to disk, and it cannot get direct access to the audio or video device on the user's machine.

Around 2004, in light of the multimedia developments embodied by Flash and Flex technologies, a new HTML5 standard was initiated to add more interactive capabilities to the web.⁵⁸ Written by Ian Hickson of Google, Inc. and David Hyatt

⁵⁸The non-standard XmlHttpRequest method, that was first implemented by IE5 in 1999, had already begun to revolutionize the static-ness of web pages by allowing a web page to request more data from the server without having to reload the page. At the same time, video sites such as YouTube start using flash as the standard way of displaying video online.

of Apple Inc., HTML5 seeks to define a major revision of the web. It will detail a few new tag elements that reflect common use factors on the web, such as <NAV> and <FOOTER> to simplify the navigation menus and tailing page footers. For practical purposes, these new elements are really just simplified shortcuts to other combined elements. Additionally and most significantly, HTML5 finally defines elements for playback of audio and video in the <AUDIO>, and <VIDEO> tags. However, like HTML 1.0 did to the image format, HTML5 makes no specification on what kind of audio and video should be supported by browser vendors. Furthermore, the current draft of the spec defines a number of additional Javascript programming interfaces.

With backwards compatibility to the previous page-based versions of HTML, HTML5 is attempting to define a massive API for interactive application development inside the browser. For the first time, HTML5 states that Javascript (a flavor of the EcmaScript language) is to be the official programming language of the web. It specifies a number of programming interfaces with Javascript bindings for 2d drawing, extra storage, database connections, server events, client sockets, background processes, and geolocation. The 2d drawing API, as defined by the <CANVAS> element, is an extensive Javascript interface for drawing shapes, patterns, gradients, images and text to screen. Initiated by Apple, some see it as a direct replacement of the Scalable Vector Graphics (SVG) specification that was

developed years earlier but was never adopted by all browsers. The storage API, known as web storage, defines persistent data storage of key-value pair data in web clients. The database API, known as web database, allows for client-side database manipulation using SQL syntax. Server-sent events defines an API for opening an HTTP connection for receiving push notifications from a server, directly addressing document elements with changes in content or styling. The client socket interface, known as the web sockets API, will enable web applications to maintain bidirectional communications with server-side processes. The web workers API defines an interface that allows web application authors to spawn background scripts that run in parallel to their main scripted process. The geolocation API defines a high-level interface that allows latitude and longitude information of the client browser's device to be determined and mapped. This focus on API (Javascript) instead of mark-up display protocol (HTML), signals the new direction of the web away from a file format and into the application domain. In theory, it should give web programmers the ability to write applications that run inside the browser client.

It should, however, be noted that HTML5 is completely theoretical at this moment in time. While some features and interfaces are exposed in the most recent beta-ware browsers, there are no browsers that have implemented the newly drafted specifications in their entirety. This is not unlike the undelivered promises

of Scalable Vector Graphics (SVG) standard. Despite being a W3C recommended standard, few browsers implement it thoroughly and consistently enough to make it universal. Additionally, the spec for HTML5 itself is still in heavy development, and Internet Explorer has yet to show much interest in taking part.

It is therefore hard to determine what exactly HTML5 will become, if it becomes anything. Some estimate that it will take 10 more years to come to fruition. At the same time, given the non-uniform acceptance of various standards among the different browsers and the enormous size of the specifications, there is also no reason to believe that it will be supported in every browser in the same ways. Even though audio and video are now purportedly implemented, the actuality of these implementations leave much to be desired. These time-based media formats are both very desirable and contentious. Vendors have yet to settle on any single format that all browsers can play. Additionally, although we can already see the <CANVAS> 2d drawing api in recent browsers, a notable exception is Microsoft's Internet Explorer, the most widely used browser.⁵⁹ Not only that, we can also see that Google is exerting much influence in the addition of specific geolocation and web database interfaces that will benefit its specific industry of mapping and searching of data.

⁵⁹A *very* recent version of Internet Explorer does support the canvas features. Still, it is unsure how compatible it is with other implementations. The fact that Microsoft is forever dragging its feet shows its malicious intentions.

From this brief tracing of HTML development, we can notice a few things. There has never really been a standard specification for HTML. Any addition or change to the format that made it into an official documented specification was almost always developed after browsers had already implemented the new features. Additionally, not all specifications make it into browsers, and not all browser innovations make it into the specs. The ongoing development of HTML takes place through disjointed and ungoverned interactions between developers, users, specifiers, and the various creators of browsers. Also, we can see how HTML was great for structural page-oriented content, but now the new focus is on the Javascript API and not on HTML protocol. In fact, it is somewhat confusing to call it HTML5 as its real material is the programming interfaces it attempts to define. Furthermore, HTML is good for situations that require a separation of form/presentation and content⁶⁰, but is bad for presentation oriented content, such as design, where form/presentation IS content. Finally, we can deduce that HTML is anything but *pure* ; that its development and evolution is driven by a dirty, hairy, messy and fragmented set of dynamics underscored by a networked animal spirit⁶¹.

⁶⁰This is especially important for seeing impaired users of the web. With an easy separation of text content and display, it is easier, although not without problems, to use a text-to-speech to text-to-braille interpreter.

⁶¹See Keynes's 1936 book *The General Theory of Employment, Interest and Money* and for a description of the mixed influences of psychology and economics that come from an instability due to the characteristic of human nature.

Chapter 3

Problem Statement

If we wish a different society, with more equality and style, it is not enough to think differently; the framework of that thinking must also be overturned. If you want to make a contribution that really makes a difference, then you will have to design the standard for communication of the future yourself. This is the politics of the standard: those who are able to determine the outline of the form determine like no other the culture of tomorrow.[41] Geert Lovink, 2009 Blog entry.

The problem space of the WWW is multi-faceted and large, consisting of a field of interrelated forces and agencies. While the WWW already exists and runs in a very specific, but ever-changing and indirectly governed manner, could it potentially run better? Well, that depends on what one might define as better. Cryptographers and security experts will ask for more security. Networking enthusiasts will want it to run faster and more efficiently. Those attracted to media bedazzle, such as myself, will ask for more media APIs, while some prefer the simplicity and directness of text-only communication. Those that require databases and storage for online sales will have yet another agenda to those who wish to

be sheltered from the psychological manipulation of the public relations industry. There is no one function or intention of the WWW and it is unlikely that there ever will be.

Furthermore, it is unclear that a single isolated problem even exists as these issues are interrelated. Stricter security often means a larger limitation on the functionality of APIs which in turn affects the fluidity of user interaction. More media APIs for the playback of data-intensive multimedia puts a strain on the efficiency of networks. New APIs for direct data storage give developers greater tools for in-browser manipulation of data. This can in turn lead to a more fluid user experience, but also open up security risks and expose users to the surveillance strategies of online marketers and other predatory informational methods.

Additionally, the problem space of the WWW only exists as individual and collective desires and ideologies create it. Before the WWW existed and became popular, it was not a necessary or even imaginable part of lived experience. It has, however, become so. It is a virtual problem with real and virtual components.

The intermixing of contributing qualitative and social factors set the stage for my analysis. If a problem space can be defined that takes into account all of the interrelated issues, the solution is certainly an artistic intervention focusing on the integration of design criteria and critical theory. What is for certain is that it

is not just a technical problem, as many of the builders of this technology often assume.

The problem space that I see and address in this dissertation project is outlined as follows:

- There are currently no alternatives to the WWW.
- Unlike the initial development of the Internet, the WWW is currently driven by commercial interests.
- Web 2.0 methods centralize user data into commercial data silos.
- Standardization process is cumbersome and unnecessary in lieu of FLOSS methods.
- Black-box technologies threaten users.
- Monolingualistic : currently uses HTML/CSS/Javascript for markup, layout, and imperative programming.
- There is still no cross-platform, royalty-free audio/video playback.
- New features favor reading over writing and publishing in the WWW.
- There is low digital literacy.
- Does not account for unknown future formats.
- The WWW is now still mostly rectangular.

The solutions for these problems, however, are elusive, interrelated, and often conflicting. A major shift in focus and function of the web has occurred as the browser accumulates more and more functionality. The browser and the operating system are beginning to meet as one folds into the other. From the side of humanities, there are few scholars that (have time to) know the intimate details of software systems. This leaves little room for serious detailed critique; or a critique that is based on more than words (vapor). From the side of computer systems research, little activity outside of closed commercial application development has gone into any theoretical or practical development for anomalous architectures built for exchanging content in many media over networks. This has encouraged a coercive web environment built by industry where a cooperative environment is in fact demanded.

The question remains: Can a viable alternative even be created given the collective energy that has already been put into the WWW? If so, what possible combination of technologies and implementations could offset or parallel the directional inertia already set by the informational and infrastructural investment in the WWW? And, if there is a significant reason why it cannot be done, why is there no literature on this impossibility; literature that includes an understanding of the semi-soft technologies that make such a communication infrastructure possible to exist and nearly impossible to re-frame?

3.1 There are no alternatives

Currently, the WWW is taken for granted as the mainstay of online communication. Most, if not all, developmental energy is focused on this one amorphous thing. Competition does not come in the form of alternatives to the running system, but instead in the form of tiny incremental *ad-hoc* adjustments.

There has been ample academic attention given to understanding the significance of the web. However, while most of this attention has been focused on the affective organization of networked society, little has spoken specifically about the software or has come from technical motivations upward. Matthew Fuller states this general problem of software critique best when he says that “...software is often a blind spot in the wider, broadly cultural theorization and study of computational and networked digital media.”[27] Eric Kluitenberg goes one step further to describe the cultural significance of software:

The study of contemporary media and technological cultures urgently requires both hardware and software analysis. It needs to understand how networks standards, technical protocols, industrial agreements, the formal logic of computing machineries and the software platforms that run on them affect the production of new forms of cultural signification.[35, p. 20-21]

If we look into the cultural blind spot of which Matthew Fuller speaks, we can see that to date, no alternative and open WWW prototypes have been speculated or built. In the past nineteen plus years since the release of the very first browser

much has changed the operative role it plays in networked social interaction. A recent shift of the web has brought it into an application domain wrought with actuarial surveillance, proprietary social sub-networks¹, and commercial advertising. General discourse on this phenomena sees the libertarian potential in such systems as it democratically (albeit commercially) opens up access on the web for user-generated content and new business opportunities. Others see this “gift economy”[5] and “free labour”[65] economy as at least partially exploitative. Web 2.0 is open for volunteers and service economies, but is not free.[33] It comes at the price of leaving users open to subjectification.[39] A new ontogenesis occurs here as these new meta-WWW networks “...are primarily concerned with establishing the technocultural conditions within which users can produce content and within which content and users can be re-channelled through techno-commercial networks and channels.”[39] New tools at the API and protocol level need to be explored to investigate the assumptions, intentions, and methodology of the WWW.[39][41]

3.2 Academy to commerce

Like the Internet, the WWW lacks any real centralized organization or government. However, in contrast to the development of the Internet, the development

¹Early forms of non-proprietary social networks exist in indymedia, friendster, and tribe.net. They all have seemed to fail. The only one that seems to remain is couchsurfing.com.

of the WWW quickly turned commercial, much to the dismay and detriment of public interest. A new methodology for how to continue the development of the WWW in the interest of the global public is necessary.

The altruistic motivation for developing the Internet came from a very delicate mix of interests. While the main impetus for the internet may have been militaristic in nature and funded through governmental grants, its design comes mostly from academics. At the time engineers were developing the internet, there were also other competing and incompatible commercial networks such as AOL, CompuServe, and Prodigy. These commercial networks, however, “..were crushed by a network built by government researchers and computer scientists who had no CEO, no master business plan, no paying subscribers, no investment in content, and no financial interest in accumulating subscribers.”[72, p.7]

In fact, in some ways, the underlying ideology goes so far as to be anarchist in nature. In a 1992 Internet Engineering Task Force (IETF) conference at MIT, David Clark, one of the founding fathers of the Internet, stated in a talk² “We reject: kings, presidents, and voting. We believe in: rough consensus and running code.”[19, p.543] This short segment, which is sometimes hailed as the IETF’s anthem, is quoted in both hardcore computer science networking contexts as well as the softer social theory contexts. What is rarely considered are Clark’s concerns

²The alternate title for his talk that focused heavily on security was “Apocalypse Now”.

for security and his questions for how to manage future growth that came directly before that in his slides:

As the Internet and its community grows, how do we manage the process of change and growth?

Open process - let all voices be heard.

Closed process - make progress.

Quick process - keep up with reality.

Slow process - leave time to think.

Market driven process - the future is commercial.

Scaling driven process - the future is the Internet.

This anarcho-democratic development is however also somewhat apolitical, focusing mostly on the technical flexibility, interoperability, and perhaps also egalitarian universal access, not any form of social justice. Paulina Borsook writes an insiders account of IETF:

...there's a kind of direct, populist democracy that most of us have never experienced: Not in democratically elected government, where too many layers of pols and polls and image and handling intervene. Not in radical politics, where too often, the same old alpha-male/top-dog politics prevail despite the countercultural objectives pursued. And not in the feminist collective world, where so much time is spent establishing total consensus and dealing with the concerns of process queens that little gets done. The IETF provides a counter-example of true grass-roots political process that few of us have ever had the privilege to participate in...[15]

No doubt, a general like-mindedness of individuals and intentions played a part in their consensus making. They all wanted to connect everyone in the network to everyone else. They were not interested in commercial profit, nor were they interested in controlling or even influencing how and why people connected to each

other. Additionally, the purpose of the Internet was to connect various kinds of networks together and make them interoperable. Thus, they were not interested in forcing any kind of physical or logical network topology on anyone.

A likely motivation for this kind of consensual ideology of technical anti-control comes from the very visible problems seen in the centralized regulation of the early telephone network by AT&T in the United States. The early telephone infrastructure rented the telephone receivers to subscribers and fully regulated the use of not only the network but also the attached devices. It wasn't until a court ruling in 1959 that users could introduce their own devices with which they could re-purpose the network. The ruling stipulated that telephone users were allowed to use their own devices so long as it didn't harm the network.[72, p.22] This paved the way for the answering machine, the fax machine, and the cordless phone.

Another likely motivation comes from government-sponsorship of the project or the academic standing of those involved. Without commercial incentive driving the planning and design of the Internet, the engineers were able to pursue other design ideals of universal access and public participation and use. Additionally, since they were paid by research grants and institutions, they could work on it professionally, devoting themselves full time to its development. This sits in contrast to something like FIDOnet, whose genesis stems from a more hobby-

ist background and whose jerry-rigged infrastructure was fraught with technical problems of scalability.[72, p.29]

Like the Internet, the WWW started as an academic, non-proprietary, and non-commercial project. See Fig. 2.5. The decision by CERN to provide the WWW format to the world without royalties emphasizes this commitment. Furthermore, the W3C's commitment to building a royalty-free universal platform of communication can be seen in their patent policy.[67]

The W3C, however, is only a community that develops and promotes standards to ensure the long-term growth of the WWW. While the W3C may promote and even devise the standards, it does not mean the standards will be implemented, adopted, or enforced. (See Section 3.4) There is no single identifiable party that is guiding the WWW's development. There is also no definable financial or infrastructural model for the creation of new features and maintenance of servers and browsers that run the WWW. The main players in the game are Microsoft, Apple, Google and Mozilla with their respective browsers. Additionally, Adobe/Macromedia and Microsoft weigh in with their proprietary Flash, Flex, and Silverlight plugin technologies. Google and Apple currently head the closed boards at the W3C[68] that are drafting the next implementation of the HTML

protocol and API, known as HTML5.³ Ian Hickson, the lead developer of the HTML5 specification is a Google employee.

Undoubtedly, the mostly unspoken intention of the WWW is to provide a universal and public means of communication. The W3C's mission statement says: "The social value of the Web is that it enables human communication, commerce, and opportunities to share knowledge." It further states that its goal is to "... make these benefits available to all people, whatever their hardware, software, network infrastructure, native language, culture, geographical location, or physical or mental ability." [69]

Undoubtedly, in absence of any publicly sourced funding or any sane and just business model for the public development of the WWW, the trajectory of the WWW is now mostly decided by industry - the ones that have the financial capital and power to influence real change on this virtual environment. This is in direct contrast to the development of the Internet which received large amounts of direct and indirect public funding and support through the military and academia.

A major conflict of interest occurs here where private industry leads the development of a public, albeit virtual, utility. This plays out in a number of interacting and delicate ways.

³HTML5 was previously known as *web applications*. It was started by the Web Hypertext Application Technology Working Group (WHATWG) founded by individuals of Apple, the Mozilla Foundation, and Opera Software in 2004.

Part of the problem is the virtual nature of the digital material involved. Unlike clean water and natural gas, there is an unlimited supply of software. It may be cloned ad infinitum at close to zero cost. Here, under the pretext of a universal and non-proprietary public system, the classic economics of supply and demand do not work. The slow shift from old proprietary development models to the new yet-unnamed models mark the failure of this classical property-driven ideology. Originally, in the budding days of the WWW, browsers such as Netscape were closed-source proprietary items for sale. Now, all browser vendors give away their software to users without financial cost.⁴ Furthermore, because of the immaterial logics of digital software, most of the browsers in use are also FLOSS - Chrome, Safari, Mozilla. Additionally, the shift away from property driven models of software allows the base platforms of these browsers to be shared. Google's Chrome browser and Apple's Safari browser, for example, use the same FLOSS rendering engine, WebKit.

Why are these browser vendors competing when they agree that the WWW *should* be a public and universal utility *and* when a large portion of the WWW browsers are not only free of cost, but also FLOSS? In absence of a property-driven model, is there any real need for competition? Wouldn't a more cooperative de-

⁴Microsoft's inclusion of Internet Explorer with the operating system set the precedence for this. They were subsequently sued for monopolistic behaviour.

velopment environment be more conducive considering this type of public virtual infrastructure?

Another dimension of the problem is that it remains unrecognized. Unlike the public funding of art and science, software development is still seen as a private for-profit industry without cultural or social significance. It is necessary to develop a constituency and critical mass to make this part of the problem mainstream and addressable.

The field of agencies and forces of this conflict are also essentially different than the telecommunication power struggle of the pre-Internet era. Historical precedence has shown that no single party will govern the existence of the WWW. In absence of central regulation, and in the absence of any collective desire for central regulation, commercial interest must turn to the regulation of the users frame of view on the Internet: the browser itself. The fight here is for influence and sway over the entry port into this communication space. In the middle of this fight is the W3C who acts as arbiter of desires.

While the ambitions of the W3C are bold and honorable, there is still too much dependency on industry to provide the egalitarian and cooperative development ideology required by a universal communication environment. Currently, there are still few public methodologies for creating new software systems that benefit and serve public interest. Due to historic circumstances, the WWW lacks the academic

(and military) support that made the Internet a publicly available phenomenon. A return to this older model of academic involvement in the creation of public and interoperable software may prove beneficial. The academy houses hundreds of thousands of well-funded and educated technologists capable of making substantial contributions to the WWW. If the academy, where I write this dissertation, is not the answer, then another model will be necessary. The market and industry driven model based on artificial competition for influence over virtual material is wasteful of human resources (See Sect. 6.1), detrimental to the public interest (See Section 3.3), too slow, and too rigid (See Section 3.4).

3.3 Centralization

The question of centralization is basically a question of hierarchy and power structures. Sometimes, like with television and broadcast media where a central authority delivers content to a passive audience, the power structure is partially inherent in the technology itself. In the case of the WWW, similar determinants are set by the technology, but are more complicated. In the WWW, a mixture of centralized and decentralized delivery mechanisms occur in a more-or-less distributed and decentralized space of communication. Where these methods are centralized and closed to participation, there is a risk of losing hold of the origi-

nal egalitarian and horizontal ideology of the WWW. There are also no incentives within the current commercial models to produce the infrastructure and tools necessary to provide users the ability to produce and distribute their own content outside of centralized holding grounds that either control or manipulate the place or context, if not the content itself, of what is published on the WWW.

Point-to-point systems of communication such as the telegraph and telephone existed previous to television and other broadcast media that dominated the 20th century media landscape. Where the telephone provided a semi-decentralized horizontal infrastructure that put users in direct contact with one another, the broadcasting archetypes were vertical in nature, often with top-down and centralized control. Where telephone provides a technical space in which users create their own content, broadcast media provide content to an audience from a central point. The precariousness of this centralized form of indoctrination is the subject of much critical theory.⁵

In the wake of this development, the Internet has been hailed as a non-hierarchical quasi-utopian and egalitarian communication infrastructure. The Internet promised a horizontal communicational playground where participants can realize both intimate point-to-point and broadcast (or rather narrowcast) communication topologies. Indeed, this is reflected in the design decisions at various

⁵Please refer to Tetsuo Kogawa's MicroFM movement for better understanding of the problem space of media broadcasting.[36]

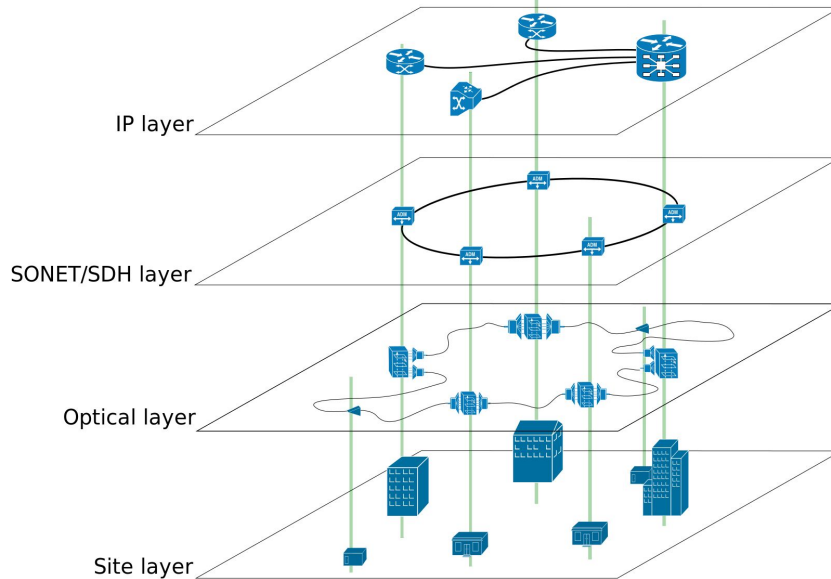


Figure 3.1: Network overlay showing how the IP layer shapes data traffic in a star pattern over top of lower physical ringed layers. source: Wikimedia Commons

progressive iterations of protocols since its inception as ARPANET in 1969. The most significant of these technical decisions are the *end-to-end* principle and the separation and encapsulation of logical communication layers in packet-switched networks. See Section 2.1.

However, this horizontal structure is not without centers. To make a truly center-less network, each participant in the network must be directly and physically linked to every other person on the network (as is in old circuit-switched telephone networks). This is, of course, an impossibility at the global scale of the Internet, even with advanced wireless technologies. Instead, the Internet achieves

a center-less experience by way of protocol layering and overlay.⁶ (See Fig.3.1) Despite the fact that users are connected to one another on the Internet by many central hubs, their user experience is one of direct connection. The much-deliberated design of the Internet, including the technology and policies that make it flexible and to some extent *open*⁷, is what makes this a possibility. The IP stack provides a distributed and decentralized experience on top of an infrastructure that contains central hubs.

At the top application layer (HTTP and HTML), the reverse is possible. Application developers can take the seemingly decentralized and distributed experience of the Internet, as seen through the WWW, and give it a centralized look and feel. Wikipedia, Gmail, Youtube, Vimeo and many other Web 2.0 entities do exactly this, by various means and for various purposes. Wikipedia, for example, uses the centralized technology of the Wiki to create a more-or-less tightly linked web of information. All info is in one logical (although virtual) place. Gmail, for example, gives users a centralized email system on top of the decentralized and distributed WWW. Facebook, Youtube, and Vimeo all create centralized social platforms with various business models and functionality.

⁶An overlay network provides one logical network topology on top of another logical or physical topology.

⁷Recent threats to the neutrality of the net are also infringing upon user freedoms online.

Technically-speaking, these overlay networks are unavoidable. They are built in to the flexibility of the system. There are also often trade-offs between the benefits and detriments of centralized or decentralized systems. For example, having one single place to find information, such as in Facebook, Gmail, and Wikipedia, is often beneficial to users who find such a system easy to navigate and operate. Another benefit that these centralized systems provide to users is a much easier and convenient method of publication. Despite the intended simplicity of HTML and the applications involved in exchanging WWW content, the technology is still too difficult for normal users to write and publish their own content in the bare formats of the WWW. The centralized applications of the WWW, especially those hailed as Web 2.0, provide many accessible methods for users to actually participate in the system.

What is of utmost concern however, is where these centers form and how they are used to gain market power and monopolistic advantages. The newer strategies of commercial ventures in the browser industry that operate under the guise of *openness* , often giving their software products away without cost, is more subtle than what came before. These strategies start with the distribution of Internet Explorer to Microsoft users without cost and continue today with Google and Apple's development of FLOSS browsers. But, not all centralization is for commercial gain.

In the case of Wikipedia, the centralized aspect is of little concern because of two main things. It offers a decentralized means by which users may contribute to the project. The project also exists as a public good, not as private property. This is evident in the licensing and guiding principles⁸. The motivations of Wikipedia, as imperfect and flawed as they are⁹, are thus inherently different than the mega-corporations that seek public attention for private gain.

Where commercial motivation is involved in these centralizing systems, there is a danger of building unbalanced and unchecked power structures. Gmail and Facebook, for example, have a precarious hold on the personal information of their users. Besides major issues of privacy, this form of actuarial surveillance poses an ethical hazard to the general public who have come to rely on their services. It is doubly dangerous because it goes mostly undetected (there are no viewable cameras or recording devices to make the surveillance apparent) and has centripetal force, pulling external user data into the system without their acknowledgment. Even though one user may want to remain excluded from these central systems, they are included by way of association. For example, with Gmail, if one user who has a their own external private email address sends email to a user with a Gmail account, their emails then become part of this surveillance. Both internal

⁸See http://en.wikipedia.org/wiki/Wikipedia:Five_pillars

⁹There is much contention about Wikipedia's neutral point of view (NPOV). See:<http://networkcultures.org/wpmu/cpov/>

and external email addresses provide a valid unique identifier for an individual. They also provide potentially very private information. Given enough emails in the system, Google can accumulate powerful bases of information to target both individual users and collectives with pinpoint advertising and manipulation. In the case of Facebook, a similar scenario is possible. A Facebook user can tag photos and post information about persons external to the system. With these systems, there is no opt-out.

