

# Supplemental Expert Report on Invalidity

**ADOBE et al v. EOLAS**

*Richard L. Phillips*

*October 27, 2011*

THIS REPORT CITES TO AND QUOTES FROM DEPOSITION TRANSCRIPTS AND EXHIBITS THAT HAVE BEEN DESIGNATED UNDER THE PROTECTIVE ORDER. BECAUSE NOT ALL OF THOSE TRANSCRIPTS HAVE BEEN DE-DESIGNATED, THIS REPORT IS MARKED

**HIGHLY-CONFIDENTIAL-ATTORNEYS' EYES ONLY**

## Table of contents

Table of contents .....	1
Appendices .....	7
I. Appendix A: CV .....	7
II. Appendix B: Claim chart exhibits .....	7
III. Appendix C: Video Exhibits .....	12
IV. Appendix D: Declaration of Dan Sadowski .....	13
Expert report by Richard L. Phillips on Invalidity .....	14
I. Introduction .....	14
1. EXPERT QUALIFICATIONS .....	14
2. GENERAL TECHNICAL EXPERIENCE.....	14
3. SOFTWARE DEVELOPMENT EXPERIENCE.....	16
II. Information Considered In Forming My Opinions.....	18
1. PATENTS IN SUIT AND DECLARATIONS.....	18
2. USPTO PROSECUTION HISTORIES.....	18
3. PUBLICATIONS AND OTHER REFERENCES.....	19
4. SOFTWARE.....	25
5. COMPUTERS.....	27
6. WWW-TALK PUBLICATIONS AND OTHER EMAILS.....	28

7.	VIDEOS .....	32
8.	LITIGATION MATERIALS.....	32
9.	OTHER .....	33
III.	Tools Available To A Person of Ordinary Skill By October'93 .....	34
1.	Workstations by Sun, Apollo, SGI and NeXT .....	34
2.	UNIX Operating System.....	35
a.	Processes.....	35
b.	Interprocess Communication Techniques.....	36
i.	UNIX IPC Techniques .....	36
ii.	ToolTalk.....	37
iii.	IPC in Other Operating Systems .....	38
iv.	NeXTSTEP Distributed Objects .....	39
3.	X-Windows .....	40
4.	Motif .....	42
a.	Background .....	42
b.	Widgets.....	43
c.	X Events, Callback Functions and user Interaction.....	45
d.	MediaMosaic, X Window Embedding.....	47
5.	Programming Techniques .....	47
a.	Source Code .....	47
i.	Scripts .....	48
ii.	Bytecode .....	48
iii.	Compiled Languages .....	48
b.	Structuring Documents.....	49
i.	Background.....	49
ii.	HTML .....	50
iii.	HTML+ .....	51
iv.	Tag Attributes.....	52
c.	Parsing .....	56
6.	Networked Environments.....	57
a.	Client-Server Computing.....	57

b.	Distributed Computing.....	58
IV.	Hypermedia Browsers That Existed By 1993 - 1994 .....	59
1.	Network-Enabled Compound Document Platforms .....	59
a.	OLE.....	59
2.	Hypermedia Browsers .....	60
a.	MediaView .....	60
b.	HyperCard .....	70
i.	HyperCard Overview and its Operating Environment .....	70
ii