The problem of centralization is also one of infrastructure. If there is no realistically viable infrastructure for users to create and maintain their own nodes in the network, then a truly decentralized and distributed system cannot take place. Currently, there is little industrial and commercial support to create this. This is evident in the APIs selected for inclusion in the HTML5 protocol that favor data consumption and exclude APIs that would give users the power of producing and publishing. I discuss this below in Sections 3.7 & 3.8. It is also unlikely that the commercial market, as it now exists, can create the proper incentive for companies to pursue the construction of these user-oriented tools that empower users with the possibility of direct publication into the system. Furthermore, the mere idea of peer-to-peer production and distribution is threatening to the established commercial power structures that benefit heavily from intellectual

property rights. There are more incentives to restrict and control the means of production and distribution, than there are to support them.

These tendencies toward centralization and consolidation into commercial providers of the WWW, under the allure and guiding disguise of *openness* should be reconsidered. New tools at the level of protocol and API in the WWW may help build a more user-directed and civic WWW.

3.4 Standardization and generativity

Although the drive towards open standards in the WWW is commendable, there is really no need for *ex ante* standardization if developers assume a stable FLOSS development model and disregard the current model of artificial competition set by the long-running but (in the specific case of the WWW) irrelevant commercial ideologies of software development. With a stable FLOSS development strategy, the source *is* the standard. Standardization then occurs retrospectively, while at the same time freeing developers to innovate future APIs and functionality without the unnecessary constraints of a political force of artificial competition. The current standardization process slows the innovation on the web and decreases the flexibility and possibilities of online communication. Furthermore, given the history of incompatibles between browsers, the sad adoption rate

of web protocol standards by commercial and non-commercial browsers alike, and the increasing complexity of the APIs and software involved, there is also little evidence to suggest that current *ex ante* standardization process of HTML5 increases interoperability. Let's look at what standards are and what they attempt to provide in context of the WWW.

A standard can be many things such as basic product requirements or safety standards. A standard may also be deemed open or closed. The definition and production of both of these, the standard and its openness, are areas of contention.[22, p.20-24][37] A standard is an abstract thing. Just because there is a written document that describes the functional specification of some protocol or application, does not mean the standard is common or accepted. WWW users and developers have experienced the lack of adherence to recommended standards since the beginning of the WWW. Where *openness* (transparency of operation, provision of public resources) is necessary and demanded, such as is the case of the WWW, there are two main interrelated components to consider: the openness of the source code of browsers and the openness of their protocol and means of communication. The first issue is one of property and liberties: who owns and controls the source code of this public infrastructure and can thus guide its future development? The second issue is one of compatibility and interoperability: how can the various parts of this infrastructure interoperate and intercommunicate?

With no interoperability, there is no communication. With partial interoperability, there is only partial communication. The openness of the source code of the browser and the protocol that runs within it are interrelated because if developers adhere to openness of the source code, the openness of the exchange protocols is mostly, although not necessarily, assumed.

Up until recently with HTML5, the WWW standards for new functionality have been mostly set in retrospect of their implementation. Important functionality like the and <FRAMES> tags, style sheets, as well as the XMLHttpRequest function of Javascript, have all been first proposed and implemented by browsers and later recommended as standards by the W3C. Currently, the new process of standardization with HTML5 is just the opposite. The WHATWG and HTML5 groups of the W3C are first making proposals and recommendations for functionality. Browser developers are concurrently or retrospectively implementing some of these recommendations. For the purpose of interoperability, time will tell if this is an improvement over the previous ad-hoc method. Both processes are, however, problematic when contrasted with the development and standardization process of FLOSS. If considered a public good and developed under stable and funded FLOSS methodologies, the current standardization process of the WWW that assumes a proprietary field of competition

amongst browser vendors is unnecessary and perhaps even detrimental to growth and performance.

As it is, the WWW and its standards evolve through a battleground of competing browser developers and vendors. Users (WWW site developers and consumers) want more functionality, easy access, multimedia, functionality and services for publishing their own content, and interoperability. Browser vendors and developers, including those that make FLOSS and non-FLOSS browsers alike¹⁰, mostly want lock-in - the ability to secure a sizable population of users that prefer their application over the competition. With a secure lock-in on the market share, a browser developer can manipulate how content is exchanged on the network, bringing a huge influence to the networked experience. In short, (soft or hard) control of the browser means control over the very window on networked communication. For the majority of vendors that are for-profit entities, this kind of regulatory privilege would mean potentially large profits and serious leverage on the psychological frame of mind of its users. It is only a small wonder that Google entered the browser market so late in the game with their FLOSS Chrome browser. The main reason for a company to invest the time, money, and resources necessary to build a WWW browser, without any direct monetary compensation, is purely for purposes of cornering the market.

¹⁰Remember that commercial developers such as Google and Apple fully embrace FLOSS development with their Chrome and Safari browsers, respectively.

At the same time, the control over this window on networked communication is not just in standards alone. It happens in and out of the browser frame, and in and out of the standardization process. The WWW experience is not just defined by what happens inside the browser using the standard or non-standard formats of HTML, CSS, and Javascript. It is also defined by how the browser is designed, what plugins and defaults are set, download and password management, and the general viewing experience with or without tabs or on-the-fly rendering, etc. Among the most important features decided outside of standards altogether are the default settings. The default settings for plugins or internal or external functionality (such as privacy and encryption) are crucial in defining the affective resonance and common behaviour in WWW populations. A current trend is to streamline the WWW user experience with centralized services that link the browser window with selectable applications. The ability to recommend and pre-select the featured applications in this environment, including their default settings and behaviour, will be key to market influence. Considering this, it is no wonder that commercial interest would now participate willingly in the standardization process.¹¹ The battleground is already starting to move outside of

¹¹Microsoft has come in and out of the process of collaboration on the standards, but is now being mostly cooperative although non-committal. See http://www.theregister.co.uk/2011/04/13/microsoft_html5_silverlight/ and <http://arstechnica.com/microsoft/news/2011/06/html5-centric-windows-8-leaves-microsoft-developers-horrified.ars> and http://www.appleinsider.com/articles/11/06/01/microsoft_demonstrates_windows_8_with_html5_apps.html

the arena of the current standardization process. The real and perceived direction of the WWW, including its formats and software, is governed by an ongoing interaction between users, developers, and browser vendors that often make and implement their own technologies. This softens the points of contention among browsers vendors, but only to a degree. Still, even if *open* standards were to be adopted, this does not mean the standards have not or will not be violated.

Violation of standards has been committed by all browser vendors including the main FLOSS contender, Mozilla Firefox. To innovate in the format and protocol of the WWW, it is necessary to break standards. The adherence to standards makes the absurd assumption that there is a singular set way to exchange mixed and multimedia events over communication networks.¹² Additionally, it makes the unspoken assumption that new formats and functionality are fixed.

New non-standard features to the format and functional operation of HTML, CSS, and Javascript inside the browser have had a huge impact on the dynamic shape and perceived experience of the WWW. The XMLHttpRequest function of Javascript is a famous example of this kind of non-standardized API addition. This one function is more or less responsible for the Web 2.0 phenomenon. It allows the browser to connect directly to a server to request new content or post

¹²On the blog for the new HTML5, one developer mentions how “Standards are like sex; one mistake, and you’re stuck supporting it forever!” With sex at least your “mistake” can later make things right. <http://blog.whatwg.org/this-week-in-html5-episode-38> (Accessed Nov. 2009)

commands without making a new page request. This provides the WWW more dynamic feel, allowing a web page to interactively update itself without refreshing the page. It gives the user a more fluid and unified online experience as it allows various services to circumvent the static fetch-and-receive policies of the HTTP protocol. This in turn boosts the speed and efficiency of intercommunication (although not necessarily interoperability) among services.

Due to the competitive field of relations involved, and due to the friction they place on ingenuity and freedom of expression, standards are also hard to popularize. Besides HTML, newly implemented standards like SVG and SMIL are two major W3C recommended standards that have only partial-adoption and implementation rates.

In this light, the problem of the WWW is not necessarily the *openness* of the WWW standards as some critics might suggest. Some say that there would be more interoperability if the commercial vendors would only stick to the standards. However, how can innovation take place if the standards are not challenged and circumvented? A new kind of openness is needed in the process, one that involves and presumes a necessary amount of forward and backward incompatibility. This can be done with or without the *ex ante* standards process. So, why not drop the current process in favor of a FLOSS process that wastes less time and energy front-

loading the standardization of specifications, but instead develops the standards after the fact of their implementation, adoption, and popularization.

It should also not be forgotten that the scalability and popularity of the WWW occurs in contrast to the monolithic, albeit semi-participatory, methods of standardization that are ongoing today. Even though consensus on what is and is not considered standard protocol in the WWW is less than 100%, the protocol itself functions as a soft regulatory system.[29][41] It is a mechanism for institutionalizing behaviour. This is a necessary vice of communication, but one that can be improved. I argue that the slack development strategies of a stable FLOSS environment will provide a greater space of design and a more flexible and consistent (although still not perfect) functionality.

The continuous evolution to better solutions is an important feature of FLOSS development.¹³ Its very openness lends itself to future possibilities and generations of the WWW. The current development strategy of the WWW is clunky and counter-intuitive if we see that the web is no longer a file format on a network, but rather a large dynamic run-time application.¹⁴ Furthermore, the necessity to produce this environment as a public good under FLOSS development methodologies that ensures the rights of users and promotes innovation, precedes and outweighs

¹³This is not to say that there are not hierarchies or competition in FLOSS methodologies. Quite the contrary. However, the hierarchies and competitive pursuits within FLOSS are in general more constructive and less stand-offish than in the proprietary domain.

¹⁴The POSIX standard does describe a consistent application programming interface for Unix-like operating systems and can be seen as a sort of protocol.

the need for the luke-warm semi-cooperative participation in the brittle process of commerce-supported standardization.

3.5 Black-box technologies

Black-box software technologies are binary-only software applications and systems that forbid users to view, study, or modify their inner workings. These technologies are detrimental to the civic space of communication in the WWW as they force users and developers to submit to the technological demands and regulations of private parties. These blackbox technologies are also completely unnecessary.

There are two components to consider in the space of the WWW. One is the source code of the applications for reading and writing exchangeable data. The second is the exchangeable data itself. For a fully open and accessible public WWW, both are ideally available to all users and developers. Exceptions to this might only be contexts where encryption is necessary (banking, private communication between individuals, etc.) In these cases where the exchangeable data is sensitive, the format is still openly readable by the parties involved. The encryption only ensures that external parties are not privy to the communication itself.

The fact that the base exchange format of the WWW (HTML, CSS, and Javascript) is open and viewable to all as a readable text formatted protocol and language is a major contributing factor to the popularity and accessibility of the WWW. Without the ability to view, copy, and modify the source of WWW pages, the WWW would not have scaled as quickly or as broadly as it did.

A major backdoor to the standardization process, the open source nature of the current WWW, as well as any FLOSS development strategy is the binary browser plugin. This application binary interface (ABI) allows potential WWW developers to circumvent and extend the browser's functionality, but without revealing or conceding the source code and associated rights to users. Major plugins that have become part of the WWW experience are Flash, Silverlight, and Java. Flash and Silverlight remain closed. Sun Microsystems, on the other hand, has slowly and painfully open-sourced the Java development environment starting with the Java Core platform 2006. These binary systems are closed systems for direct control, allowing for ownership of the exchanged code.[58] The binary plugin is a closed hole in an otherwise semi-open system.

Previously, it was expected that the plugin would offer developers the context in which to make innovative formats on the WWW. Fortunately, the WWW is slowly changing and offering more and more functionality that would otherwise be offered by a plugin. This, however, is not occurring without contention.

Any future WWW alternative should either dismiss the possibilities of plugins altogether or require copyleft licencing mechanisms that ensure the source code of any WWW application is viewable and usable by all.

3.6 Monolinguistic

One of the features of the WWW and other software applications is the division of operations into language specific components. For example, the main skeleton of a WWW page is written in HTML. As a markup language, its job is to logically divide textual content into separate sections. On top of and in addition to this skeleton are two other languages with their own domain specificity: CSS and Javascript. CSS can style these text elements to change the font, color, and other visual attributes. Javascript is an interpreted scripting language that offers the browser a way to interactively and procedurally operate on the document object model (DOM)¹⁵ and CSS components of a page. Because the WWW format evolved from the original HTML, the additional CSS and Javascript languages arrived only as secondary languages.

Of these languages, Javascript is the most important, however. Because it gives the WWW developer the functionality to drive and control a WWW appli-

¹⁵The DOM is an internal representation of the markup of a page. It is represented as a tree graph with one root element and many branches. This provides Javascript programmers with a means of accessing and manipulating the logical elements of a WWW page.

cation, HTML and CSS become only convenience languages when compared to the possibilities and functionality inherent in Javascript and its several APIs. Unlike HTML and CSS, with Javascript, a browser can inject and draw new content within a page in all ways that are technically possible. Of the three, it provides the largest of possibilities. If given further capabilities to read and write file formats, such as what is being proposed with HTML5 and what already exists in my Underweb prototype, HTML and CSS are no longer necessary specifications. A developer could invent her own markup and styling languages that suit her programming intentions and methods, all driven by Javascript or some other language. In fact, this is already what is happening on the WWW with the JSON formats which developers have now popularized and prefer over the XML styled markup exchange.

Beyond this idea of procedurally driven content on the WWW by means of a scripting language is a polyglot environment in which the browser allows for programming in multiple languages, not just one scripting language. While limiting the development environment to one language may provide some needed constraints on an otherwise complex multifaceted software environment, I believe it is also too constraining. With new introspection technologies¹⁶ that allow script-

¹⁶These are mostly developed under FLOSS practices since they are not constrained by the commercial intent and can openly seek out methods to inter-operate with external programming language technologies. In a commercial setting, this is generally not the case.

ing languages to automatically and seamlessly bind to external APIs, binding to one language is about as easy as binding to another. Even though any general purpose programming language is Turing complete¹⁷, the syntaxes and semantics of these artificial languages provide developers with unique characteristics that they may prefer or that may be specifically tailored to address the task they wish to complete. In this way, the diversity of artificial languages is not unlike the diversity provided by the many natural languages that exist.

Furthermore, in addition to interpreted scripting languages, the browser could also offer a more low-level access to APIs, input devices, and output devices. This would provide the developer with more powerful and direct means of programming an application for the WWW. For instance, because Javascript is a weakly typed interpreted language and cannot directly manipulate memory, it is unable to process data in an efficient manner. The trade-off is that it is easier to program. However, if the WWW were to allow a strongly typed language with direct (and possibly sandboxed) memory access, it would give developers many more possibilities to innovate. This would, in turn, lead to a very different space of communication and activity on/in the WWW.

¹⁷In computability theory, a rule-based system that can manipulate data, such as any programming language, is said to be Turing complete or computationally universal if and only if it can be used to simulate any single-taped Turing machine.

3.7 Audio/Video playback

It has only been a little more than fourteen years since browsers have allowed the exchange and display of images (albeit in a limited format.) It has only been a little more than four years since browsers allowed images with a fourth alpha channel for image overlays and transparencies. There are still no common or uniform <VIDEO> or <AUDIO> tags in the standard that allow for the playback of time-based media without an external player or plugin. The playback of audio and video in the WWW is such a desirable, but also basic, component that one would think it would already be a standard specification. However, after twenty years of development including multiple open source and royalty-free codecs, audio and video are still not part of the official WWW. The lack of a standardized and *open* audio-video playback component of the WWW casts a huge shadow of doubt on the entire process of standardization and leaves WWW users with the undesirable situation of using blackbox technologies for playback.

The overwhelming majority of video playback is handled by the proprietary and closed-source Flash and Silverlight plugins. Furthermore, as various commercial entities vie for leverage to inject their preferred media formats and codecs into popular use on the WWW, there is no real sign that audio-video playback will ever become standardized. Even though there has been limited and disjointed

progress to get simple media playback into the browser, it is still very unclear whether any consensus will be reached on what codec¹⁸, if any, will be standard in all browsers. Without any open and standard codec that is supported everywhere, the problem remains the same; no reliable, consistent and uniform video playback on the web.

In many ways, the debate over audio-video playback resembles the issues and deliberations over the tag of the first iterations of the HTML specification. Without any consensus on what format to use, the W3C simply forwent the specification of a format altogether. (See Section 2.4) Like the initial specification of the tag, the <VIDEO> tag only states that a video is ontologically defined in the markup language without saying which codecs or container formats are recommended as standard. Video is, however, much more technically and politically complicated.

The technical complications of video stem from a number of inherent features of motion pictures. First, they are data intensive and require intricate and complicated algorithms to make their data streams smaller so that they may be transmitted on the web. Whereas an image requires compression on a single two dimensional grid of rectangular picture elements, a video necessitates compression

¹⁸Codec is a portmanteau that stands for encoder-decoder. It is the algorithmic recipe for how to compress audio and video data into a smaller bit stream for storage and for transfer over the Internet.

in both spatial and time domains, often with extra problems of data packing and audio synchronization. At the same time, the target and scope of compression is ever-changing. Increased bandwidth on the WWW and increased screen resolution is continuously offsetting the desirability of previous compression formats that were designed and tuned for low-bitrate communication. Secondly, in addition to compression, contemporary video content may also consists of subtext elements, multiple audio or video tracks, audio in various channel configurations, as well as clickable interactive components. Additional desirable features for video on the WWW that have not yet really been developed are bit-rate peeling, synchronization with other media, time-based tagging and linking, indexing methods for searching, and multicasting infrastructure. Each add a significant amount of technical complexity to the issue. This is without even considering a standard method of playback - where to put play and pause buttons, whether or not there should be a scroll bar for time seeking in the video, etc.

The political ramifications are also quite large and complex. Audio and video are arguably much more alluring than singular images or even web pages. The market for music and televisual content is thus absolutely immense. Therefore, the contention over who or what governs this territory, from the methods of distribution down to the technical formats, is that much greater and significant. Since audio and video are arguably the most contentious areas of intellectual property,

none of these parties are eager to support an open media format without first securing some revenue model from which to hold and retain intellectual property value in the market. Some parties, such as Google and Apple who already have ample distribution methods, are more willing to submit to open formats. However, it is not just a simple question of who wins out, but also one of whether a market exists or not. Without an institutionalized technological format (proprietary or not) and political method of enforcing so-called property rights, none of the parties will be able to reap financial gain from the system. Furthermore, given the political climate of this media, further technical requirements such as watermarking, copyright metadata, and content protection may become necessary, adding complexity on top of complexity.

In lieu of all of this, the HTML5 working group originally recommended the Ogg container format and the Theora and Vorbis compression formats for video and audio, respectively.¹⁹ Ogg Theora and Vorbis both have great compression with great image and audio quality. They also perform well using little processing power to decode. Compared to alternative proprietary formats they are neither technically much better or worse. Above all, these two formats meet the royalty-free requirements of the W3C. But on December 10, 2007, all references to the

¹⁹See <http://html5.org/tools/web-apps-tracker?from=1142&to=1143> to view the patch file that marks this change in the written document.

Ogg formats (or any other concrete format) in the recommendation have been dropped. The specifications now state:

It would be helpful for interoperability if all browsers could support the same codecs. However, there are no known codecs that satisfy all the current players: we need a codec that is known to not require per-unit or per-distributor licensing, that is compatible with the open source development model, that is of sufficient quality as to be usable, and that is not an additional submarine patent risk for large companies. This is an ongoing issue and this section will be updated once more information is available.²⁰

On December 12, 2007, just two days after the retraction of the Ogg recommendation, the W3C hosted a *Video on the Web Workshop* where international parties submitted position papers on the subject of audio and video in the WWW.²¹ The presented positions were fairly predictable. A majority of the presenters exhibited the desire for a royalty free and uniform audio-video format. There were some notable objections, of course. Adobe stressed its premier role as the number one delivery mechanism for online video content and argued for the adoption of a proprietary format, one that is supported by its Flash player.²² Disney was outright against open standards, saying it "...is in the business of creating compelling online experiences, and our goals may at times conflict with the accessibility and openness that standards encourage."²³ Nokia was either misinformed and uneducated about free software or directly attempted to manipulate the audience with

²⁰See <http://www.w3.org/TR/2009/WD-html5-20090423/video.html>

²¹See <http://www.w3.org/2007/08/video/positions/>

²²See <http://www.w3.org/2007/08/video/positions/adobe.pdf>

²³See <http://www.w3.org/2007/08/video/positions/WDIG.html>

complete misnomers. It said “Anything [...] including a W3C-lead standardization of a “free” codec, or the active endorsement of proprietary technology such as Ogg, ..., by W3C, is, in our opinion, not helpful for the co-existence of the two ecosystems (web and video), and therefore not our choice.”²⁴ Youtube, who already has a largely profitable online video system in place, made no statement about formats. Instead it stressed the potential for making money: “The potential for Video monetization is clearly there – just a matter of when.”²⁵ Apple, who is an aggressive opponent of Adobe’s Flash²⁶ says that “...the standardized support of multimedia elements at the HTML5 level is valuable even if we cannot immediately settle on audio/video and container formats.”²⁷ Both Microsoft and Google did not submit position papers. Their silence is telling in and of itself.

Without a format recommendation from the W3C, the decision is left to market forces. This risks exposing users, unnecessarily so, to proprietary formats that may demand royalties. The history of both the GIF and MP3 formats demonstrates this. CompuServe released GIF as a free and open specification in 1987. After becoming a popular format for the WWW, Unisys tried to impose royalties on users for its patented LZW compression used in the GIF format.[7] A similar scenario occurred with MP3. Prior to 2002, Thomson Multimedia’s MP3 licensing

²⁴See <http://www.w3.org/2007/08/video/positions/Nokia.pdf>

²⁵See <http://www.w3.org/2007/08/video/positions/Youtube.html>

²⁶See <http://www.apple.com/hotnews/thoughts-on-flash/>

²⁷See <http://www.w3.org/2007/08/video/positions/Apple.pdf>

policy was very permissive, stating that “...no license fee is expected for desktop software MP3 decoders/players that are distributed free-of-charge via the Internet for personal use of end-users.”²⁸ Any proprietary influence on the format of audio or video will subject the WWW to the uncertain whims and desires of private interests.

The standardization process is also very arduous. In 2011, four years after this meeting and seven years after the inception of the HTML5 working group, there is still no consensus on formats. Instead, there are now three competing codecs at hand. Ogg Theora/Vorbis remains one of the most significant contenders. H.264, the leading codec in consumer grade video cameras and arguably the most popular among industry, is the second one. Even though H.264 has open source encoders and decoders available, it is still encumbered directly by patents and royalties. In many countries it is illegal to use without paying license fees to the MPEG-LA. A third is a new one called Webm that was specifically developed by Google to be WWW ready. To do so, Google acquired a video codec company named On2 in 2010. Subsequently, Google released On2’s VP8 video format as a royalty-free open standard. The Webm format combines VP8 video with Vorbis audio in

²⁸This is quoting an article (http://www.theregister.co.uk/2002/08/30/mp3_codecs_no_longer_free/) that quotes the licensing proposal from archive.org’s way-back machine. This machine attempts to archive the WWW including the versions and changes of a single site. Both the original licensing site and the way-back machine’s archived version are no longer online. Only the quote from the article remains.

a Matroska container for a full-fledge *open* codec competitor with heavyweight industrial support.

Browser	release	Ogg Theora	H.264	Webm (VP8)
MS IExplorer	9.0	no	yes	no
Mozilla Firefox	5.0	yes	no	yes
Google Chrome	13.0.782.112	yes	no (re-moved)	yes
Apple Safari	5.1	no	yes	no
Opera	11.5	yes	no	yes

Table 3.1: Listing of current browser support for the three main video codecs.

There are, however, problems with all of these and Table 3.1 shows the lack-luster support for any single one. First, because of the competitive playing field and high stakes involved, no one party is likely to concede ground to the other. Microsoft, for example, would have to surrender a lot of market territory (directly through the video format itself or indirectly through influential sway on the market) if it were to adopt the Webm format from Google in their Internet Explorer. Secondly, unlike the closed proprietary formats such as Flash and Silverlight, the open formats give no safe-guard against online piracy. For Netflix, which is a major video provider in the US and which runs on Microsoft's Silverlight, a switch to an open format would jeopardize their entire business model. Lastly, like with Ogg Vorbis and other royalty-free codecs, only time will

tell if Webm is subject to submarine patents.²⁹ A blog post from YouTube describes some of the finer more technical problems with an open video format: <http://apiblog.youtube.com/2010/06/flash-and-html5-tag.html>.

Keeping the guidance and development of the WWW in the hands of the software industry is proving to be as much of a problem to the WWW's ongoing evolution as it is a benefit. With ongoing contention about the very basic and straightforward issue of video playback, user experience and personal expression on the WWW suffer. This points yet again to a need for a switch to FLOSS development strategies. However, at the same time it points to a need to circumvent this techno-political quagmire altogether with an entirely new WWW framework.

3.8 Read, Write, Publish

The WWW includes a number of functionalities that provide users with the limited and disjointed ability to consume textual and multi-media content. It, however, lacks any real production tools for direct participation. Furthermore, the current specifications and recommendations of the W3C refrain from developing

²⁹A submarine patent is a patent whose issuance and publication are intentionally delayed by the applicant for a long time. The filer of the patent waits until the innovation is fully implemented to demand royalties from the party who did the real work. Please see <http://www.thisamericanlife.org/radio-archives/episode/441/when-patents-attack> for an in depth account of current mafia-style patent practices in the software industry.

them, despite the fact that developing a writable format was at the very inceptions of the WWW.

Tim Berners-Lee often reminisces about his initial conceptualization of the WWW including both a viewer and editor.³⁰ With an editor, users would be able to not only consume content on the WWW, but also write it. In the brief history of the WWW, there were attempts to build HTML editors directly into the browsers. These efforts, however, were entangled by problems of technical complexity and lackluster interest among browser vendors.

The technical complexity of writing in the WWW has to do with two things. First, the format is not as easily writable as the creators believed. If they were, overlay markup languages such as Wikis would not have become as popular as they have. In a 2005 interview with the BBC, Berners-Lee has this to say: “The idea was that anybody who used the web would have a space where they could write and so the first browser was an editor, it was a writer as well as a reader.” He continues: “What happened with blogs and with wikis, these editable web spaces, was that they became much more simple.”[8] The second most significant technical entanglement to writing in the WWW is the server-client split. Writing the format of the WWW is not the same as publishing in the WWW. Because the server and the client are two separate applications, mostly on separate machines altogether,

³⁰An interview on this subject can be viewed at http://www.readwriteweb.com/archives/interview_with_tim_bern timers-lee_part_1.php

there is an necessary extra step in publishing content that requires the user to upload her data to a WWW server. Uploading both textual and multi-media content has proven to be too difficult for the majority of WWW participants. The wiki, the blog, and the social networking sites minimize the effort necessary to both write and publish. It could, however, be even less complicated, involving fewer mediated steps, and without the constraints of centralized commercial services.

The lack of enthusiasm for creating a writable web is also historically notable. NCSA Mosaic was the first successor to the original browser that is credited with popularising the WWW. It abandoned the editing features of its predecessor entirely. The Netscape corporation briefly sold a version of its browser called Netscape Gold that included editing features which was also later abandoned. Since then, none of the major browsers on the market included a browser with default editing capabilities. Needless to say, none developed any kind of infrastructural support for publishing either. This was left to the free market. Furthermore, the ability to publish in the WWW remained a quietly unnoticed issue until the blog and wiki scene grew in popularity and social networks exploded as a new phenomena.

Where users are able to view a specified media format on the WWW, they should not only be able to write that format, but also publish it.³¹ This can take on many forms and demeanors, the most significant of which are server sockets and audio-video encoding and streaming. These are fairly easy, although not trivial, to implement.

While new functions for bi-directional communication are being defined in the web sockets recommendation that pose a vast improvement over the single directional protocol of HTTP, it only exists as a client initiated service. With web sockets users can connect to a foreign server, but are not allowed to act as a server themselves. For this, server sockets would be necessary, but are not proposed.

Even though the controversy continues over audio-video playback on the WWW and should be commended for the effort put in to reach a universally compatible solution for all, the outcome still leaves much to be desired. One ideological step beyond the playback of time-based media lies the problem and necessity of universal encoding and streaming audio-video content on the WWW. Despite the fact that plugins such as Google's voice chat software and the proprietary Skype already demonstrate the popularity of transmitting a live moving image (web

³¹Some might argue that there is minimal desire on the WWW to produce, or that there will always be more informational consumption than production. However, without having the necessary tools for production made available, this is a difficult argument to hold.

camera) on the WWW, the overall silence on the provision of these tools in the standardization process is telling.

In contrast to the need for productive tools, the HTML5 working group defines a number of new functionalities (some of them partially implemented in browsers) that provide users with new ways of viewing and interacting with WWW content. The focus of the WHATWG and HTML5 group is (at this time) on such things as the canvas 2D drawing methods, drag-n-drop frameworks, cross-document messaging, microdata, and database functionality known as web storage. Another noteworthy API that the W3C is developing, but that is not part of the WHATWG or HTML5 group recommendation, is geolocation. These are all fairly interesting and useful technologies, but also exposes the primary interest of current commercial ideologies. The interactive components are especially attractive to commercial interest because of the allure and perceptual commitments it requires of users. The microdata recommendation, which allows for the nesting of semantics within existing content on web pages, is a priority for a large part of the software industry that make revenue from search methods. While the ontological schemas that companies like Microsoft, Google and Yahoo create may demonstrate informational prejudice, the technology is not suspicious in and of itself. However, the fact that it is a priority over other possible software technologies such as audio-video encoding capabilities does demonstrate the assumptions and needs of its

makers. The same can be said of the new database functionalities of web storage and geolocation.

Other problems of developing a read/write/publish system have to do with more generalized problems of infrastructure and literacy. The infrastructure for hosting a server and making content public on the WWW is still quite deficient. Most WWW users do not run their own servers and therefore rely on third-party services. This is changing, however, as mobile device technology improve and become cheaper. Furthermore, as bandwidth becomes broader and more ubiquitous with new wireless technologies, the trajectory towards building a more direct infrastructure for WWW users becomes that much more possible. New projects like the Freedom Box³², that was initiated in 2011, already show a desire to build this kind of infrastructure necessary for do-it-yourself participation in the WWW. Another significant factor in establishing a productive technological environment is how to provide the necessary know-how to would-be writers in the WWW without dumbing down the technology and without subjecting users to centralized services of surveillance and predatory marketing. I discuss the problem of technological literacy below in Sect. 3.9.

³²The freedom box is a project by Eben Moglen that sees itself as a “A publishing platform that resists oppression and censorship.” See <http://freedomboxfoundation.org> for the project page.

It appears as if the WWW is and wants to be a participatory medium. However, without a general inclination to provide the productive tools necessary to contribute content directly into the space of communication, the WWW will never be able to fulfill this ideological predisposition. Without the specific functionality that would provide WWW users and developers with serious and accessible means of production, such as server sockets, full audio-video playback, and media encoding, the WWW is crippled by a technologically defined consumer-orientation.

3.9 Literacy

With the (artificial) language-based systems of the WWW also comes the problem of literacy: who is able to both read and write these systems.³³ It is difficult to build up to literacy when the technological formats are ever-changing and the tools for writing and publishing are next to non-existent. Any new format or framework for the WWW must also provide layers that make it easy for new

³³This is separate from, but related to, the problem of publishing. It is also related to the general problem of increasing technical sophistication. Imagine a somewhat terrifying scenario where a future smart home requires a technology specialist to replace a light bulb. The current political hubbub surrounding the simple light bulb is telling. See http://en.wikipedia.org/wiki/Phase-out_of_incandescent_light_bulbs. Not only is the light bulb the purest of media according to McLuhan (it is a medium without a message), it is also becoming a nexus of policy debates and philosophical discussions on the sustainability of the sustainability argument. The care of new federally mandated compact fluorescent lamps (CFLs) require specialized knowledge to discard. Unlike the incandescent bulb, it carries toxins that are harmful to the environment. Here, the new medium of light requires extra literacy to *use*. This *use* is also both more direct and basic than simple reading or writing. It is a kind of total literacy where reading, writing and publishing in the medium of light converge by way of directly engaged experience.

users to write and publish in the system. The design emphasis must be placed on giving users not only powerful tools of expression, but also universal access to the means of production. This entails either education or design simplicity, or perhaps both.

One of the advantages of the original HTML format was that it was a fairly easy language to learn and use. It was based in English and only added a thin layer by which users could add structural definitions to textual documents. A header was placed in a simple <H1> tag. Bold face text was declared within a tag, etc. Even though HTML had a minimal vocabulary and can be considered much easier than today's very complicated versions, it was also intended to come with a user interface editor so WWW users could easily read and write the web format.[17] This editing capability, however, fell to the wayside as new browsers raced to implement new features. Wiki's and similar overlay systems are now able to serve as some form of editor, but within the browser confines. Often, this comes at the price of having to go through some centralized and minimized method of formatting content.

Furthermore, media literacy behaves much like and unlike literacy regarding the book and related formats. Like printed literature, these new media formats require an educational component to make use of them. Just like the affect the book had on the societies of the fifteenth century, as these new media systems

become more and more enmeshed in the daily ongoings of human culture, the more important literacy becomes for everyone.³⁴ However, unlike the book, these media systems come in pluralistic and ever-changing formats. The complexity, flexibility, and changeable nature of the new technological formats eludes any direct measure of literacy.

It is still unforeseeable whether or not anyone will be able define a fixed idea of what literacy means in these new informational spaces. Also, unlike the book's very regimented format, these media systems have multiple layered languages and ways of interacting. Their literacy, therefore, also comes in various kinds and levels. There is the expert level that knows the very low-level functionality. There is also the novice level that may or may not be able to navigate the surface functions of these systems. In between, lies a broad gamut of literacies. If the development of these systems are left to experts, they often come up with expert solutions that are often very flexible and robust, but too difficult and complex

³⁴It may at some time become necessary to teach basic programming skills to everyone, the same way that reading and maths are taught in all schools. A basic assumption of contemporary society is that it is better to be literate than not. Being illiterate is seen as a handicap. Why do we not assume the same thing for other artificial languages that program our co-evolving technological machinery? Is this language and literature that operates our techno-machinery not as significant, if not more so? Also, the language of computer programming consists not only of many languages, but of many subjects. The grammar, syntax, and semantics of these languages are fairly easy to learn, but the subjects and objects of this language are as vast and diverse as are all knowable subjects. Sometimes, the application of computer languages to a specific subject opens up new areas for the subject matter. In math, there are new problems that are solved by computational proofs. In the visual arts, there are new areas of imaging, video, and design, that are opened up by computational and procedural instruction and that come directly from programming languages.

for novice level literacy. If by some strange chance the development of a media system is left to novices, there is a large chance that it will be too specific of a solution to remain interesting for long or prove powerful and useful enough for others.

Furthermore, if media designers only cater to the needs for convenience by users, it is possible they create a slack educational response where users *read* software only as products and not as a medium of production; where their participation is key to the very essence of the medium. The indirect side-effect of this is a consumer oriented computing culture where users expect things to be made and delivered for them in a package. The software is supposed to be the friend of the user, and thus user-friendly. It is not supposed to demand anything of the user that lies outside so-called intuition and common interfaces.³⁵ If, on the other hand, designers request too much from users, their media objects go unused or only used by experts. This is essentially the culture we now see growing around mobile phone applications and cloud computing. Lazy users - who may be rightfully lazy - gravitate to what is easiest to use, further reinforcing their lack of literacy. What is needed is a user-elegant approach that at the same time seeks to build a system that balances the various proficiencies and capabilities

³⁵Unfortunately, the common-ness of these interfaces, much less the illusions they pose to users through metaphors, lack any serious critique. See *Command Line Poetics* by Florian Cramer for an exception.[21]

of users while maintaining the intent to provide powerful methods of production. The target level and placement for this balance will also undoubtedly change over time as new functional operations are invented or desired.³⁶ Altogether, this is surely a difficult task, but one that is still unverbilized and unseen in the current production of the WWW. The issue is further complicated by the fact that any real measure of technological literacy is lacking.

If we consider the articulation of communication on the WWW as languages or methods of textually mediated intercourse, there are at least two layers in which it occurs. The layer above the screen is *speaking* the language of newspapers, magazines, music, video, television, and film.³⁷ This is the layer at which we *read* media objects of all kinds. Another, sub-operative layer below the screen is *speaking* multiple structural languages of an acoustic/ambient or informational nature, with protocols for interactive manipulation of the top layer.

The manipulability, programmability, and interactivity components of the sub-layered languages is what makes literacy such a difficult item to identify and

³⁶I also believe, somewhat counter-intuitively, that there will be a natural tendency to more and more complex interfaces as newer generations of users become more and more literate, more and more able to not only read but also write in these systems. I believe they will slowly drop the simple systems for more complexity and richness as they learn how to operate them. The question then will be, what force is leading this trail to literacy? Unfortunately, I think it will most likely be the same convenience-providing centralized commercial systems that now exist. The ease of use will then be masked by the centralized infrastructure that maintain and own the means of productions instead of providing the means of production first hand.

³⁷Video, film and television are separate, but similar, *languages*. See *Vernacular Video*[56] and *Cinematic Video*[55] by Tom Sherman for an interesting take on the now dominating factors of digitization that have changed both film and video.

define. This includes not only the ability to click, navigate, and maneuver the informational landscape of the WWW, but the ability to write and construct ones own data path. Because the WWW consists of users and their content, the ability of users to directly publish is of utmost importance. The layers of technology and infrastructure that mediate between users should be as thin as possible and lie outside of private interest, much like how the Internet functions. Where this is not possible, open and accessible default capabilities are needed to provide sufficiently usable tools so that the actual ability of users to write in the system is not overshadowed by the technical difficulty of doing so. As it is in the WWW today, the tools and avenues that are provided are centralized commercial services such as YouTube, Gmail, Facebook, etc. which come with their own problematics (See Sect. 3.3).

3.10 Unknown and multiple formats

The technical makeup of the WWW consists of functions, languages, and parameters that allow developers to craft and design information so that it users can read, view, hear, or otherwise experience it. Currently, the number of representational formats allowed on the WWW is considerably small compared to the number of possible existing formats. With a new development strategy based

on FLOSS methodologies, more formats could easily be included. Additionally, unknown future formats would be more easily implemented and adopted. If we consider what the intention and possibilities of the WWW are, this becomes more of a prerogative.

The space of possible future applications of the WWW is certainly larger than the present one. The projective communication and design space of the WWW includes the space of all possible applications on a single computer as well as the interactive and inter-networked relationships it forms with other computers and users worldwide. It is the space of overall convergence of office software suites, photo and imaging applications, and soon to be audio-video operations into one networked mega-system. It includes functional processes for generating, editing, and manipulating data of all (plausible) sorts and kinds. Not only does it include all of these conceivable file and application transfer formats (FTP, HTTP, OSC)³⁸, file formats (.wav, .mp3, .flac, .doc, .xml, .pdf, .ttf, a.out, .blend)³⁹, and display formats (CRT monitor, LCD, touchscreen, loudspeakers, cell phone, tablet, and projections) it also includes the many logical groupings and decisions that go into defining those formats. Furthermore, it includes the technical possibility to

³⁸FTP is file transfer protocol. OSC is open sound control. In some ways it also includes lower transport layer formats such as TCP and UDP.

³⁹For a long, non-exhaustive list of file formats please see http://en.wikipedia.org/wiki/List_of_file_formats

interlink each of these processes with one another, including format translations where necessary.

The possible formats of communication on the WWW are, thus, only physically bound by the digital resolution of display devices including their ability to represent colors or acoustic frequencies, network bandwidth, computing power, and input mechanisms such as the keyboard, mouse, and tablet. These physical limitations, however, already offer an incredibly vast, almost infinite, space of possibilities. The logical (and virtual) formats, as well as the procedural languages that drive them, help designers and developers search and navigate through this incredibly immense space of possibilities. These formats do not exist without problems, however.

The problem of logical or virtual representation is essentially two things: knowing first of all what to represent, and then how to represent it and encode it into a format that is transferable, communicable and perhaps translatable. For the former problem of knowing what to represent⁴⁰, one must consider if the visual representation of text is different than an image? Is an image different than an idealized graphical element such as a circle? For the latter problem of how to

⁴⁰All representational formats must also make a number of assumptions, not only about the kind of informational content that it should capture and contain, but also how it *should* be used by its users. Before getting to secondary questions of how to represent real world data, difficult primary questions such as “what is a frame?”, “what is an image?”, “what is a camera?” [16, p.65] are already tough to answer. Even simple questions such as “what is a color?” and “what is a text?” cannot avoid the problems of representation. The limited formats of the WWW make more assumptions than what is necessary.

format a logical representation, a developer must ask herself if programs should encode circles and other idealized elements as a raster matrix or as an idealized form such as a mathematical equation or semantic encoding? Furthermore, with elements that vary over time, how many different ways can time be systematically encoded?

The problem is further exacerbated by the large intentions and loose formatting constraints of a universal WWW that *should* be able to communicate any plausible format. Within specific and tight constraints, the problem of representational formatting is minimized. Television, for example, has a limited and standardized frame size and frame rate that are agreed upon by sender and receiver. The same kind of format limitations is true for telephone in its various forms. However, in the WWW, where many media converge into one, the problem of format is a fairly large problem couched within unspoken demands of universal expressibility - the unwritten goal of providing a means by which anyone can say anything to anyone else at any time and for any length of time.

With this in mind and excluding the possibilities for sound, tactile, or olfactory transmission, one could certainly imagine a WWW consisting of a bare minimum framework that can directly address a rectangular grid of colored pixels through some programmable API. Since, the actual visual frame of any WWW browser is only composed of color units or pixels, this would indeed offer the developer the

least amount of practical restrictions on what she can develop within the physical space of gridded and rectangular units of light on screen. While direct pixel access to the screen is necessary, it is also incomplete. A developer needs more guidance and shortcuts in navigating the huge space of possible designs that are available within the screen resolutions we have today. This guidance comes with the formats, APIs, and programming languages that give the designer high-level components that she can use to combine and aggregate elements on screen.

Above the most basic design requirement of pixel manipulation lies another compositional layer consisting mostly of three basic high-level components: text, raster graphics, and vector graphics. Additionally, one could add an extra acoustic element to this basic setup so that sound and image may inhabit this space simultaneously and perhaps synchronously.⁴¹ How these sub-compositional elements are encapsulated, combined and prioritized is up for grabs, so to speak. The way they are assembled into a method that produces designed elements on screen also has a number of ramifications.

The implication of these loosely bounded representational limits finds an expression in the WWW that is mostly understated. Because the combination of these basic elements acquire an expressive articulation that is much more than the sum of its individual basic parts, theorist also attribute similar attributes to

⁴¹One could also add newer mathematical criteria such as Geometric Algebra as a basic component.

this kind of visual or acoustic composition as they do language.⁴² Like language, the specifications and parameters of these low-level combinational aspects display properties similar to grammars and syntaxes where their combinational phrasing greatly influences the upper-layer semantics; although mostly by indirect and partially deterministic means.

How these elements are defined and combined in the final visual output will vary greatly depending upon the parameters and encodings we give them through formatting. A geometric circle is not just a circle, but one of a specific color, a specific line type and width, a specific line color, etc. There is also no geometry that covers every expressible form. A text is not just an ideal text, but must also become visible through graphic representation. Fonts and their graphemes make the idealized form of written language visible to users. Metrics that allow users to select text or place texts on screen with proportional spacing are incredibly intricate and complex. Images and time based media complicate the design terrain even further.

Just as idealized geometric logic, raster color information, or semantic textual descriptors come in multiple dimensions, images are also only as flat as their internal features. Any image is not the same thing as the idealized forms that compose the image. If we look at figure 3.2, we see a naturalistic setting with the

⁴²See Lev Manovich's *The Language of New Media* for just one of many examples



Figure 3.2: Digital photo montage. source: Wikimedia Commons

earth as moon, a statue of liberty and other iconic architectures that are out of their environment, a lake, clouds, and a Asian boatman. What we do not see, and what may or may not be recorded in the image format, are the sub-structural properties that make up the image. These properties probably include some sort of meta data that might provide an internal comment in text form about the image, including info about the author, the time it was made, the application that was used to make it. However, there could be potentially infinite other properties recorded in the file. It *could* include the aperture, exposure time, and light reading settings of the camera that took the images. It *could* also include GPS data about where the image was captured, in which direction the camera was ported, and given the lens settings, could even provide coordinates of the view-port associated with that image; what was the depth of field and how wide or narrow of field of

view. Cameras are of course changing⁴³ and will often record much of this data, if not all, in the comment meta-data of the image file, but not for a mixed image like this one. It *could* include the many number of layered images that go into making the file, each with their own alpha masks that isolate the photographed object from the background. A user could then turn visual elements on and off in the image. It *could* also include the potential number of photos that went into capturing the single image that we see on screen. For example, cameras can now capture multiple images at once or in rapid sequence, each with various capture settings for exposure time and aperture setting. A viewer can algorithmically alter these images to create panoramas, high-dynamic-range imaging, as well as varying focal lengths, positioned views, or stereographic output.⁴⁴ An image might also include all three dimensional coordinates, all the textural components, all the physical parameters of lighting, etc. that go into its visible attributes. Recording and formatting this amount of data, of course, approaches such a scale of complexity that only philosophers can debate its practicality.

⁴³Given the mixture of GPS, bluetooth, camera phones, etc, the camera is no longer the simple and innocent creature it used to be.

⁴⁴See Stanford's 2005 paper[71] and video on high performance imaging at <http://graphics.stanford.edu/papers/CameraArray/CameraArray.mp4>. Also see the most recent Kinect device for xbox, and S.K.Nayar's work on computational cameras.[46] Incidentally, he also highlights the necessity of policies of openness in this document, saying that "In order to give its end-user true flexibility, a programmable imaging system must have an open hardware and software architecture...".

While there have been attempts to develop more sophisticated image formats, the general implementation and acceptance of them has been less than satisfactory. Both MPEG-7 and quicktime VR propose new ways to capture, display, and annotate multidimensional images that come from reorganizing the assumptions on what an image is or may be. The MPEG-7 standards (1996-2004) is an attempt to articulate an expression of real-world ideal descriptors in multimedia content: <http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm>. A lot of time and research went into developing the so-called standards, but they were never implemented and popularized. They are now arguably out of date. Quicktime VR and other similar technologies also provide new ways of viewing visual data. This is not to say, however, that the evolution to more sophisticated formats is the goal. Simple formats with strict specifications, such as Twitter, also have a role to play in the WWW. The goal is an openness to new unknown formats, not just more intricate ones.

Additionally, because the visual elements of the WWW live in a digitally computational domain, they are capable of acquiring extra expressivity through artificial programming languages that drive and manipulate the parameters of these basic formats. The final sum expression on screen will greatly depend on how these representational formats are defined and how they are programmatically driven. See sect. 3.6. Also, considering the network effect inherent in a civic space

of communication, how much and where preference is given to some elements over others will influence the overall feel of the entire design space of the WWW as users adopt default formats.

The process of representing and encoding immaterial information should not be underestimated. These discursive elements both reveal and conceal through their formal articulation. The formats and methods of representation on the WWW have become so universally accepted that they are almost fully obscured by our assumption that what we are dealing with is not a constructed semantic environment but a mere tool or surface. A proper configuration of the WWW should also include enough technological leeway for new formats to appear. This is best done under a FLOSS development strategy that makes fewer written or unwritten assumptions about what formats are allowed or possible.

3.11 Rectangles

Because the original intent of the WWW was conceived and designed by engineers (with arguable amounts of aesthetic intuition) to provide a global informational system based on text, the default layout mechanisms of HTML are still blocky and rectangular, following a rather standard off-the-shelf newspaper-like layout. If you compare this with posters, CD's and other print or digital media,

the aesthetic difference is immediately and visibly perceivable. The reason for this is two-fold. The ongoing development of the WWW is adhering to HTML as the only first-class format, unnecessarily so. Also, although HTML can individually mimic any other 2D design exactly, the default behaviour of HTML that provides this rectangular, almost rigidly cubist, awareness of information is what guides the general look and feel of the WWW.

HTML has been the format of the WWW since its inception. Although other formats are allowed to sit inline with HTML, no other format can be used *in place of* HTML. SVG and XML were proposed as alternatives, but have both been encumbered with problems of complexity and adoption/implementation rate. They are also both incredibly similar to HTML, so much so that one is really just a subset of the other. However, it remains to be seen if they can be integrated as one. One claim to keep HTML as a first-class entity is backwards compatibility. However, historically speaking, HTML has never been compatible among browsers, much less backwards compatible. Furthermore, adding an additional first-class format would not exclude HTML.

The default behaviour of HTML is another reason for the rectangular appearance of the WWW. It is based on a flow layout algorithm that stacks rectangular elements from left to right and from top to bottom on screen. It has no default properties for rendering anything but rectangular shapes. It also has no algorithms

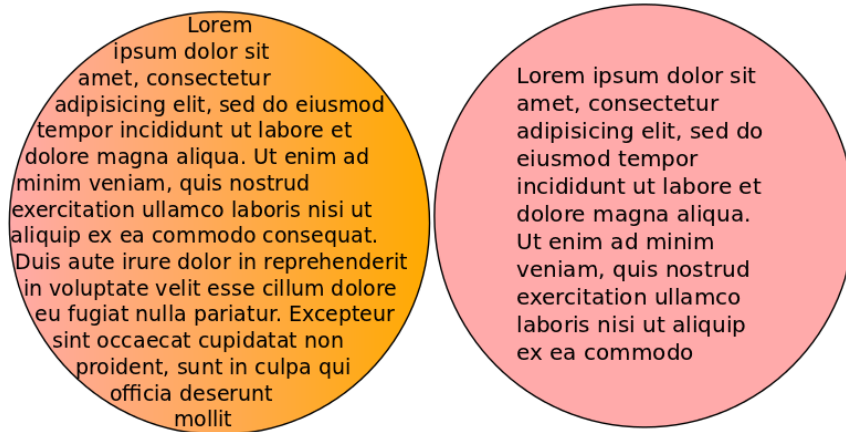


Figure 3.3: Simple graphical example of two possible visual elements.

for rendering text inside or along the path of non-rectangular forms. Indeed, a developer *may* design a page with image elements to provide odd-shaped graphical components to a page. They may also render textual information as an image. However, this layout strategy of has a number of disadvantages.

Allow me to explain with an isolated visual example. Figure 3.3 shows two possible graphical elements, each with text. HTML cannot render either of these two graphics without workarounds.

The element on the left is possible to draw in HTML, but only if the entire element is rendered as a JPG or PNG and included as an element in the HTML page. A developer could also create server-side script that renders a graphical representation of the page where each pixel is single colored <DIV> tag.

See, for example, <http://aug.ment.org/dissertation/img2div.php>. In both approaches, the text would not be a logical unit of selectable text data, but only a raster graphic. Additionally, since they both use rasterized elements, they are not scalable to variable sizes without resolution loss and/or have potential aliasing issues. The scalability of vector graphics is becoming an increasingly important design issue as screen resolutions increase and screen sizes and dimensions become more varied. The second approach would use so much memory that it would bring even a recent computer with a large amount of RAM to a stand still. The circular graphic on the right has more potential in standard HTML, although also not without work-around solutions. In addition to the two work-around methods for the left graphic, developers could render the central text part of the right side graphic as a standard `<DIV>` tag, and then attach and align the curved portions of the circle as raster images. This is indeed a popular approach by current WWW developers who would like to add non-standard corners to their text frames, but comes with similar issues of scalability.

There are also other potential work around methods to achieving non-rectangular layout of text and geometry. One might include using separate `<DIV>` tags to group and align the text manually, line by line. The point is that they remain as work-around solutions from the default functionality and demonstrate that any one format will have a specific tendency toward an unstated design goal. What-

ever this unstated goal is, as is defined by its default methods, will also heavily sway the overall look and feel of the design space of the WWW. At the moment, this look and feel is overwhelmingly rectangular.

Both of the graphics above are also possible to create and render in the SVG format, and were in fact created with the Inkscape SVG editor. Unfortunately, the SVG does not render properly in recent versions (August 2011) of either Firefox or Chromium.⁴⁵ The SVG working group has changed its recommendation for text and is still not finished. As a result, very few SVG renderers currently implement either the old or the new syntax of SVG 1.2 flowed text.⁴⁶ Both are also potentially doable with bleeding edge CSS or canvas technology. In the case of CSS, this again brings up issues of compatibility. In the case of the canvas option, it brings up issues of complexity.

Introspection of the SVG code of the two graphic elements above demonstrates the complexity involved. See the source code below in Fig. 3.4. When compared with the underlying programming methods that render these graphics to screen, it adds little to no extra simplicity. The shortcomings of SVG are also one of the contributing factors that went into the development of the <CANVAS> API. However, I believe that while the <CANVAS> API adds needed functionality to

⁴⁵I did not test on Explorer or Opera, but imagine they have similar problems.

⁴⁶See http://wiki.inkscape.org/wiki/index.php/FAQ#What_about_flowed_text.3F for a brief summary of the problem.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- Created with Inkscape (http://www.inkscape.org/) -->
<svg id="svg2" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns="http://www.w3.org/2000/svg"
height="450" width="800" version="1.1" xmlns:cc="http://creativecommons.org/ns#" xmlns:xlink="http://
www.w3.org/1999/xlink" xmlns:dc="http://purl.org/dc/elements/1.1/">
  <defs id="defs4">
    <linearGradient id="linearGradient5301" y2="293.79" gradientUnits="userSpaceOnUse" x2="383.36" y1="293.79"
x1="108.07">
      <stop id="stop5271" stop-color="#FAA" offset="0"/>
      <stop id="stop5273" stop-color="#FA0" offset="1"/>
    </linearGradient>
  </defs>
  <g id="layer1" transform="translate(0,-602.36204)">
    <path id="path2985" d="m382.86,293.79a137.14,135.71,0,1,1,-274.29,0,137.14,135.71,0,1,1,274.29,0z"
transform="matrix(1.3863175,0,0,1.4075631,-130.23929,420.35123)" stroke="#000" fill="url(#linearGradient5301)"/
>
    <path id="path2999" d="m382.86,293.79a137.14,135.71,0,1,1,-274.29,0,137.14,135.71,0,1,1,274.29,0z"
transform="matrix(1.3863175,0,0,1.4075631,254.37489,413.66786)" stroke="#000" fill="#FAA"/>
    <flowRoot id="flowRoot5311" style="letter-spacing:0px;word-spacing:0px;" font-weight="normal"
xml:space="preserve" font-size="40px" font-style="normal" font-family="Sans" line-height="125%"
fill="#000000"><flowRegion id="flowRegion5313"><use id="use5315" y="0" x="0" xlink:href="#path2985"/></
flowRegion><flowPara id="flowPara5317" font-size="18.98703003px">Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in
voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa
qui officia deserunt mollit anim id est laborum.</flowPara></flowRoot>
    <flowRoot id="flowRoot5332" style="letter-spacing:0px;word-spacing:0px;" font-weight="normal"
xml:space="preserve" font-size="40px" line-height="125%" font-style="normal" font-family="Sans"
fill="black"><flowRegion id="flowRegion5334"><rect id="rect5336" y="-140.36" width="258.42" x="392.09"
height="233.92"/></flowRegion><flowPara id="flowPara5338"/></flowRoot>
    <flowRoot id="flowRoot5340" style="letter-spacing:0px;word-spacing:0px;" line-height="125%" font-
weight="normal" xml:space="preserve" font-size="40px" transform="translate(53.467001,867.46925)" font-
style="normal" font-family="Sans" fill="#000000"><flowRegion id="flowRegion5342"><rect id="rect5344"
y="-180.47" width="227.23" x="425.51" height="282.93"/></flowRegion><flowPara id="flowPara5346" font-
size="20px">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit </flowPara></
flowRoot>
  </g>
</svg>
```

Figure 3.4: SVG source code for Fig. 3.3.

draw graphics and animations, it still comes only as an addition to the first-class HTML format. It is also so close to the underlying C API that renders graphics to screen - in many cases this is simply the Cairo graphics API - that it would make more sense just to expose these underlying APIs.

If developers eliminate HTML as the first-class format of the WWW (upon which CSS and Javascript live as second-class entities), and instead replace it with

a slim software infrastructure that could potentially support multiple first-class formats and programming languages, the outwardly visible expressive features of the WWW would be much more diverse and interesting. This is what I propose in my Underweb browser.

Chapter 4

Methodology

The problem space of the World Wide Web is essentially a trade-off of various interrelated and interdependent aspects of digital communication using technological gadgetry. Traditionally, many of these problems are often isolated into individual sub-sets and quantitatively studied under the guiding framework of engineering or computer science. However, taken as an interrelated whole, the problem space of the WWW is specifically not a quantitative issue, but rather a qualitative mix of political, aesthetic, and technical issues for which an artistic intervention is better suited. My methodology is based on the rich 20th and 21st century traditions of artistic development based on critical theory.

This project is deeply rooted in the theory and technicalities of digital networking and computational aesthetics, where plausible functionalities are greatly decided by technical specifications for how various kinds of collectives and individuals exchange media elements through networks. As such, a truly interdisciplinary

approach is necessary to recognize aesthetically insufficient techno-political specifications and work to build a larger possible design, communication, and computational space. As a method of investigation and analysis, this experimental research focuses on the blurry cross-section of theoretical application and practical code, drawing connections between design, communication, networking, systems engineering, and critical humanistic analysis. The instruments of this research are the gcc and valac compilers, available free and open-source software libraries, and networked computers. The procedure is cyclical iterations of programming, debugging, release, and analysis.

My interdisciplinary role in this dissertation is as artist and technologist. Just as psychologists study the human psyche in the field of psychology and philosophers write in the area of wisdom, technologists study the field of technology - a discipline and branch of knowledge which examines artificial tools, systems, and their many functions. Technologists study the entire spectrum of media and its associated ways of taking form. They are particularly interested in how software - the immaterial stuff that gives dynamic form and agency to contemporary digital media - shapes the world in various abstract and concrete levels. While my methodology, including tedious hours spent writing and understanding software, appears like computer science or computer engineering, it differs in profound yet subtle ways. Unlike the engineer, I am not particularly interested in the speed

or efficiency of technical apparatus, but more in the affective interdependence and co-evolving features of man and machine. Speed and efficiency are only interesting technical problems if they contribute to how they spark interest, inform social patterns, and affect individual or collective behaviour. As these elusive quantitative components are not necessarily measurable entities, my research here is the process of building and making itself, and thus closer to the functional aspect of art.

Chapter 5

The Underweb

In this chapter, I discuss the impetus and decisions that went into the making of the practical component of this dissertation. The practical component comes in the form of a software framework that I call the Underweb. It is an experimental and prototypical browser that I produced using only free software. My intent with this development is to offer a new technological frame and framework through which to experience networked, communication, and informational spaces. In many ways, it is an anti-framework in that it aims to produce a very thin pseudo-layer that makes virtually all frameworks possible within it. I discuss how this browser could potentially create a new technological space of user interaction and what implications I think this has for new user experiences. The Underweb framework provides partial solutions to the problems of the WWW that I discussed in Sect. 3.

5.1 Technical Development of the Underweb

The way the Underweb aims to deliver proposed solutions to the problem space presented in this dissertation is through an *empty* frame and open framework. I say it is *empty* because it is built on already-existing technology and is meant to provide only a thin open layer for users to fill with their own functionality. In conjunction with that, I provide a very simple scene-graph library for drawing container shapes, time-based media, and text on screen in a manner in line with, but alternative to the way the current WWW lays out a page. The scene-graph library mixed with other networking options forms the libmentiras internal library.

The development of this prototype has gone through three main stages. I discuss the first two only briefly and then move on to the particulars of my current prototype.

This original research has evolved over the two plus years since I formally started putting my ideas together.¹ My very first developments started with building a scene graph for webpage-like layout. Starting in C++, I used both the Antigrain software drawing library[54] and the V8 Javascript engine of Google's Chrome browser[32] to build a test browser that could draw anti-aliased vector graphics on screen using only Javascript as its driving language. Feeling unsatisfied with the isolated complexity of these libraries, I started to look for software

¹The ever-changing WWW has also evolved much in those two years.

components that were better integrated and provided more solutions for everyday low-level computing problems. I then started to de-construct other existing browsers and look at their parts. From there, I turned to the spidermonkey Javascript engine[26] and the Cairo[44] and Pango[60] drawing libraries that are used in the Mozilla browser project. Additionally, I used the FreeImage[25] and Gmerlin multimedia libraries[50] for image decoding and audio-video playback, respectively. Sticking with C++ as my main language, with these libraries I found more inter-compatibility and simplicity of programming than in the previous stage. Although, Antigrain did provide a much more flexible API and perhaps even higher-quality rendering at the time, I needed to drop it for the sake of integration. Switching to these libraries, I was able to build a prototype that could render shapes as well as text layouts on screen, partially support both client and server TCP sockets, play many forms of audio and video in sync, and provide a simple synthesis example in audio. Furthermore, during this development process I was able to make a number of contributions to the Gmerlin and Puredata projects in the form of heavy debugging and original software. All of this development was still based on exposing APIs to only a single scripting language, Javascript. However, this second-stage prototype was growing in complexity and my research was taking me in another direction that focused on infrastructural issues as well as aesthetic, networking, and multimedia issues. These libraries

lacked the lower-level infrastructural capabilities I was looking for such as multilingual API binding support, dynamic plugin support, garbage collection for memory management, event timing etc. I therefore shifted my focus on embedding my work in the much larger free software toolkit schema of GTK+ and the object-oriented language, Vala.

This current Underweb framework is written in the Vala programming language and uses the many GTK+ technologies. The techniques provided by GTK+ such as GLib, Pango, Cairo, and GObject form the basis for a huge section of free software development. From the website: “GTK+ is a highly usable, feature rich toolkit for creating graphical user interfaces which boasts cross platform compatibility and an easy to use API.”[61] It is undoubtedly one of the most popular XWindow based² user interface technologies of the free software world, only (softly) competing with KDE’s QT framework.³ It provides the underlying functionality and methodology for a large part of the desktop and user interface, and commands a huge community of producers and users. Both the WebKit and Mozilla HTML rendering libraries use some portion of the GTK+ project. GTK+ also represents an ungodly number of man-hours of work that are readily

²Originating from MIT in 1984, the X Windows system is a hardware abstraction layer and network protocol that provides device independence to developers who write graphical user interfaces for networked computers. It is currently maintained by the X.org foundation at <http://freedesktop.org>

³While KDE is written in C++, and GTK+ is written in C, since they are both free software, there are other layers of compatibility that allow users to run KDE apps within GTK+ and vice versa. Competition in the free software world is very different than in the proprietary domain.

and freely available to me or any other developer. The advantages of placing my work within this larger context can be outlined as follows.

GTK+ consists of software sources for user interfaces and a rich community of technical expertise. The community aspect of the GTK+ environment provides me access to the knowledge that went into building the software through IRC forums, email groups, and the openly readable software. Other free software browsers such as Mozilla take advantage of and contribute to this as well. Additionally, because many functions and methodologies have been tried and tested within this community, I do not have to re-invent the wheel for basic technical stuff like internationalization of text, character encoding translations, bidirectional texts, low-level networking issues, abstract data types such as lists and maps, etc. There are other more technical advantages, too many to report in this section. I will highlight only a few starting with libcairo and libpango.

Cairo is a cross-platform 2D graphics library with support for multiple output devices that include the X Window System, Quartz, Win32, image buffers, PostScript, PDF, and SVG files. “Cairo is designed to produce consistent output on all output media while taking advantage of display hardware acceleration when available.”[44] Written in C, but with bindings to many languages, libcairo provides an API to perform basic low-level drawing operations such as lines, curves,

bezier curves, affine transformations, compositing, and anti-aliasing with sub-pixel accuracy. Cairo is tightly integrated with libpango.

Pango is a software library for the layout and rendering of text that provides the core font and text capabilities of GTK+. Its emphasis is on modularity and internationalization, consisting of dynamically loaded modules that handle text layout for particular combinations of scripts and font back-ends. “Virtually all of the world’s major scripts are supported.”[60] Pango handles the many difficulties of rendering multiple and intermixed foreign glyphs to screen. In combination with GTK+ and libcairo, it provides a tightly integrated and elegant solution for many problems of user interface design.

GTK+ comes with a combination of other useful technologies upon which the Underweb browser depends: GLib, GObject, and GObject introspection. GLib is a general-purpose cross-platform software utility library that, in some ways, provides an alternative to the “C++ and STL”⁴ combo in C. It provides many useful data types, macros, type conversions, regular expressions, string utilities, file utilities, threading, messaging system, a main loop abstraction, reference based garbage collection, etc. It is based on the GObject system.

GObject, and its lower-level type system, GType, are used by GTK+ to provide a portable object-oriented C-based API. It was also developed with the fore-

⁴Many developers use C++ and the Standard Template Library (STL) to solve generic programming problems.

thought of accommodating automatic transparent API bindings to other compiled or interpreted languages. Like C++ and Objective-C, it provides a complete object system to the base C language, but without the introduction of any new syntax or compiler intelligence to C itself. It is strictly a separate library. Additionally, like Java, it does not support multiple inheritance. Furthermore, its emphasis on messaging provides the ability to easily encapsulate software components for reuse. GObject was specifically designed to meet the needs of a GUI toolkit with cross-language interoperability. To that end, the additional GObject introspection library smoothly facilitates *lazy* cross-language bindings, a budding necessity of contemporary software development.

A current common practice among software developers is to program their applications in at least two languages and on two levels. They often write mission critical tasks in C (or C++) and bind those functions to some dynamically-typed language with a managed runtime for non-crucial application logic. The low-level and strongly-typed language provides speed and efficiency at the cost of more verbose and difficult semantics. The high-level dynamically-typed language, such as Lua, Ecmascript or Python, provide a more concise and easier language for writing logic for configurations, layouts, dialogues, etc. GObject introspection[62] provides developers of GObject based software an easy-to-use infrastructure in which to bind these other languages. For example, GObject introspection allows

a developer to bind the low-level C functions to both the Ecascript and Python languages without any extra programming or design.

There are two other critical components that form the software makeup of the Underweb browser: libpeas[49] and Gstreamer[45]. Libpeas is a GObject-based plugins engine that provides extensibility to the Underweb browser. Where current WWW browsers load *pages*, the Underweb considers each loadable item to be an application; or, what was previously known as a plug-in. These *applications* are fully functional programs that may also display items to screen in a hypertext-like way. With libpeas and GObject introspection, WWW developers can program applications for the Underweb browser in C, Vala, Javascript, and Python. C and Vala bindings are provided directly. Javascript and Python bindings are provided on the fly through GObject introspection and the Seed[64] and PyGObject[63] projects, respectively. Other languages are also possible, but not currently supported.

Gstreamer is a very extensive pipeline-based multimedia framework based on GLib and GObject. “The applications it supports range from simple Ogg/Vorbis playback, audio/video streaming to complex audio (mixing) and video (non-linear editing) processing.”[45] Formerly, I had used the Gmerlin toolchain for multimedia encoding and decoding. It is a much simpler and stronger API written with a basic C interface. Gstreamer is much more complex and involved requiring much

more verbose code. However, its flexibility allows for a broader set of uses. It also provides much easier and better integration with the GTK+ and Vala environment in which I am now developing the Underweb.

5.2 The Underweb Framework

The name Underweb comes from two things: it provides direct access to the underlying APIs of current browsers as well as the ability to underlay (as well as overlay) the application layer of HTTP. It was written with *openness* in mind: to both maximize the *openness* of user experience on the WWW and under the considerations of free software principles. It is admittedly only a personal project at this point in time, one that stems from my own individual and artistic assumptions about what the user experience on the WWW could potentially be. This artistically informed ideology is also heavily laden with technical intuition that comes from extensive years of programming software applications within collaborative free software methodologies.

At the moment, the Underweb framework provides a simple *frame* with a GTK+ window, a location bar and a tabbed sub-window. See Fig. 5.1. Like with other browsers, users type in the URI⁵ location of an online or offline application document, upon which the Underweb browser may then display and run the re-

⁵Uniform resource identifier. Also known as URL, uniform resource locator.

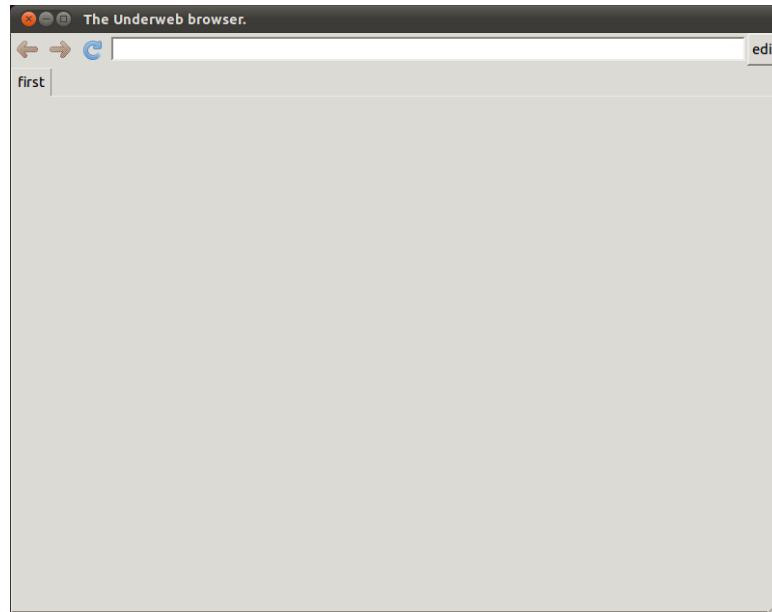


Figure 5.1: The Underweb without any loaded content showing a location bar, tabbed frame, and an edit button.

trieved software. I call the viewable data of the Underweb application documents because they are neither full-on software applications or static documents. They are a mixture of both. I explain this below in the libmentiras section. There is also an edit button that allows users to create and edit these application documents in the multilingual libmentiras format.

All application documents for the current Underweb are what would be traditionally known as a plugin. Based on the filename ending, either .py, .js, .c, or .vala, the Underweb browser loads the document and introduces it to the memory space it manages. The .py and .js files are interpreted by Python and the libseed Javascript engine, respectively. With .c and .vala application documents, the Un-

derweb performs an extra step to compile and load the shared object files using the gcc and valac compilers.

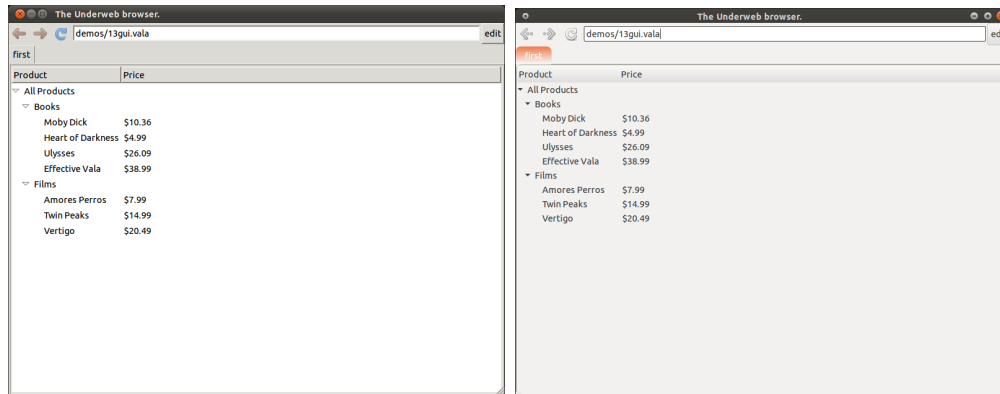


Figure 5.2: The Underweb browser running a native GTK+ user interface example. The two windows are running the same application document, but under a different desktop theme.

The Underweb makes few assumptions about the kind of application documents a user may want to produce. It also has no limits on the various free software APIs that are potentially usable by the developer. For example, a developer is able to produce applications that make use of any available GObject library, of which there are many professional-grade implementations of multimedia interfaces (gstreamer, cairo, opengl⁶, sdl⁷, poppler⁸) databases (couchdb⁹,

⁶<http://www.opengl.org/> OpenGL is the industry standard for hardware rendered graphics.

⁷<http://www.libsdl.org/> Simple DirectMedia Layer is a cross-platform multimedia library designed to provide low level access to audio, keyboard, mouse, joystick, 3D hardware via OpenGL, and 2D video framebuffer.

⁸<http://poppler.freedesktop.org/> Poppler is a PDF rendering library.

⁹<http://couchdb.apache.org/> CouchDB is a document-oriented database that can be queried and indexed in a MapReduce fashion using JavaScript.

sqlite¹⁰, postgres¹¹), graphical toolkits (GTK+, Clutter/Mx¹²), networking(lib-soup¹³, GIO¹⁴, d-bus¹⁵), etc. See Fig. 5.2 for an example that loads GTK+ widgets directly into the browser. Because GTK+'s theming engine is fully programmable with CSS styling, the interface can mimic and tightly integrate with the graphical look and feel of the user's preferred operating system. A developer is also not bound to use any external library, but with the ability of writing low-level C code can instead write computing intensive applications directly. The Underweb browser can in fact load, compile, and run iterations of itself in a truly autopoietic manner.

The Underweb exposes to the developer an object oriented interface for writing application documents. Each application document is essentially a software object as would be understood in Javascript, Python, and Vala. Besides object ini-

¹⁰<http://www.sqlite.org/> SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.

¹¹<http://www.postgresql.org/> PostgreSQL is a powerful, open source object-relational database system.

¹²<http://www.clutter-project.org/> Clutter is an open source (LGPL 2.1) software library for creating fast, compelling, portable, and dynamic graphical user interfaces. It is often used for mobile interfaces. Mx is the widget toolkit it uses.

¹³<http://live.gnome.org/LibSoup> libsoup is an HTTP client/server library for GNOME. It uses GObject and the glib main loop, to integrate well with GNOME applications, and also has a synchronous API, for use in threaded applications.

¹⁴<http://developer.gnome.org/gio/2.28/> GIO is striving to provide a modern, easy-to-use Virtual File System API that sits at the right level in the library stack. It has a very well developed networking API.

¹⁵<http://www.freedesktop.org/wiki/Software/dbus> D-Bus is a message bus system, a simple way for applications to talk to one another. In addition to interprocess communication, D-Bus helps coordinate process lifecycle; it makes it simple and reliable to code a *single instance* application or daemon, and to launch applications and daemons on demand when their services are needed.

tialization which happens when the file is loaded by the Underweb browser, this object exposes hooks for two public functions that are called when the object is activated or deactivated. The bare minimum of boilerplate code that is necessary for an Underweb application document is shown below. The syntax is different, although very similar, for Javascript, Python, and Vala.

javascript boilerplate code

```
__UWEB__OBJ__= {
  activate: function() {
    print("activate");
  },
  deactivate: function() {
    print("deactivate");
  }
};
extensions = {
  'MentirasMyActivatable': __UWEB__OBJ__,
};
```

python boilerplate code

```
class __UWEB__OBJ__(gobject.GObject, Mentiras.MyActivatable):
  __gtype_name__ = '__UWEB__OBJ__'
  window = gobject.property(type=gobject.GObject)
  def do_activate(self):
    print "activate";

  def do_deactivate(self):
    print "deactivate";
```

vala boilerplate code

```
[CCode (cname = "G_MODULE_EXPORT peas_register_types")]
public void peas_register_types( Peas.ObjectModule module) {
    module.register_extension_type (
        typeof(Mentiras.MyActivatable), typeof(__UWEB__OBJ__));
}

public class __UWEB__OBJ__ : GLib.Object, Mentiras.MyActivatable {
    public ScriptableWindow window {get; set;}
    public __UWEB__OBJ__() { }
    public void activate () {
        print("activate\n");
    }
    public void deactivate () {
        print("deactivate\n");
    }
}
```

To develop for the Underweb, one would simply fill out the `activate` and `deactivate` functions with code that should be called when the application document is loaded or unloaded, respectively. Additionally, it would also be possible to provide users with the ability to stop and start a running application without unloading it. The `__UWEB__OBJ__` directive is a pre-parser substitution variable. When the Underweb retrieves an application document, the pre-parser then replaces all `__UWEB__OBJ__` directives with a SHA1¹⁶ encoded hash digest based on the application document's full URI. This ensures that, internally to the browser, there are no namespace collisions between any two application documents.

¹⁶SHA stands for secure hash algorithm. It is an algorithm designed by the National Security Agency and published by the NIST as a U.S. Federal Information Processing Standard.

The `window` object is available to all Underweb application documents in every language, and gives the developer access to the parent GTK+ widget. With it, a developer can add any GTK+ graphical widget to the display. At the moment, it only has one property, the `title`, with which you can set the display label of the current tabbed window.

One very useful library that the Underweb automatically hooks into is the free software WebKit engine that can load and render HTML content, including the latest HTML5 specified content. Webkit includes Glib and GObject bindings as a measure to improve speed in accessing the document object model of an HTML page.¹⁷ This also allows the Underweb browser to seamlessly include regular HTML content inside of its visual frame, essentially making the current WWW a subset of the Underweb communication space. Figure 5.3 shows how easy it is to do so.

Alternatively, an internal HTML renderer could be written using the Cairo, OpenGL, or libmentiras libraries. Ideally, many markup forms would be invented and used that cater to specific design goals of various groups and individuals. Additionally, developers could invent their own template formats as are already popular in server-side programming environments. This would make WWW development easier to maintain and update by enabling developers and designers to work

¹⁷See <http://trac.webkit.org/wiki/GlibBindings>

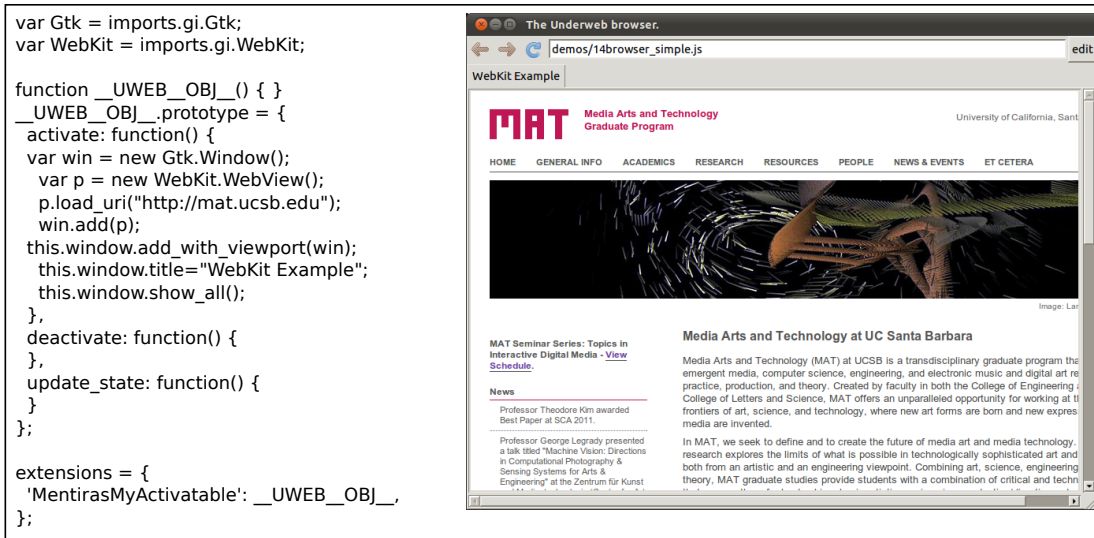


Figure 5.3: The Underweb browser running an internal WebKit rendering engine that can load *normal* WWW pages.

together easier, but without the clunky and complicated server-side templates. By placing the communicable focus on the lower level programming languages instead of the higher level markup languages, I believe designers and developers are given more leeway and incentive to invent new high-level formatting in the client browser. This could include minimal and stricter markup and template languages that are suitable for beginners.

As mentioned several times in this dissertation, the ability to not only open bi-directional communication, but also to listen on a port for requests on the WWW (act as a server) is a very crucial capability to include in a civic space of informational exchange. Other useful libraries that potentially solve problems associated

with writing and publishing on the WWW are the networking libraries, GIO and libsoup. I have wrapped these into easy-to-use components of libmentiras, and discuss them below.

Altogether, the Underweb provides a very flexible and thin pseudo-layer for potential WWW developers. It avoids the pitfalls of the current WWW by providing developers with tools that allow them to create application documents in many popular languages and using many available and professional free software libraries. Underweb developers may also develop application documents in a strongly typed language such as C or Vala for more speed and efficiency in order to perform computation greedy operations. In many ways, the Underweb framework brings functionality that was previously reserved for the base operating system into the realm of the browser, however without the constraints of the monolingistic and commercially driven development and standardization strategy of the current WWW. Furthermore, the Underweb framework can envelope and include already existing WWW content through the Webkit bindings. Additionally, I provide a library that I call libmentiras as an elegant solution for handling graphical, acoustic, time-based, and textual data in a networked environment. I turn now, to this.

5.3 Libmentiras

The Gypsies rightly contend that one is never compelled to speak the truth except in one's own language; in the enemy's language, the lie must reign.[23] - Guy Debord

Not to lie about the future is impossible and one can lie about it at will[28, p. 11]- Naum Gabo¹⁸

Libmentiras is a software library that I wrote in the Vala programming language to facilitate Underweb developers. Developers that use my Underweb framework may also decide to use the libmentiras library to write applications that need audio-video decoding and streaming, layout for texts and shapes, animation, and networking support. However, because the framework of the Underweb is open to use other libraries, this remains optional.

In libmentiras, I have focused mostly on providing what I feel is missing functionality in the current and projective HTML5 implementations for the WWW. I believe these challenge the assumptions of the guiding aesthetic and networking design principles of the WWW: the division of content and form, and the division of server and client. The combination of the Underweb framework with libmentiras and other external libraries makes few strong assertions about the relationship between form and content. At the same time, it provides the base functionality that allows the browser client to also act as a server in the network.

¹⁸As first read in Richard Barbrook's *The Californian Ideology*. [4]

To that end, I have written software components that assist the developer in drawing simple curved shapes that may contain wrapped text elements, full support for the reading and streaming (writing/publishing) of audio and video content, the support for TCP and UDP server and client sockets, a fully integrated HTTP server, functions for displaying text inline from URI, and the loading and saving of files. Additionally, I have provided basic facilities for point-and-click editing and manipulating the shapes and textual contents in a libmentiras scene-graph layout of shapes.

The syntax for creating these software items are very similar, but will vary according to language and the language binding. For example, accessing member variables of a GObject in Python requires the use of a `props` definition. To set the stroke width of a shape object of libmentiras in Vala or Javascript, one simply sets the member property directly: `shape.stroke_width=1.0;`. However, in Python, one must set an intermediating `props` member, like so: `shape.props.stroke_width =1.0;`. The access of member variables and calling of member functions is, however, uniform throughout each individual language. My examples are provided in the Vala language, except where otherwise noted. For the sake of brevity, I also leave out the boilerplate code in much of the following examples, and only show the activate portion of the application document.

5.3.1 Graphics

Libmentiras uses a very simple schema for rendering graphics to screen. It consists of a layout engine (scene-graph) that aggregates shapes in a doubly-linked list. Everything that displays text or graphics, including time-based media such as video, is derived from a shape object. A layout is created like so:

```
var layout = new LayoutEngine();
window.add_layout(layout);
```

With a layout, developers may create and add shapes like so:

```
var layout = new LayoutEngine();
window.add_layout(layout);
window.title= "Image Example";
layout.background_color={1,0,0,1};
var s = new Shape();
layout.append(s);
```

In addition to holding and rendering shapes, the `LayoutEngine` object also has a settable property for background color as a four member array of double precision floating point numbers. A developer may also attach callback functions to mouse events and start a default timer that checks regularly for shapes that are in need of updating. I discuss these below in Sections 5.3.3 and 5.3.4.

Appending a shape to the layout puts the shape on the end of a list of shapes. When rendering is needed due to a change in any of the visible structure of the shapes, the shapes are drawn to screen in order, rendering new shapes on top of the previous shape.

The shapes are designed be as simple as possible. Shapes consist of paths and their respective points, text, and a 2D matrix. The path may be closed or open. It may also be filled and stroked with color. The syntax will be very familiar to anyone who has done graphics development in other platforms. In fact, it only adds a thin layer of functionality on top of the libcairo graphics engine. The following creates a simple shape with a red fill, a blue line stroke with a stroke width of 7.3 pixels, the output of which can be seen in Fig. 5.4.

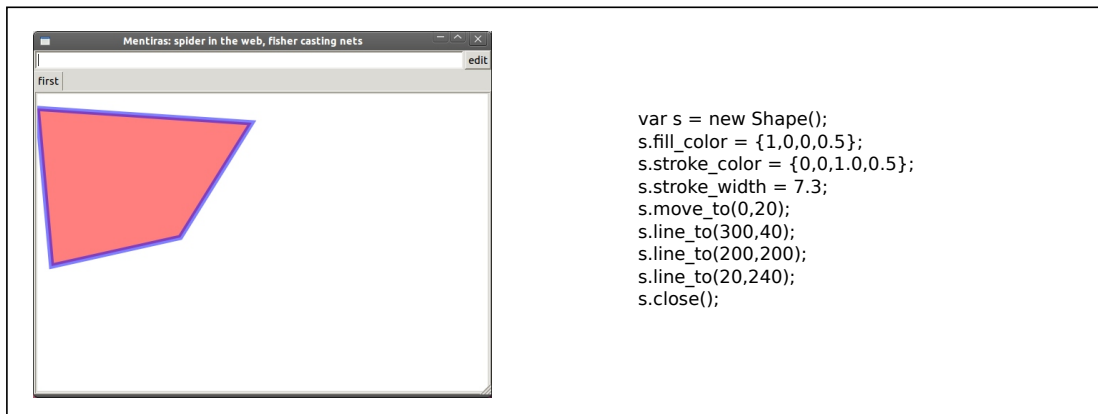


Figure 5.4: Simple graphic example showing linear edges.

Colors are defined as an array of 4 double precision floating point values representing each red, green, blue and alpha transparency. The vector path of the

shape is determined by drawing commands such as `move_to` and `line_to`. The positions of points are given in Cartesian coordinates starting at the top left as (0,0) and going to the bottom right. Bezier curves may also be drawn using three points. Fig. 5.5 demonstrates how to create a simple shape with curved edges.

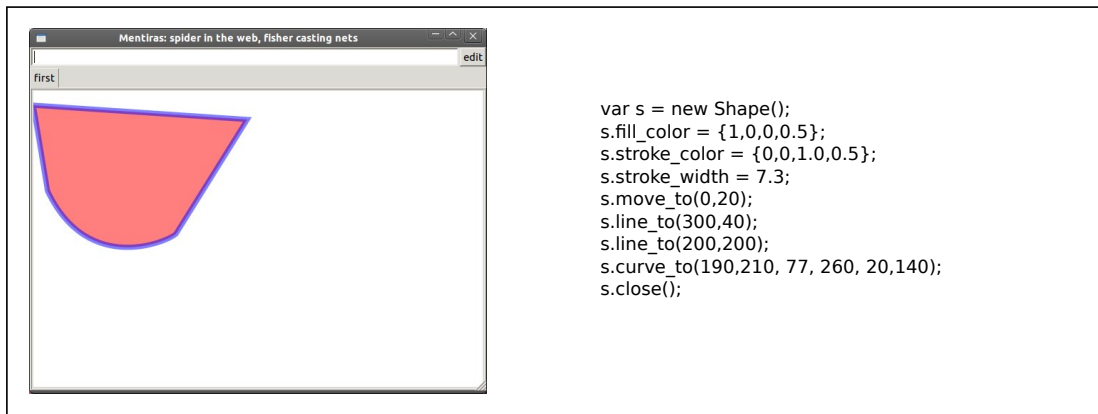


Figure 5.5: Simple graphic example showing curved edges.

5.3.2 Text

Shapes also include optional textual components for formatting. A developer may choose to have plain text or one that includes a bit of markup. She may also choose to word wrap a text inside of the contour of the shape or within a rectangular bounding box. The `text_type` property of the shape object will set text rendering to accept either `MARKUP` or `PLAIN` text. The markup is defined by the pango library.¹⁹ Fig. 5.6 shows an application document with three shapes,

¹⁹See <http://developer.gnome.org/pango/stable/PangoMarkupFormat.html>.

red, green and blue, each with an alpha component. The order in which they are appended to the layout determines which shape is blended on top of the previous one.

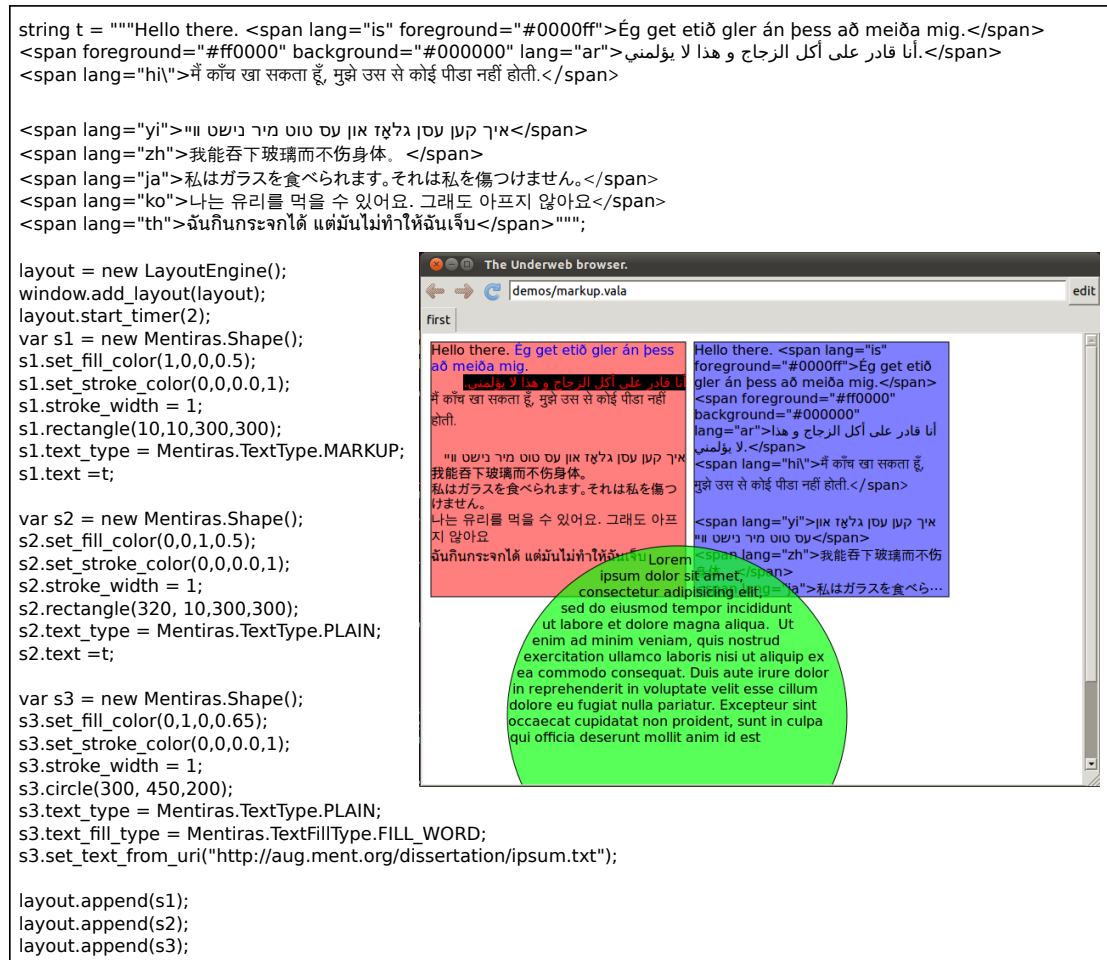


Figure 5.6: Red shape showing markup text. Blue demonstrating the same markup as plain text. Green shape demonstrating word wrapping inside a curved shape.

The `text_fill_type` property of the shape object will word wrap the text inside of the shape (`FILL_WORD`) or inside of the shape's rectangular bounding box

(BOX). Internationalization of text is also handled by libmentiras through the pango API with proper rendering of right-to-left and left-to-right scripts.

Figure 5.6 also shows the green shape loading a text through an HTTP call to an external server. Unlike the current WWW, where text is encumbered by same-origin policy security restrictions, a libmentiras shape can safely load a text from an outside source without interpreting it as executable code. Because the `set_text_from_uri` function can only load text into memory and does not interpret it as code in any way, this keeps the browser safe from this kind of cross-site scripting attacks.

The `font_name` of a shape's text may also be explicitly defined, if not done so in markup. The `font_name` property accepts a string value in the form of "[family-list] [style-options] [size]", where family-list is a comma separated list of font families, style-options is a whitespace separated list of words where each word describes one of style, variant, weight, stretch, or gravity. Size is a decimal number (size in points) or optionally followed by the unit modifier "px" for absolute size. Any one of the options may be absent.

The code snippet in Fig. 5.7 shows an example with three shapes, each with text wrapping. It also introduces the concept of a two dimensional matrix. The `Matrix2D` object in this example creates a new matrix that can be attached to the `shape_matrix` member of a shape. Additionally, one may attach it to the

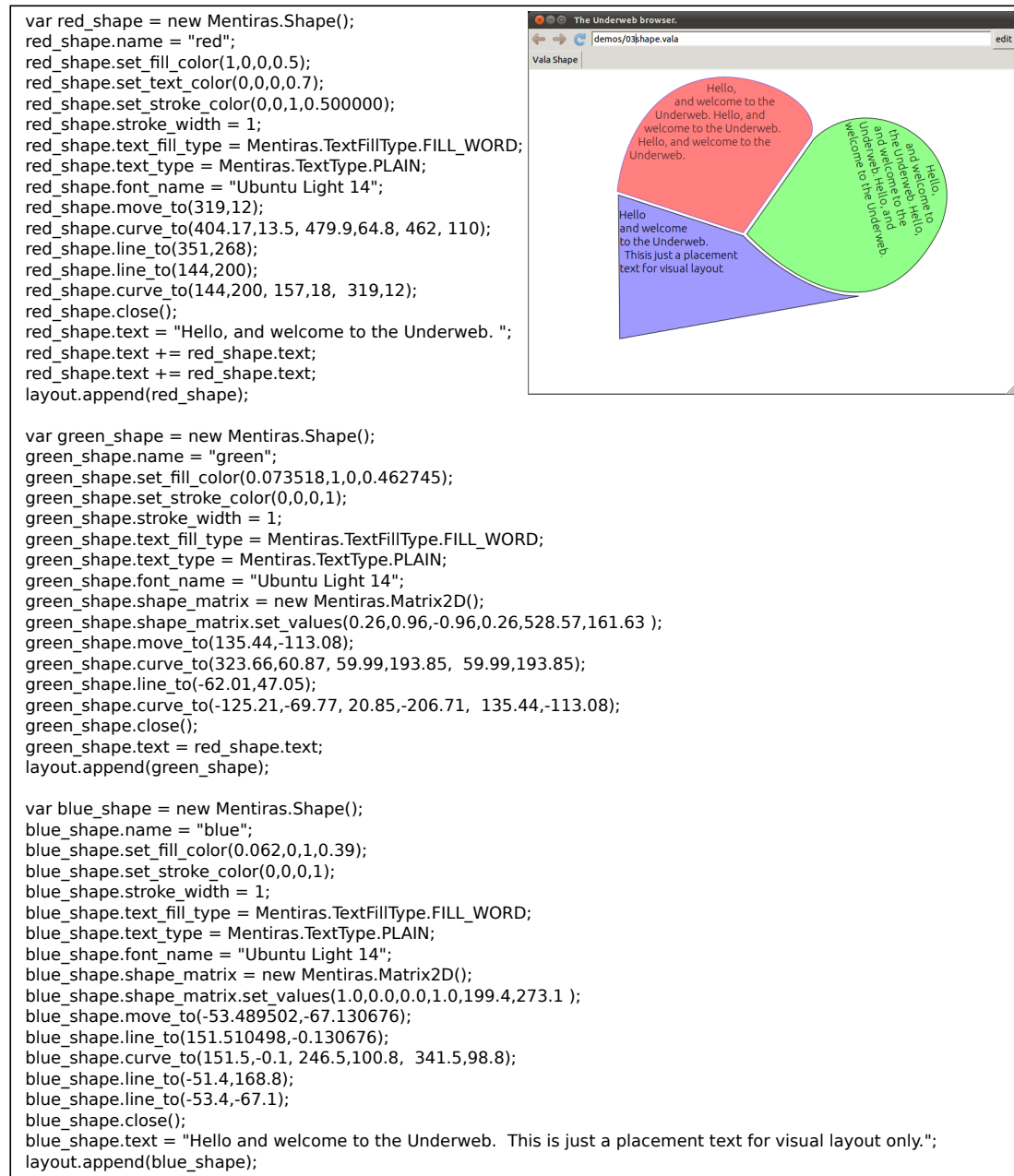


Figure 5.7: Example showing matrix rotation on word-wrapped shapes.

`source_matrix` member of a shape. This, however, is only useful for shapes that have an extra surface, such as an image, video source, or generic libcairo drawing context. I discuss this below in Sect. 5.3.5. With this matrix, a developer can change the positioning, rotation, scale, and general skewing of a shape (or source) on screen through matrix operations such as translation, rotation, and scaling. The text also follows any matrix manipulation on the `shape_matrix`. If however, the developer wishes to transform the points of the shape directly, without matrix operations and without manipulating the text, she may also perform `translate`, `rotate`, and `scale` directly on the shape.

5.3.3 Interactivity and mouse events

Both the `LayoutEngine` and the appended shapes have event handlers for mouse input. The `LayoutEngine` has only three handlers for mouse move, mouse click down, and mouse click up. Shapes have those three plus three more for mouse over, mouse out, and mouse drag. All events happen within the main loop of the Glib event system. To attach callback functions to them, a developer must connect a Glib signal to the function.

Fig. 5.8 shows an application document with 2 shapes. When mouse events are received on the green shape, the red shape prints the coordinates of the mouse.

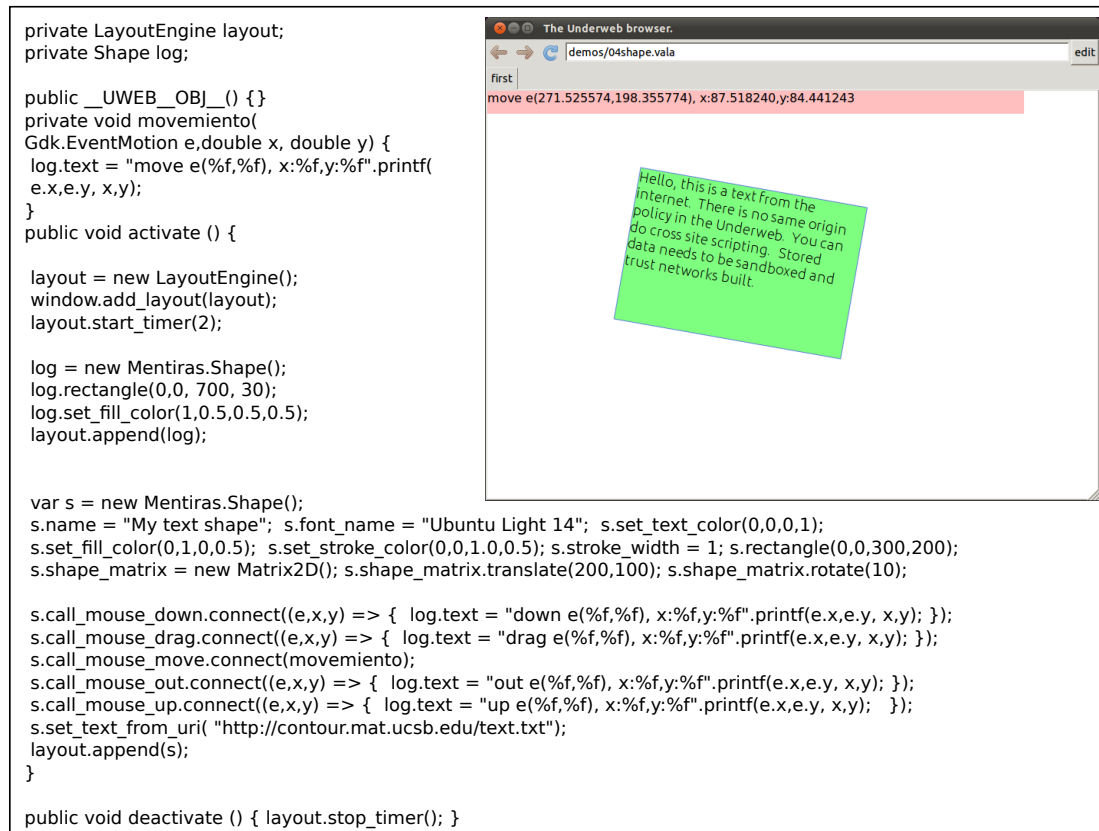


Figure 5.8: Example showing mouse event handling.

How to set the callback functions will vary according to language. With Vala, the process is straightforward. The example in 5.8 shows five anonymous, or lambda, functions attached to the mouse down, drag, move, out, and up signals.

5.3.4 Animation and timing

Timing is handled by the glib event loop. Thus, to create animations, developers may add callback functions to the even loop that are called at regular

intervals or with a timed delay. The Timeout object of the Glib system allows the developer to attach a function to the main event loop that is called with a defined timed delay. If the callback function returns true it continues to call that function at said interval. If it returns false, the timeout is taken out of the cycle.

The code in Fig. 5.9 animates a wave-like graphic to screen. The animate function is called every 10 milliseconds.

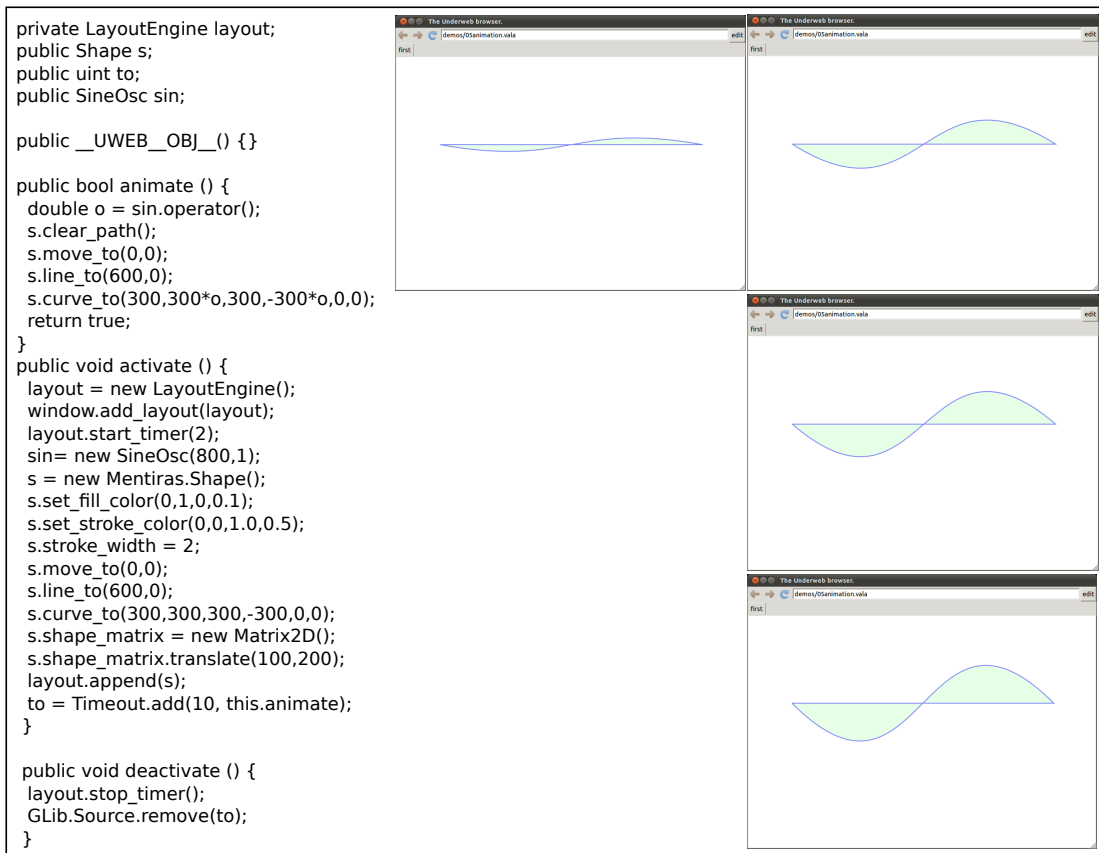


Figure 5.9: Example demonstrating timing and animation using a sine wave oscillator.

The animation code in Fig. 5.9 introduces some additional timing and event concepts of libmentiras. Each layout also has a timer function that can be started and stopped. The `start_timer` function tells the layout to start a regularly timed function that checks the scene graph for shapes that need updating. If a shape has changed in any way that requires a visual update, it then draws them to screen. The only argument it takes is an integer value designating the interval time in milliseconds. The `stop_timer` functions stops this procedure and removes the timer from the event loop. It is important to start a timer on any `LayoutEngine` that will have interactivity or time-based media. Without it, the layout will not be able to update itself automatically. Having this function as an optional feature instead of a default behaviour provides the developer with more fine tuned control over timing for resource-intensive applications.

Besides the timing functions, this code example also introduces a basic sine wave oscillator. The `SinOsc` object calculates a sine wave, and is part of a very basic audio synthesis engine internal to libmentiras. I discuss more about the audio features of libmentiras in Sect. 5.3.9.

5.3.5 Images

Shapes also include a component that enables the rendering of arbitrary pixel data inside of the shape's boundaries using a libcairo surface. Fig. 5.10 shows

two shapes, each with the same source surface created procedurally with a `Cairo.ImageSurface`. Unlike the current HTML5 canvas API, this gives the developer easy and direct access to fast pixel manipulation. The syntax for both are extremely similar. However, using the lower laying API of `libcairo` instead of HTML5's canvas API, the developer is able to bypass an extra layer of mediation. She is also able to create an HTML5 canvas-style API internally on top of `libcairo` if she so desires.

The handling of surfaces within a shape requires extra attention. How the path of the shape is positioned on screen and later modified with matrices will determine its view and placement. All surfaces are rendered to screen at position 0,0. Matrix operation on both the source and shape matrices will modify this position and view. Fig. 5.10 demonstrates this behaviour. Each shape's path was created at 0,0. On the second, right-most shape, matrix operations are given to both the `shape_matrix` and the `source_matrix`. In this example, one can observe that operations on the `source_matrix` are calculated in addition to the `shape_matrix`.

Images, like text, are a basic type in the WWW and require high-level operations to load and display them to screen. In `libmentiras`, this is handled by the `Image` object, which is directly derived from the `Shape` class. Using a `Gdk.Pixbuf` object that can read many image formats including SVG, the `Image` object loads an image from the local hard disk or from an external URI and sets the shape's

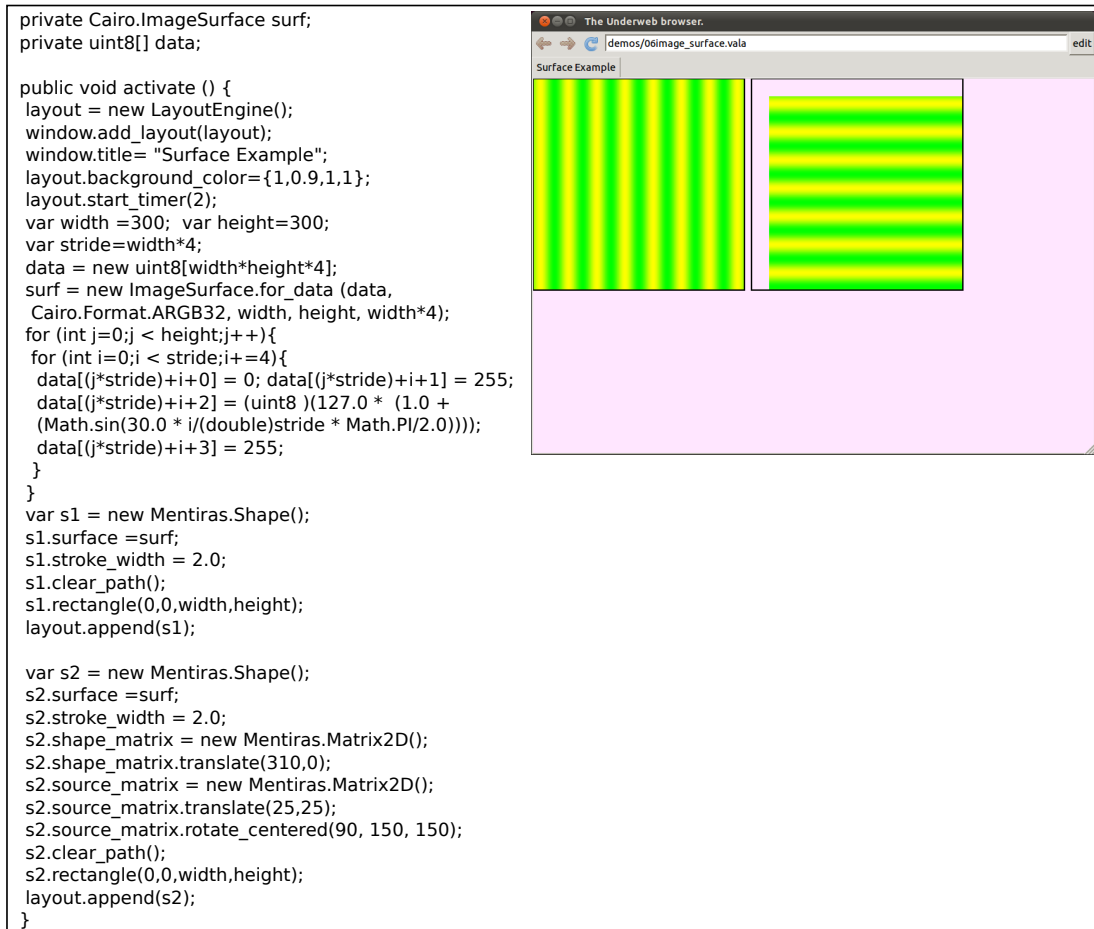


Figure 5.10: Example demonstrating surface generation and matrix operations on source surfaces.

path structure to be a rectangle of the same size as the loaded image. The path is drawn as a rectangle with its top-left-most point at 0,0. Translating the shape matrix on the image will move the image in the respective direction.

The example code in Fig. 5.11 shows image handling with 4 images. The first image loads a file from a URI and places it at 0,0 on screen. The second image



Figure 5.11: Example demonstrating image loading and display.

is loaded from the local hard disk and placed to the right of the first. The third image is loaded from an SVG file. The fourth image in the example demonstrates how to encapsulate an image within an irregular shape.

5.3.6 Custom drawing

A developer may also use a shape to implement her own custom drawing routines. Fig. 5.12 shows how to subclass the `Shape` object and attach a custom draw routine to the `draw_cb` signal of the `Shape` object. This callback function exposes the cairo drawing context to the developer, allowing her to perform any libcairo drawing routine directly.

In the example provided by Fig. 5.12, the `myDraw` function of the `Blah` object is called every 13 milliseconds as defined by the `start_timer`.

5.3.7 Widgets

The lack of widgets in the WWW has always been a sore spot among developers. While HTML provides simple widgets such as text input and buttons, it has always lacked more intricate interactive widgets such as a slider.²⁰ Developers often created their own widgets with a mixture of Javascript and `<div>` tags. The Underweb browser gives the developer direct access to broad range of lower-level GTK+ widgets, the presentation style of which is determined by system-wide settings and thus fully integrated into the look and feel of the user's desktop. The

²⁰A slider widget is in the HTML5 standards recommendation, but is not yet functional on the Mozilla 6 browser (August 2011).

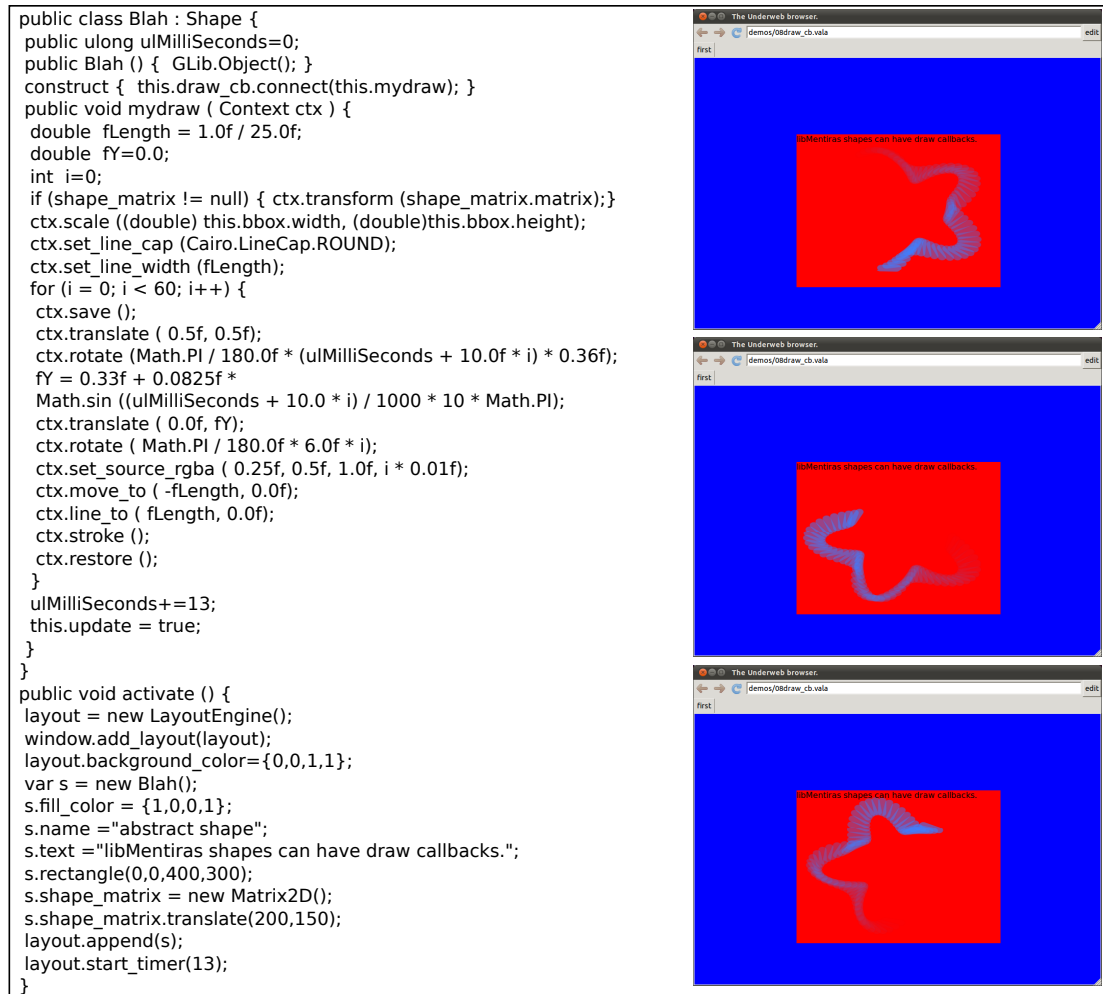


Figure 5.12: Custom drawing example with a callback using libcairo directly.

developer may also place these widgets directly inside of a libmentras `LayoutEngine` as shown in Fig. 5.13.

For the example in Fig. 5.13, I create two new GTK+ widgets that I custom designed and the code for which is not shown, called `ShapeRGBA` and `LayoutRGBA`. Each control the red, green, blue, and alpha color components of the shape on

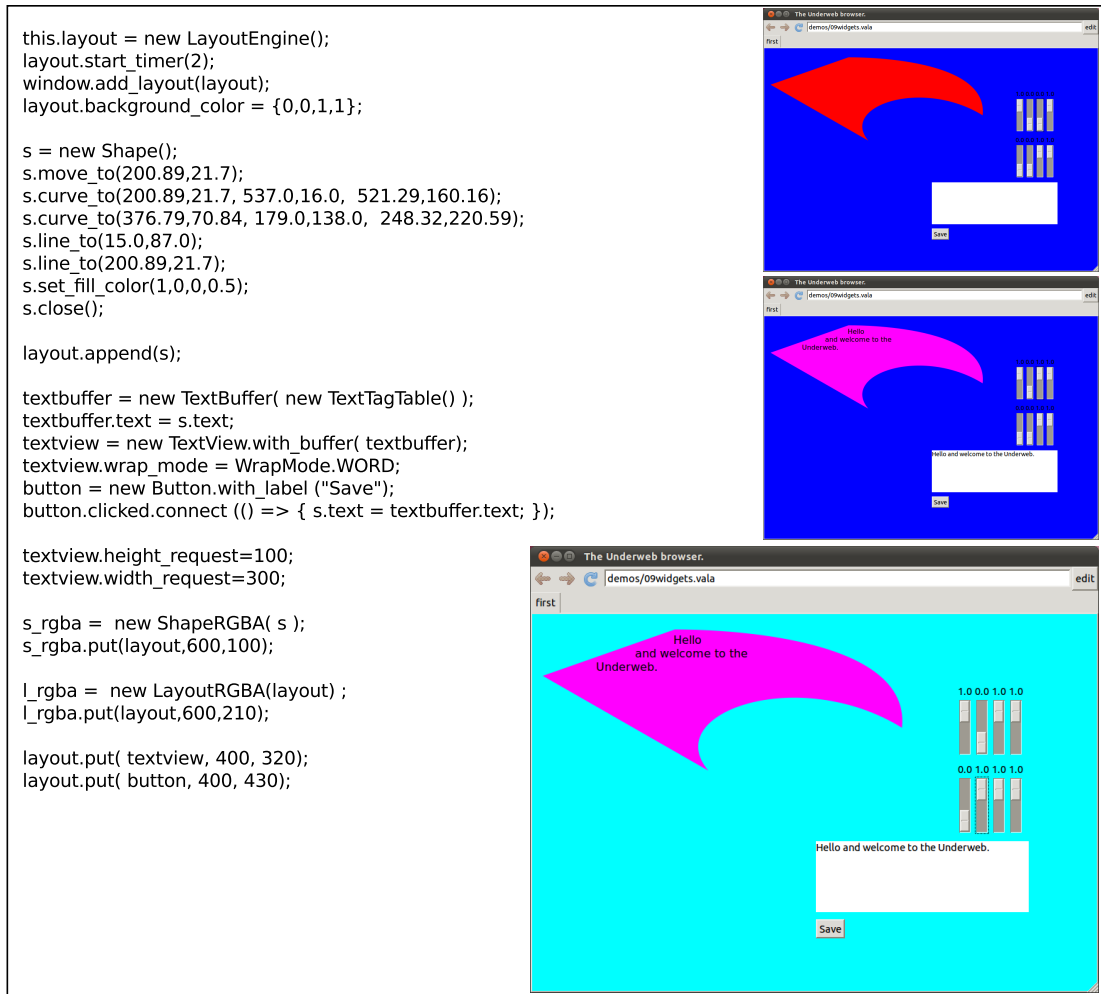


Figure 5.13: GTK+ widgets inside of a libmentiras portion of an Underweb application document.

screen and the layout background, respectively. Additionally, I create a text box and button. When text is entered into the text box and the user clicks the *Save* button, it sets the shape's text. All widgets are placed within the layout using the `put` method of the `LayoutEngine` object.

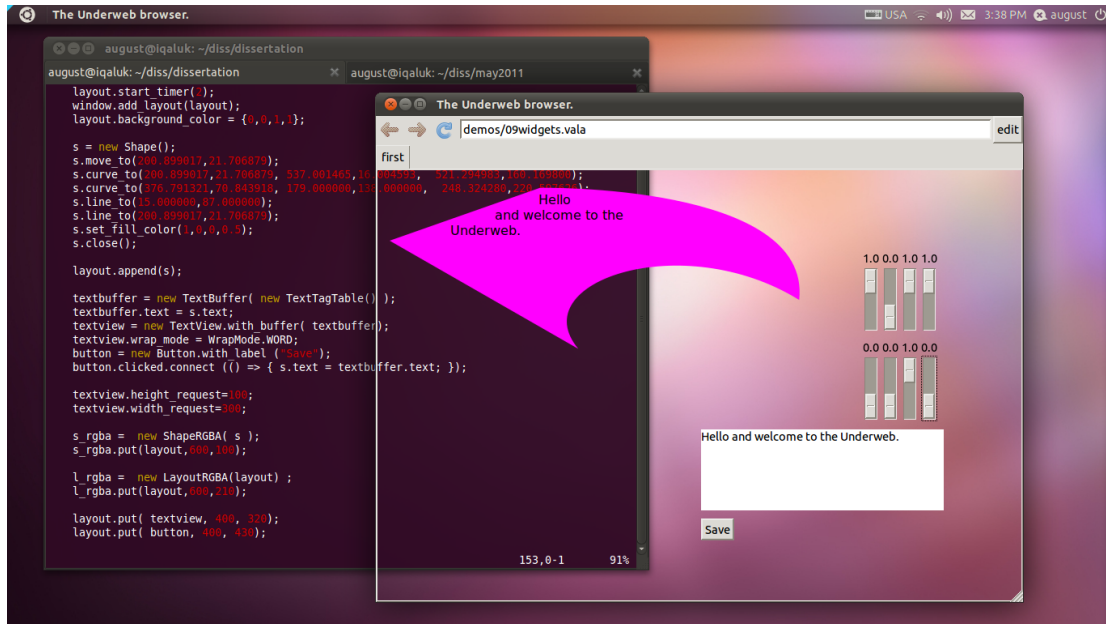


Figure 5.14: Libmentiras also supports completely transparent background in its application documents.

Fig. 5.14 demonstrates another interesting feature of libmentiras and the Underweb in general: transparent backgrounds. Unlike the current WWW, an application document of the Underweb could consist of shapes on screen without any additional visual frame other than the user's desktop.

5.3.8 Video

A major point of contention on the WWW has been video playback. Libmentiras makes it easy to play various video formats from a URI or from disk using the Gstreamer multimedia framework. Fig. 5.15 shows a two video objects; a

DVD sized MPEG stream on the left and another 720p HD formatted MP4 on the right. Like the Image object, the VideoPlayer object of libmentiras is derived from Shape. When a video is loaded, the path is automatically set to be a rectangle at 0,0 with the width and height of the loaded video as defined by its frame size and aspect ratio. The developer may, however, override this and place the video within an irregularly defined path as is shown in the right-hand video of Fig. 5.15.

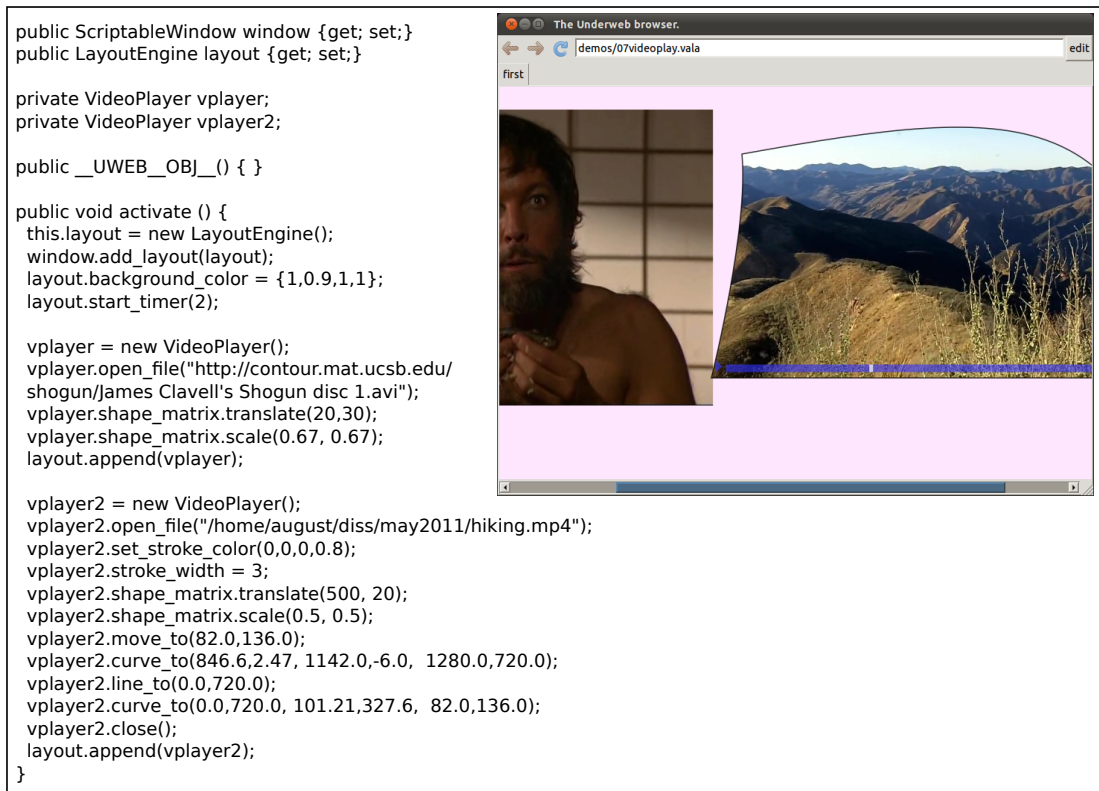


Figure 5.15: Example demonstrating video playback from local file and streamed over HTTP.

The `VideoPlayer` object also comes with a default set of controls for play, pause, and seeking. When the user moves her mouse over video, the controls appear and become functional. The developer may extend or override these behaviours to suit her needs by deriving a new object from the video class.

Besides video playback, the Underweb can both encode and stream audio-video online. Fig. 5.16 demonstrates how to read data from a camera and stream it to an external Icecast server.

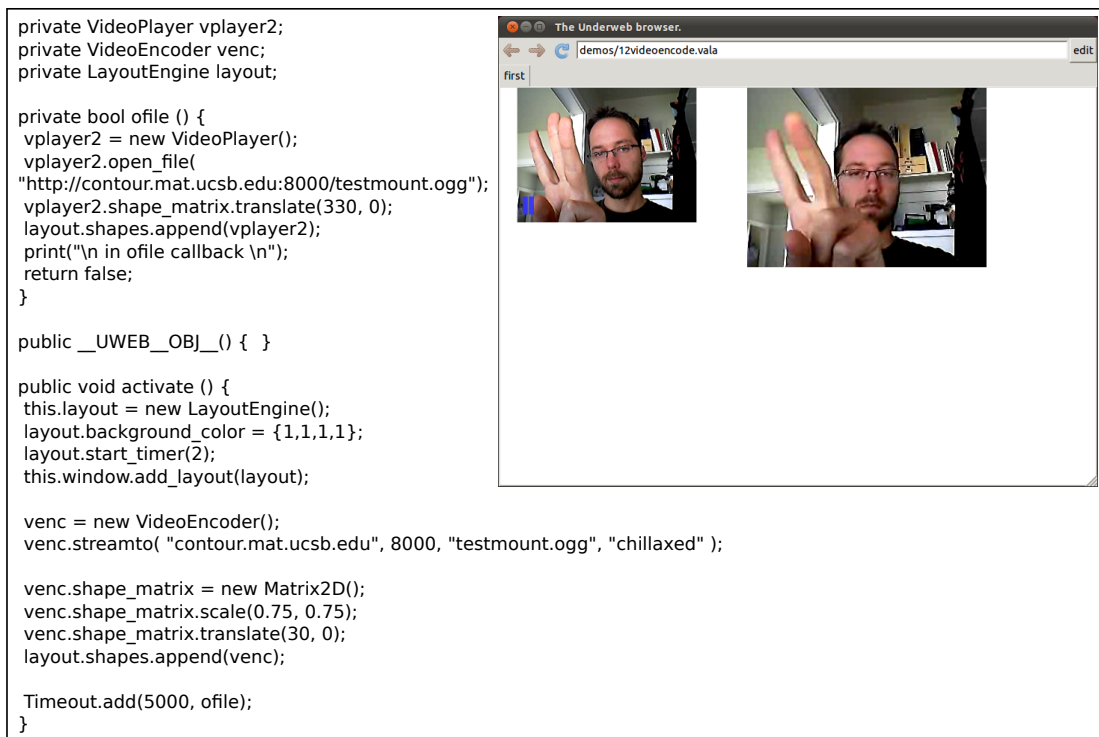


Figure 5.16: Video encoding and streaming example.

In Fig. 5.16, the left-side video shows the video encode stream as it is being sent to the sever. The right-side video shows the playback with some delay. A timeout function that creates and loads the playback stream is set for five seconds to give the video encoder enough time to set up and send data.

A developer may also want to create her own media encoding objects. Since the Gstreamer library is directly accessible to her through this framework, she may forgo the rather simple high-level `VideoPlayer` and `VideoEncoder` objects to create custom media encoders and playback systems. The developer may go so far as to create new Gstreamer plugin objects directly through the Underweb framework.

5.3.9 Audio

Consistent playback and synthesis of audio has long been absent from the WWW. While audio playback remains contentious and inconsistent among browsers, audio synthesis is just now making it into bleeding edge browser developments. Libmentiras presents a few methods for direct synthesis, manipulation, and playback of audio data. Because the Vala language compiles first to C and then to machine code, it is fast enough to handle real-time audio and video synthesis.

Fig. 5.17 shows how libmentiras currently handles audio synthesis. It creates and plays a simple Buzz oscillator that changes frequency depending on mouse motion in the X direction.

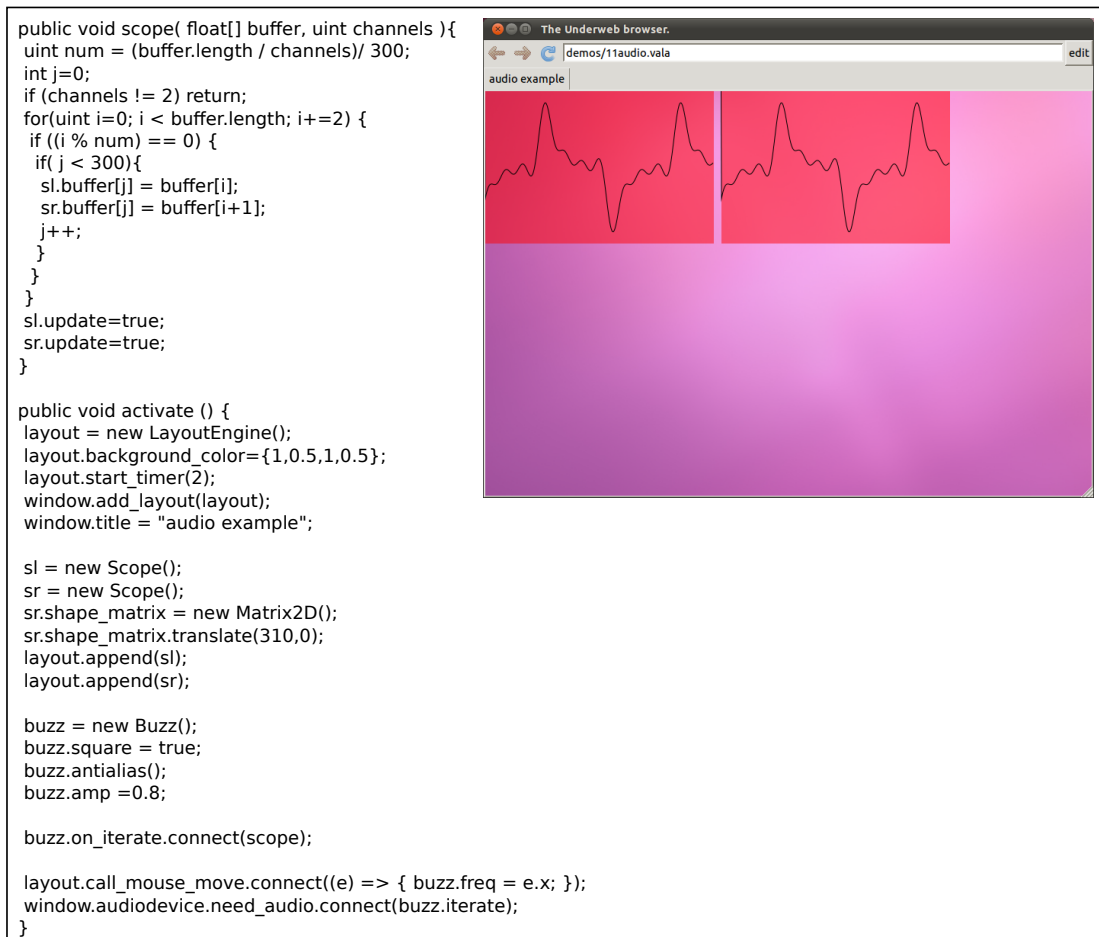


Figure 5.17: A buzz oscillator with a custom draw callback function that draws the waveform within a scope display.

The `Scope` object is derived from `Shape` and uses a custom drawing routine to display the waveform of the currently playing audio buffer. It is not shown in the example code. The `Buzz` oscillator calls the `scope` function on each iteration over a buffer to display the waveform. The global `audiodevice` object may be accessed through the `window` object. Connecting the `buzz.iterate` function to the

need_audio signal of the audio device object tells it to how to get new audio for playback.

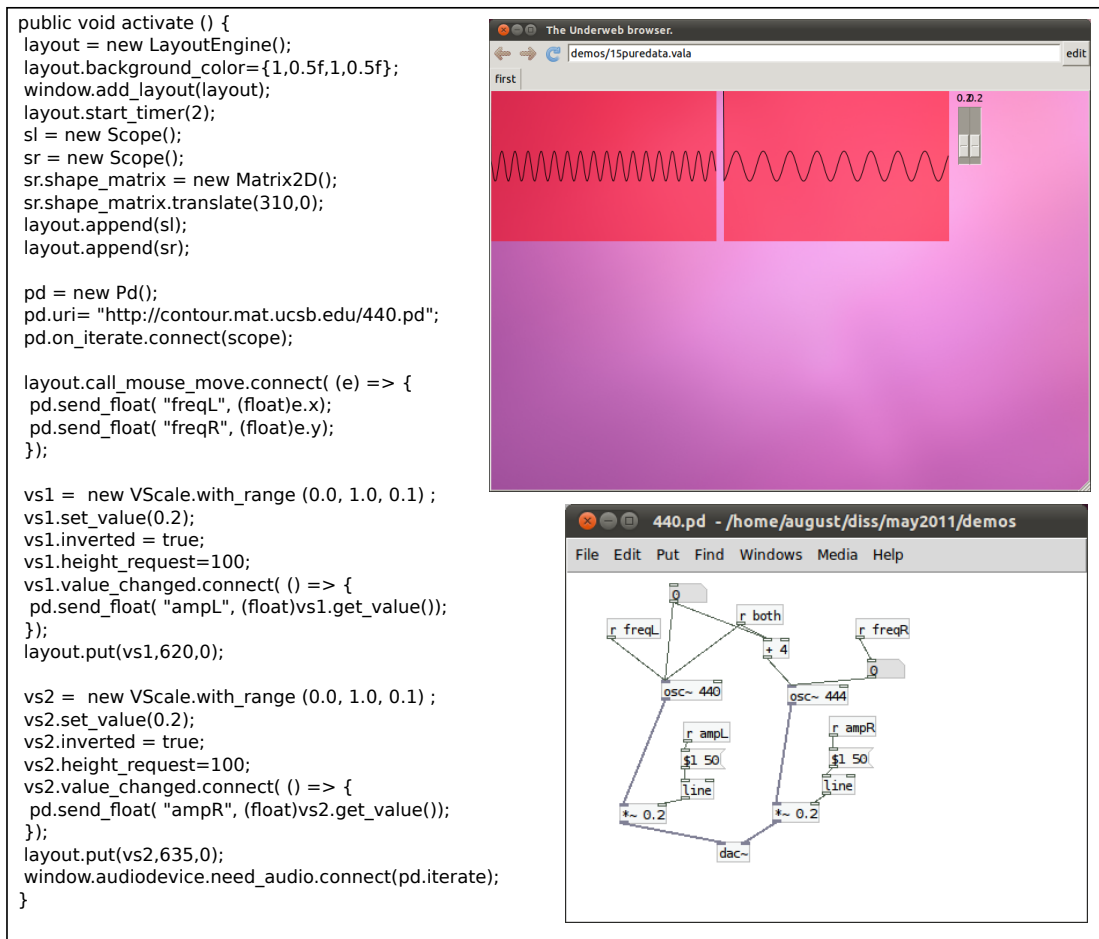


Figure 5.18: Libmentiras links directly to libpd and can run puredata patches internally.

Additionally, because the Underweb is purely free software, it may include other free software inside of it, such as the Puredata. Puredata is a graphical programming language that allows a developer to build a flow control diagram,

known as a patch. These patches can be used to build audio synthesis engines, among other things. Fig. 5.18 shows an audio synthesis application document that uses a Puredata patch as its synthesizer. The top-right image in the figure shows the Underweb output. The bottom-right image shows the loaded Puredata patch, called 440.pd. A developer creates a Puredata engine with `pd=new Pd()`, and loads a patch through its `uri` property. Control can be sent to the patch from libmentiras through `receive`, or `r`, Puredata objects. The example in Fig. 5.18 uses the `freqL` and `freqR` to send control variables based on the user's mouse motion in the X and Y directions, respectively.

5.3.10 Editing

All Underweb application documents that use the libmentiras `LayoutEngine` may be directly edited within the browser. A user clicks on the *Edit* button and sees a new dialogue in which she can create and edit shapes in the layout. This new dialogue provides the user or developer with a point-and-click interface with which she can set all modifiable properties of a generic shape. Any and all shapes may be edited, including those like the `Image` and `VideoPlayer` objects that derive directly from `Shape`. The speed and responsiveness of the browser is also fast enough to allow for the shapes containing videos to be edited while playing.²¹

²¹Depending, of course, on the video size and compression as well as the CPU of the user's machine.

Fig. 5.19 shows editing operations in the Underweb browser. The dialogue where a user may set control parameters is on the left. On the right is an Underweb application document in edit mode with two images.

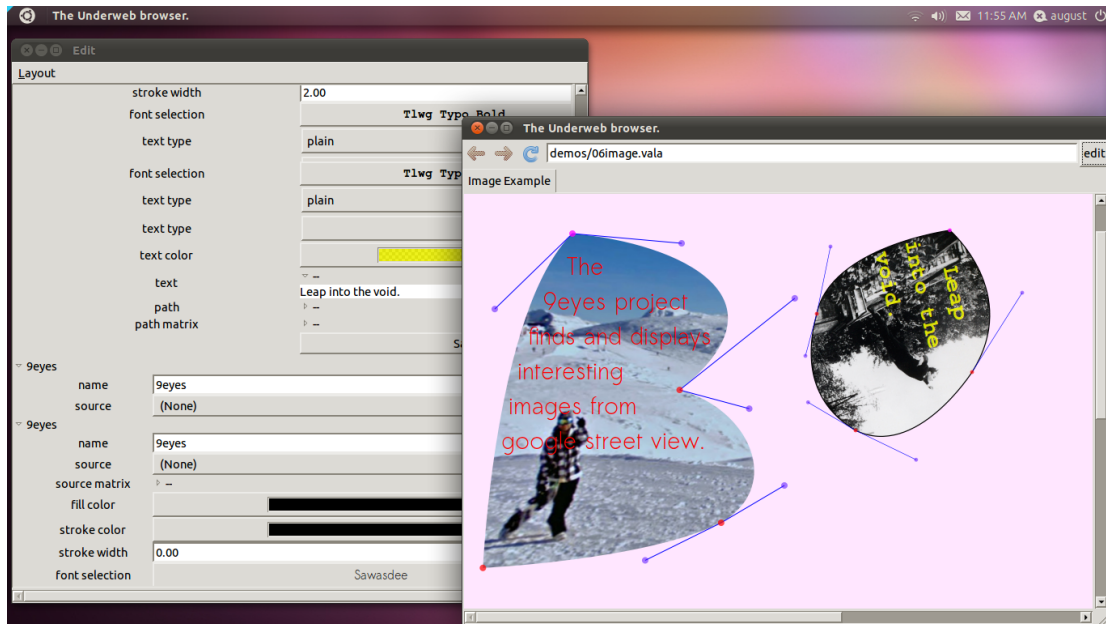


Figure 5.19: Editing operations using libmentiras in the Underweb browser.

Each shape has a number of path points, shown in red. Each path point has at most two control points, shown in blue. Clicking and dragging on the control points and path points of a shape will have a varied effect based on which mouse button was pressed. If a user clicks and drags any point with the left mouse button, the point will follow the mouse movement until the user lets go. If a user clicks a path point with the middle mouse button, that point will be deleted. A right

click on any path point will iterate through the addition and subtraction of two control points.

Additionally, the Ctrl button can be used to modify the behaviour of editing. Ctrl-clicking with the left mouse button on any path point will move the entire shape. Ctrl-clicking with the middle mouse button will scale the shape. Ctrl-clicking with the right button will rotate the shape.

5.3.11 Networking

Like the video playback and encoding features of libmentiras, the networking features are also key in giving new functionality to the WWW that go above and beyond the projective recommendations of HTML5.

Using the GIO and libsoup libraries, libmentiras is able to provide direct access to lower-level software methods for serving and retrieving data over networks. Fig. 5.20 shows how to use the `ServerTCP` object of libmentiras to create a simple TCP socket server and display its incoming data on screen. The code on the left shows how to derive from the `Shape` class and bind a message handler to the `on_message` signal of the `ServerTCP`. Every time the server receives content from a client, it calls the `message_handler` function that sets its `text` property to show the message. Additionally, it then sends the message back to the client, prepended with “server says:”.

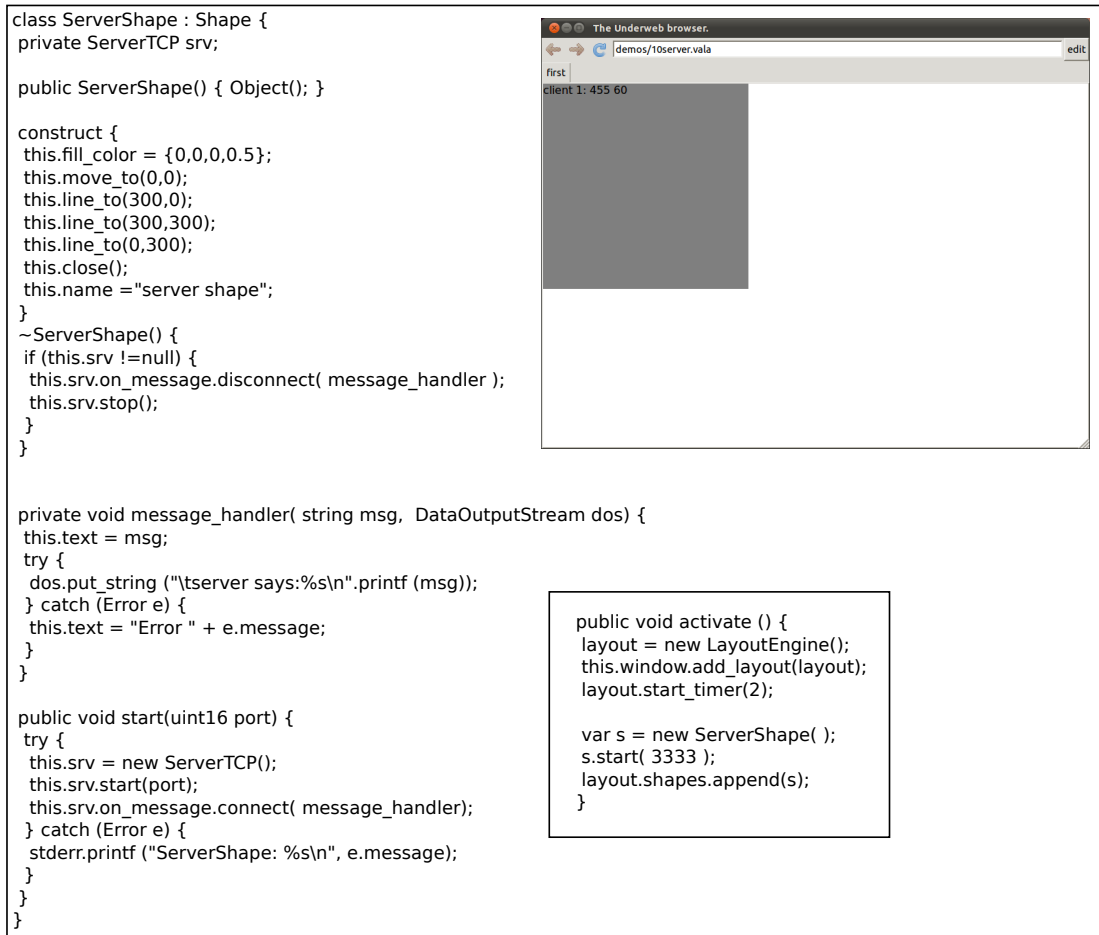


Figure 5.20: Simple TCP server example in libmentiras.

The client example in Fig. 5.21 does a similar thing. It creates a `ClientShape` object that derives from the `Shape` class. Internally, it uses a `Client` object to communicate over a TCP socket. When a mouse moves over the `ClientShape`, it sends its mouse coordinates to the server. When it receives data from the server, it displays it as a text inside of its shape.

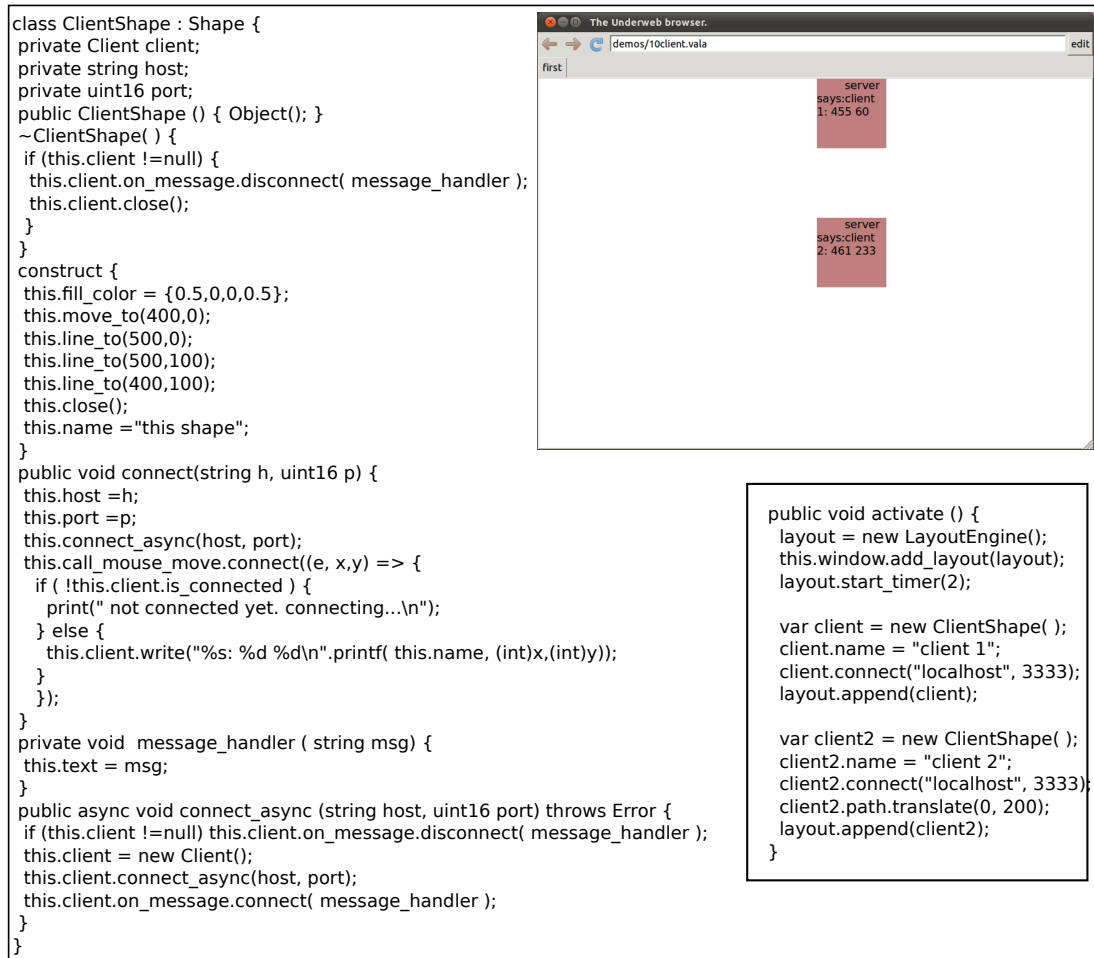


Figure 5.21: A simple TCP client example for bidirectional networking.

Libmentiras also has similar objects that communicate via UDP. With these high-level networking object, the internal complexity of networking is hidden from the developer, allowing her to focus more on developing her own applications.

Altogether, this provides the developer with the bare-bone tools to create her own protocols and private peer-to-peer networks inside of the Underweb WWW. I

imagine and encourage other developers to use this framework to build their own social networks outside of centralized commercial systems.

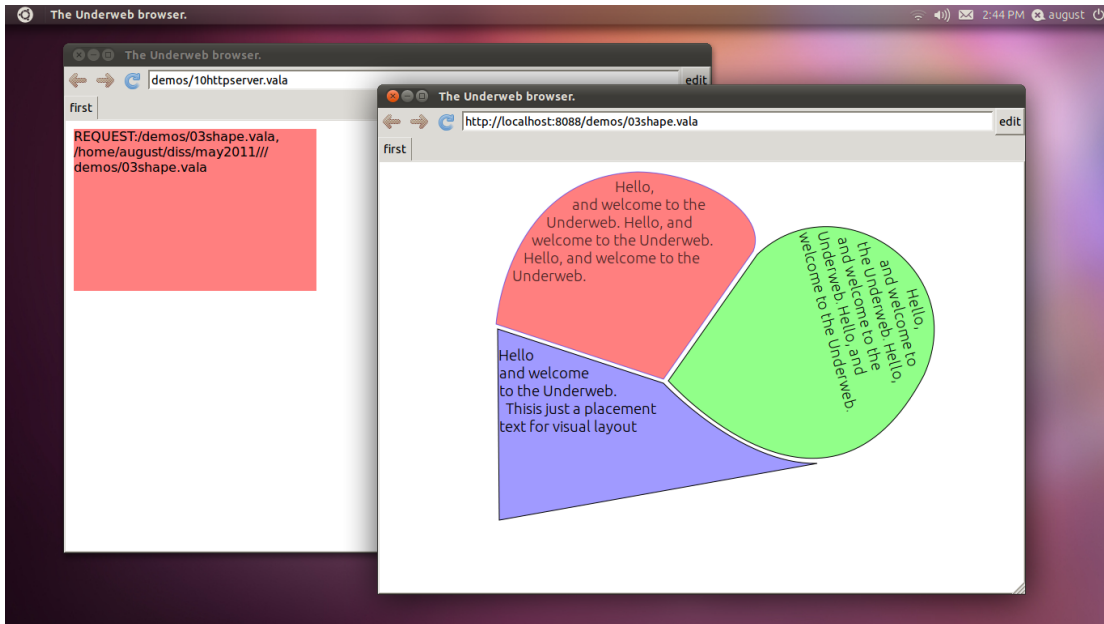


Figure 5.22: A libmentras application document serving another external Underweb browser via HTTP using the libsoup API.

Additionally, libmentiras provides a very basic HTTP server. Fig. 5.22 shows one Underweb application document serving another via HTTP. Besides HTTP, I imagine and encourage developers to write application documents that handle other protocols, such as Open Sound Control (OSC)²² or the Real-time Transport Protocol (RTP).²³

²²Open Sound control is a content format for messaging among computers, sound synthesizers, and other multimedia devices that are optimized for modern networking technology.

²³See <http://www.ietf.org/rfc/rfc1889.txt>. RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services.

Chapter 6

Discussion

The purpose and function of the WWW is very much a result of the assumptions that go into its design. I have so far discussed a number of the assumptions that have been put into the engineering of the WWW and how I think they are flawed. There are, however, a number of creative ways of aligning the current and future assumptions that go into a new WWW design.

There was a time in the early heydays of the WWW when experimentation in browser design was encouraged. A good example of this early energy and experimentation in WWW *formats*¹ besides the commercial and academic ones I mention in my introduction were The International Browserdays in the Netherlands.

¹I say formats here lightly. Many experimentations such as the WebStalker were simple visualizations or mockups made in flash.

The International Browserdays took place 4 times between 1998 and 2002, in conjunction with various academies and institutions of art.² The Browserdays festival invited students and artists to submit alternatives to the then existing browsers in the form of ideas and sketches. Its premises were rather bold and visionary for the time. About the festival, one of the festival organizers says “Its assumption may be as naive as radical: design does matter.”[40] Many of the entries are not documented on the web. However, the organizer contends that the submissions consisted of a variety of entries including everything from “... obvious director/flash works to performances to interface critique.”[40] Furthermore, the results remained closely tied to the discipline of graphic design. From the little documentation that exists of the entries, one can see that the submitted projects often lack technical dexterity, but instead exhibit ingenuity and a strong power of suggestion.

The most interesting of these browserdays experiments is the wining submission for the 6th iteration. It is a simple video document of an old Dutchman speaking in front of a camera about a “personal browser” - a browser that grows with him at birth recording all of his memories and ideas. See <http://www.nl-design.net/browserday/6/pages/watch.html>. It is a personal browser, possibly in ref-

²See <http://www.nl-design.net/browserday/6/> for the last iteration of the festival. I believe the idea and title stem from the Dutch Boekenweek which is a festival of Dutch literature and books that has taken place yearly since 1932.

erence to the “personal” computer or PC, and is interesting because it is pointed at data that we, at least theoretically, cannot or do not yet record. This sibling-like informational entity turns the idea of a browser on its head. Instead of pointing the browser’s focus on data or information external to the “user”, this personal browser should focus on all the internal user data - the personal memories, thoughts, and ideas intrinsic to the user himself.

With my Underweb prototype, I also re-frame the purpose of the browser, structure and format of the WWW. I want the browser to be an active producer of content, not just a passive consumer. The Underweb browser, for lack of a better term, is no longer a just a dumb client in the system. It is an intelligent component of the WWW that includes the same functionality that was previously reserved for servers and operating systems alone.

In this research, I also seek to re-frame how the dynamics of interoperability and innovation take place within the civic space of the WWW. What I offer is not a perfect solution, but more like a starting piece of a yet-to-be-determined puzzle. I agree with Clay Shirky when he said that the WWW was a “joke” of a protocol, but that its strength was its imperfection.[57] In that vein, I also offer my research here as a technically imperfect prototype.

Before discussing future research, I consider some questions and doubts that have surfaced during this development.

6.1 Reinventing the wheel?

Can you imagine a wheel-less society? Can you imagine a society with only one kind of wheel? Can you imagine a society with only one language or one way of saying things? One genre? Does the WWW already do everything it is supposed to do?

My first impetus for this research comes from the difficulty of programming for the web. As such, I have previously thought of the problem in mostly technical terms supported by a technical agenda of *improving* the web in a singular linear fashion - take problem "A" in system and fix with solution "B". I have already met some resistance to this idea, and I believe some defense tactics are useful here.

The more technically inclined will say that there is no reason to change the underlying structure of the web. To them, it *does* what it is supposed to do. The use of the web defines and justifies its purpose and vice versa. In this mentality, there is no distinction between a thing's surface and its underlying mechanisms. There is also no difference between function and its affect. In fact, there is hardly any concern with affect at all as function and affect are tautologically defined in a recursive autopoietic manner. If two fundamentally varied underlying systems produce the same result on the surface screen, there is no perceived difference. The only concern is to build better systems that responds faster, allow more through-

put, and achieve very specific and very myopic goals.³ Unexplainable phenomena such as why Facebook prevails over its antecedents that provide basically the same functional behaviour remain unexplained and are, indeed, unmeasurable. However, it is this form of tautological system where artistic methods live, breath, and ultimately work best.

My objectives with this research are not to reinvent an existing system, but to recognize the already changing system of the WWW and suggest running alternatives. The current WWW is already reinventing itself daily. What was once a file format with only one function - to hyperlink one document in a single format to another document in the same singular format - has now become a multi-functional interactive computational space with incredibly varied, but still limited, file formats. The problem domain and methodology of this dissertation is thus non-linear. The problem of an insufficient WWW infrastructure is changing as solutions are provided.

Furthermore, I argue just the opposite, that the current development strategies of the WWW are perpetually reinventing the wheel. There are 4 main browser developers, each producing virtually the exact same product. The inanity of this is further underscored by the fact that all developers give away their browsers

³I should note that this is a very difficult and noble position to take, for which I have much respect. The work and concentration that goes into engineering and systems design is hard and concrete. We can thank the existence and stability of the Internet as a whole on these principles.

to users without financial remuneration. Why are they wasting precious development time and energy in this game of artificial competition? Where rival browser vendors compete to produce the essentially same give-away product, they could be cooperating to produce a public good.

6.2 Too late?

One question that I have been asked when presenting this research is, “isn’t it already too late?” The WWW is already enormously popular. Additionally, there already exists an astronomical amount of content formatted to run on the web.

My answer is “yes, of course”, but also “well, it depends.” Formats and systems come and go, and are often superseded by newer technologies or by shifts in social dynamics. Archie and Gopher were both information retrieval systems that have been superseded by the WWW. Altavista was superseded by Google. Myspace was superseded by Facebook. The dominance of Internet Explorer is now being superseded by FLOSS browsers. In fact, the current WWW is already worlds different than what it initially was.

What I offer is not really a competing format, however. What I offer is a very embryonic prototype that gives developers a larger superset of design than what does the current WWW. The ideas I present could be taken and integrated

in any WWW browser in direct or altered form. Or, in a similar fashion, if the application presented in this research were to catch on, HTML would still be part of the system, perhaps remaining one of many possible formats loaded by a WWW browser. Since the Underweb can also display HTML formatted content via various mechanisms, it makes for a nice transitional space.

In some ways, I believe I am also too early. The implementation details and complexity of HTML5 are just now starting to surface. How this plays out is yet to be seen. If there is no collective need for alternatives now, perhaps there will be in the future. Without an existing alternative, it will be hard to tell.

6.3 Future Research

Besides developing the software to implement more features, there are a number of areas in which I would like to extend future research. The most salient of these are in the areas of security, encryption, sustainability, design, and licencing. I'd also like to investigate possibilities that are now being provided by the increasingly ubiquitous micro and mobile computational platforms.

Although there can be no absolute security on the WWW and the concept of security itself is very multidimensional and dynamic, I believe one avenue of future research for the Underweb is a serious investigation of security concerns.

As it is now, I suggest only a guiding principle for how to implement a security model; that it should aim for loose security models that offer more functionality and freedom of expression at the expense of requiring more responsibility of its users, and shy away from tighter security models that prefer user constraints over flexibility. I would like to investigate sandboxing strategies mixed with newer trust network technologies so that users can set their own security policies that protect their systems from external attacks. The sandboxing would give users the ability to turn sensitive functions such as file reading and writing on or off. A network system of trust such as that used in PGP encryption would additionally allow a user to gauge the validity and risk of a foreign application document.

Another area of future research will be encryption. To protect users from the actuarial surveillance of centralized cloud technologies, one potential solution is the provision of encryption tools. If encryption was available to all as a default technology of the WWW, users could still use these central services while at the same time protect their personal data from outside introspection.

The sustainability of free software development is another area of future research for this project. While FLOSS provides rights and freedoms to the users of software, it does little to provide financial justice to the developers. There are a number of models that have made FLOSS projects as useful and sustainable as they are today. There are also newer models on the brink from crowd-sourcing

sites such as <http://kickstarter.com>. Like the publicly source funding of Internet development, I believe that any research in the development sustainability of the WWW must also entertain a model that includes public funding. The heart of the problem is not only a civic one, but also a cultural issue. Software is increasingly becoming an unspoken cultural and epistemological necessity of contemporary society and should therefore be subject to the kinds of financial support that science, art, and education receive.

Because the Underweb can potentially exchange binary application documents, licencing issues should be explored further. Like the patent and royalty policies of the W3C, it might be necessary and worthwhile to explore licensing options that balance the rights and freedoms of users and developers. At the moment, the Underweb is licensed under the Gnu General Public Licence (GPL). This stipulates that any application document that is created for the Underweb and distributed publicly must provide its source code and must also be licenced under the GPL. However, there may be cases where the GPL restrictions no longer apply or are too strict. Some further research that brings this discussion to a broader public will be necessary.

One design and infrastructural area that I'd like to further investigate is the frame of the browser itself. Since Underweb documents may also be rendered with a transparent background, the need for an encapsulating graphical frame may no

longer be necessary. If I were to eliminate the frame altogether, the perceived gap between the desktop and the browser would become even smaller.

Another implementation area that could use some investigation is the ability to load and unload Underweb application documents as if they were background modules and not pages. A user could load an application document that had no visual output and allow it to run in the background. This would allow Underweb application documents to function not only as applications and documents, but also as a shared object library or as services for other application documents.

Yet another area of potential research is integration with Ebon Moglen's FreedomBox project. The FreedomBox project aims to build a low-energy, cheap, and tiny WWW server that users can plug directly into a wall socket. According to their pamphlet, it "... is a project that combines the computing power of a smart phone with your wireless router to create a network of personal servers to protect privacy during daily life, maintain beachheads of free network access during times of political instability, and open lines of communication during natural disasters." [43] In a way, the FreedomBox project is trying to do in hardware what I am attempting to do in software - namely, provide users with the realistic means to create their own communication infrastructures. Against the better judgement of many nay-sayers and corporate managers, WWW users have exhibited the will

to produce their own content. It remains to be seen if they can also create their own platforms and infrastructure for this content as we see with Wikipedia.

6.4 Significance & Limitations

It seems passé today to speak of "the Internet revolution." In some academic circles, it is positively naïve. But it should not be. The change brought about by the networked information environment is deep. It is structural. It goes to the very foundations of how liberal markets and liberal democracies have co-evolved for almost two centuries. [9]

The significance of this research is very broad and geared toward a general global user base. By current estimates, the Internet contains at least four billion pages of information, pumped out by around 50 million hosts to nearly a billion users worldwide. The current use of the World Wide Web, defined and abused by competing browser applications and standardization committees, describes a battlefield of conflicting design intentions. Besides changing and improving the cooperative nature of global online communication, this research defines new infrastructures for many forms of scriptable mixed media with applications to VJ's, DJ's, information visualization, online social networks, graphic design and general multi-platform application development.

This research contributes directly to the areas of communication, digital humanities, web studies, and the emerging area of software studies. In these areas,

there are marginal but still significant gaps of knowledge to be filled where lower workings of computational systems meet larger workings of cultural and cognitive systems. The innovations I propose in the form of my Underweb browser seek to provided tangible knowledge in the form of working tools that will either stand on their own or as an example that provokes new research in this field.

I also have humble aspirations that this research helps transition the development of software tools from info-capital methods into the realm of academic, intellectual, and free cultural institutions. As the cultural and social relevance of software is recognized (much like shared roads or public schools), it will become necessary to find vital means by which to study, develop, and nurture (at least certain kinds of) these self-reflective and dynamic artifacts outside of the for-profit infoeconomy. It is my hope that this research helps set the stage for a *materialistic* software and transmission arts study in the humanities (outside of electrical computer engineering) at the University level.

The limitations of this research, however, are not only set by the technical difficulties - which are admittedly vast in and of themselves - but also by social acceptance. A new protocol and browser for the web is somewhat useless without a user and contributor base. Under these assumptions, the scope of this proposal is limited to the development of provisional and demonstrative code, documentation, theoretical developments in the form of text, and community organization. A

full-featured browser by a single individual is not a realistic goal for this research. Instead, an organized sub-set of code and development, based on already existing open source APIs, has been generated to highlight the design issues and act as a possible solution to the current mess we call the WWW. If this much is possible, a larger simultaneous aim of this proposal is to recruit interest and ideas from designers, software developers, and general users to turn these core ideas into a solid, usable, and linked application framework.

Chapter 7

Conclusion

I have discussed the current trajectory and problem space of the WWW. I have also presented an argument and novel prototypical browser for an alternative WWW that provide partial solutions to the problems I discuss.

The problems and issues that inform my motivations can be loosely placed under 4 categories: political, infrastructural, networking, and aesthetic issues.

The political issues of the WWW stem from a number of factors, but mostly from its precarious placement as mediator of a civic space of information, a generally unavoidable nuisance of technologically driven communication spaces. Technologists simply cannot create a mediated space of communication without some sort of political friction from the relationships it builds between inhabitants/participants or in the way it shapes and forms the content of communication. However, how the process of collective development creates and maintains this mediated space is of utmost importance.

For my work here, the political issues revolve around the centralization of the WWW into commercial holding grounds by way of overlay networks, the precarious “openness” of the WWW that includes a lack of protection of user freedoms, the existence and proliferation of proprietary plugins that threaten users with black-box technology, standards bodies that are directed by commercial interest, and low digital literacy that is leaving a majority of authorship roles to so-called experts. The centralizing tendencies of this social and cultural momentum also includes few realistic means of opting out. Many of these issues are interrelated with each other as well as with external systemic problems. The territory in which these issues take place is the browser: an abstract software layer that mediates between the users and the outside online world and between the user and his or her desktop.

This layer of abstraction on top of the desktop operating system functions as a layer of soft control.¹ Through this, it also forms another layer between one user and other users worldwide. If privately owned for-profit services such as Google, Facebook, and Youtube shape and regulate dominant spaces of activity in this layer, there is a risk of forming commercialized centers of control in what was otherwise a horizontal, anarchic, and distributed WWW.

¹There are, of course, security ramifications implied in this layer that I disregard. My belief is that a WWW design should favor more functionality at the risk of less security. Others, such as system administrators, will certainly disagree.

How to resist this form of closed centralization is a key concern for many people, and I believe there is room for technical and pragmatic adjustments in the system of online communication to change this. Key technical features that give users the ability to easily write and publish online are still missing in the WWW. The recent proposed standards of HTML5, which are far from complete or popularized and implemented by browser manufacturers, lack the technical functionality to encode and serve data. Without these tools inside the browser, users and developers have no easy way to create their own informational spaces and are left to the guises of centralized services.

The networking issues fall around missing networking capabilities in the proposed standards. The current and upcoming WWW cannot yet produce client sockets, although functionality for it is in the works. Without client sockets, there can be no direct connection between users for chat and other real-time communication. Not only do current work-arounds, such as Comet and other AJAX or <IFRAME> methods, waste bandwidth, but they also only provide convoluted methodology for something that could be much simpler.

There is no plan whatsoever for server sockets. Without server sockets on the user end (in the browser itself in this case), there can be no real distributed networking at the logical layer, leaving users technically exposed to some form of centralization, unnecessarily so.

While there is limited and disjointed support for audio-video playback, there is no plan for audio or video encoding and/or streaming. Furthermore, where any media format is concerned, users should be provided the tools to not only read it, but produce it. Even if only a limited number of users would want this capability, without it, users are left with a read-only consumer framework.

The missing infrastructural functionality can be outlined as such. There is no support for multiple scripting languages. Without this, users are locked into one monolingual meta-framework for writing WWW constructs. There is also no consensus in the WWW on how to provide generativity of its protocols. In other words, there is no procedure for updating or inventing new protocols other than some haphazard standardization process governed by a committee of mostly industrial participants. Since the strategies of the software industry still function on archaic notions of property that resist methods of sharing knowledge, rights, and productive spaces, this kind of generational activity can only happen in the domain of FLOSS development. Trying to do so under proprietary frameworks would again force the need for committees and standards and build an unnecessary artificial battleground for competition between software vendors. These factors of openness are subtle and complicated. However, within the ecology of online communication they point in the direction of free software development and protection of public wealth. The fact that almost all browsers on the market

are built using FLOSS libraries is partial proof of the fact that the obtuse logic of openness (or freedom if you wish) is prevailing against the proprietary logic of software culture of the 80s and 90s.

There are also additional, but very important, aesthetic concerns that the Underweb addresses. The WWW is now still mostly rectangular. This has to do with its guiding format of text layout invented by engineers. Using normal HTML, it is possible, but very difficult, to make non-rectilinear shapes. Compare this with Adobe's Flash to see how different aesthetics come from different underlying protocols and functionality. Furthermore, the current WWW developer is constrained to use simple boxed layouts for text. A developer cannot have text easily flow inside of curved shapes or along the edges of shapes. Also, the layout in general is still directly derived from hypertext, and is therefore also still page based. A new framework that intentionally assumes that a web application can dynamically load new information and reconfigure itself internally could provide creative leeway for the investigation of new interaction patterns.

Besides the issues of page dynamics, there is the very important issue of limited multimedia formats. Without any consensus on audio and video formats, there can still be no open way of viewing multimedia online, and for everyone. Furthermore, the inclusion of one format necessarily excludes others. With free software development, there is no technical reason to limit the browser in that

way. Currently, there is also no real way to include new formats. New, undiscovered multimedia formats could significantly change the aesthetic experience of the WWW. There are also no easy editors available for producing any of these new dynamic layouts. The algorithms involved in many of the advanced rendering techniques also resist the kind of standardization that would be necessary under current WWW management. They are far too complex. It would be easier, faster, more robust, and universally compatible to simply write them in FLOSS code than in description formats for standardization. The source code already is the standard in many cases. Furthermore, while there are some sketchy developments in the works, there are no real audio synthesis engines for the WWW. The ones that do exist, come in a singular form that work only in one browser.

I believe these issues are technically and practically avoidable, but only through artistic intervention, good logical and technological design, and economic and social justice. My work on the Underweb framework aims for mostly the first two.

The Underweb framework and its internal libmentiras library provide partial solutions to these problems. It gives the developer a thin and flexible technological layer through which to build new communication formats and infrastructures, all within an extensible and multilingual environment that can be programmed in C, Vala, Python and Javascript. The C and Vala languages also give the developer a

chance to directly implement computation intensive algorithms that are impossible to do within the current WWW structure.

Using the Underweb framework, a developer is also no longer limited to the format specifications of the WWW, but can write any kind of software application using any number of available free software APIs. Software libraries that are written in the GObject system are the easiest to use. However, other FLOSS libraries are also available. The developer may also directly include current HTML WWW content by using the WebKit API.²

The Underweb also gives users and developers the missing tools and means of writing and publishing in the WWW. It provides low-level socket functionality for bidirectional communication in both client and server form. It provides audio and video playback, synthesis, and encoding mechanisms. The provision of these tools becomes more and more of an imperative as more and more user data is being placed into centralized commercial systems.

Additionally, the libmentiras library of the Underweb framework assists the developer in drawing simple curved shapes that may contain wrapped text elements, imaging, video playback, procedural texturing, animation, and text internationalization. The layout of an Underweb application document that uses libmentiras

²The Mozilla Gecko engine is also potentially available. The Underweb could implement a new URI scheme that could inform the user of the intended rendering engine type and version, and then use that one specifically. For example: `http+webkit1.8://aug.ment.org` or `http+gecko4.2://aug.ment.org`. This could give users a fine-tuned method of choosing various layout-engines. The engines themselves are just shared object libraries.

is also editable with a point-and-click graphical interface to allow technically unskilled users a viable means of producing their own content.

While the Internet brings people in closer and more direct contact with one another, it is still a space mediated by technology. The medium is not only the message and massager, the medium is the arbiter of form. Until telepathy is an option, this aspect of technologically mediated communication will always be unavoidable. The underlying question that this dissertation investigates is how and in what ways is this form arbitrated. To that end, I present the Underweb framework as a running alternative to the existing WWW.

Bibliography

- [1] M. Abrams, editor. *World Wide Web - Beyond the Basics*. Prentice Hall, 1998.
- [2] A. Anderson, A. Collins, Krishnamurthy, and J. Zahorjan. Pcp: Efficient endpoint congestion control. *Proc. USENIX Symposium on Networked System Design and Implementation (NSDI)*, 2005.
- [3] K. Andrews, F. Kappe, and H. Maurer. The hyper-g network information system. 1(4):206–220, 1995. http://www.jucs.org/jucs_1_4/the_hyper_g_network.
- [4] R. Barbrook. The california ideology. <http://www.hrc.wmin.ac.uk/theory-californianideology-mute.html>, 1995.
- [5] R. Barbrook. The hi-tech gift economy. http://subs01.c3.hu/subs01_2/contributors3/barbrooktext2.html, 2000.
- [6] R. Barbrook. *Imaginary Futures: From Thinking Machines to the Global Village*. Pluto Press, 2007.
- [7] M. Battilana. The gif controversy: A software developer's perspective. <http://www.cloanto.com/users/mcb/19950127giflzw.html>, 1995.
- [8] BBC. Berners-lee on the read/write web. <http://news.bbc.co.uk/2/hi/technology/4132752.stm>, 2005.
- [9] Y. Benkler. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press, 2006.
- [10] T. Berners-Lee. Information management: A proposal. <http://www.w3.org/History/1989/proposal.html>, 1990.
- [11] T. Berners-Lee. Worldwideweb: Proposal for a hypertext project. <http://www.w3.org/Proposal.html>, 1990.

Bibliography

- [12] T. Berners-Lee. Hypertext markup language - 2.0. <http://www.ietf.org/rfc/rfc1866.txt>, 1995.
- [13] T. Berners-Lee. Frequently asked questions. <http://www.w3.org/People/Berners-Lee/FAQ.html>, 2000.
- [14] T. Berners-Lee. How it all started. <http://www.w3.org/2004/Talks/w3c10-HowItAllStarted/>, 2004.
- [15] P. Borsook. How anarchy works. on location with the masters of the meta-verse, the internet engineering task force. http://www.wired.com/wired/archive/3.10/ietf_pr.html, October 1995.
- [16] E. Branigan. *Projecting a Camera*. Routlage, 2006.
- [17] R. Cailliau and C. Petrie. Robert cailliau on the www proposal: "how it really happened.". <http://www.computer.org/portal/web/computingnow/ic-cailliau>, 1997.
- [18] D. Clark. The design philosophy of the darpa internet protocols. *SIGCOMM*, 1988.
- [19] D. D. Clark. A cloudy crystal ball, views of the future. In *Proceedings of the Twenty-Fourth Internet Engineering Task Force*, pages 539–543. Internet Engineering Task Force, July 1992.
- [20] C. Cockburn and S. Glen. Pagelink, an early browser proposal. <http://www.siliconglen.com/pagelink/>, 1990.
- [21] F. Cramer. (echo echo) echo (echo): Command Line Poetics. <http://www.forkable.eu/utils/texts/ONLINE-BUT-NOT-FREE/digitalartistshandbook.org/commandlinepoetics/commandlinepoetics.pdf>.
- [22] H. de Vries, H. de Vries, and I. Oshri. *Standards Battles in Open Source Software*. Palgrave macmillan, 2008.
- [23] G. Debord. Panegyric: Volume i. <http://debordiana.chez.com/english/panegyric.htm>.
- [24] A. Deveria. Timeline of web browsers. http://en.wikipedia.org/wiki/Timeline_of_web_browsers, 2011.

- [25] H. Drolon and F. van den Berg. Freeimage. <http://freeimage.sourceforge.net/>.
- [26] B. Eich and Others. Spidermonkey. <http://www.mozilla.org/js/spidermonkey/>.
- [27] M. Fuller. Introduction. In M. Fuller, editor, *Software Studies*, pages 1–13. MIT Press, 2008.
- [28] N. Gabo and A. Pevsner. The realistic manifesto. In S. Bann, editor, *The Tradition of constructivism*. Viking Press, 1974.
- [29] A. Galloway. *Protocol:how control exists after decentralization*. MIT press, 2004.
- [30] Garnter. Gartner highlights key predictions for it organisations and users in 2008 and beyond. <http://www.gartner.com/it/page.jsp?id=593207>, 2008.
- [31] R. A. Ghosh and P. Aigrain. Study on the: Economic impact of open source software on innovation and the competitiveness of the information and communication technologies (ict) sector in the eu. http://ec.europa.eu/enterprise/sectors/ict/files/2006-11-20-flossimpact_en.pdf, 2006.
- [32] Google. V8. <http://code.google.com/p/v8/>.
- [33] Ippolita, G. Lovink, and N. Rossiter. The digital given: 10 web 2.0 theses. http://journal.fibreculture.org/issue14/issue14_ippolita_lovink_rossiter.html, 2009.
- [34] D. Katabi, M. Handley, and C. Rohr. Congestion control for high bandwidth-delay product networks. *SIGCOMM*, 2002.
- [35] E. Kluitenberg. *Delusive Spaces*. NAi Publishers, 2008.
- [36] T. Kogawa. Micro fm. <http://anarchy.translocal.jp/radio/micro/>, 2006.
- [37] K. Krechmer. Cathedrals, libraries and bazaars. In *SAC*, pages 1053–1057, 2002.
- [38] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesely, 2003.

- [39] G. Langlois, F. McKelvey, G. Elmer, and K. Werbin. Mapping commercial web 2.0 worlds: Towards a new critical ontogenesis. http://journal.fibreculture.org/issue14/issue14_langlois_et_al.html, 2009.
- [40] G. Lovink. Andrej mrackovski: Winner of the browser competition 1999. http://project.waag.org/browser/browserday_1999/index1.html, 1999.
- [41] G. Lovink. Standards for all. <http://networkcultures.org/wpmu/geert/standards-for-all/>, 2009.
- [42] P. Lunenfeld and G. Lovink. Enemy of nostalgia: Victim of the present, critic of the future. http://muse.jhu.edu/journals/performing_arts_journal/v024/24.1lunenfeld.html, 2002.
- [43] E. Moglen. Freedombox. <http://freedomboxfoundation.org/doc/flyer.pdf>.
- [44] Multiple. Cairo. <http://cairographics.org/>.
- [45] Multiple. Gstreamer. <http://gstreamer.freedesktop.org/>.
- [46] S. Nayar. Computational Cameras: Approaches, Benefits and Limits. Technical report, Jan 2011.
- [47] R. Packer and K. Jordan, editors. *Multimedia from Wagner to Virtual Reality*. W.W. Norton & Company Inc., 2001.
- [48] R. Paul. Gartner: 80 percent of commercial apps to use open source by 2012. <http://arstechnica.com/open-source/news/2008/02/gartner-80-percent-of-commercial-software-programs-will-include-open-source-by-ars>, 2008.
- [49] peas team. libpeas - gobject plugin system. <http://live.gnome.org/libpeas>, 2011.
- [50] B. Plaum. Gmerlin. <http://gmerlin.sourceforge.net/>.
- [51] E. Raymond. The cathedral and the bazaar v.3.0. <http://catb.org/~esr/writings/homesteading/cathedral-bazaar/>, 2000.
- [52] Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 1984.

Bibliography

- [53] B. Schneiderman and G. Kearsley. *Hypertext Hands-On!: An Introduction to a New Way of Organizing and Accessing Information/Book and Disk*. Addison Wesley Publishing Company, 1989.
- [54] M. Shemanarev. Antigrain geometry. <http://www.antigrain.com/about/index.html>.
- [55] T. Sherman. Cinematic video. 2006.
- [56] T. Sherman. Vernacular video. 2011.
- [57] C. Shirky. In praise of evolvable systems. <http://www.shirky.com/writings/evolve.html>, 1996.
- [58] R. Stallman. Why software should not have owners. <http://www.gnu.org/philosophy/why-free.html>, 1994.
- [59] R. Stallman. Why open source misses the point of free software. <http://www.gnu.org/philosophy/open-source-misses-the-point.html>, 2010.
- [60] O. Taylor and Others. Pango. <http://www.pango.org/>.
- [61] G. Team. What is gtk+. <http://www.gtk.org/>, 2011.
- [62] G. I. Team. GObject introspection. <http://live.gnome.org/GObjectIntrospection>, 2011.
- [63] P. team. Pygobject - glib/gobject/gio python bindings. <http://live.gnome.org/PyGObject>.
- [64] S. team. Seed - gobject javascriptcore bridge. <http://live.gnome.org/Seed>.
- [65] T. Terranova. Free labor: Producing culture for the digital economy. http://muse.jhu.edu/journals/social_text/v018/18.2terranova.html, 2000.
- [66] E. Thacker. *Protocol:how control exists after decentralization*, chapter Foreword: Protocol Is as Protocol Does. MIT press, 2004.
- [67] W3C. W3c patent policy. <http://www.w3.org/Consortium/Patent-Policy-20040205/>, 2004.
- [68] W3C. W3c members. <http://www.w3.org/Consortium/Member/List>, 2009.
- [69] W3C. W3c mission. <http://www.w3.org/Consortium/mission>, 2009.

Bibliography

- [70] N. Wardrip-Fruin and N. Montfort, editors. *The New Media Reader*. MIT Press, 2003.
- [71] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy. High performance imaging using large camera arrays. In *Proceedings of ACM SIGGRAPH Transactions on Graphics*, volume 24, pages 765–776, July 2005.
- [72] J. L. Zittrain. *The Future of the Internet*. Yale University Press, 2008.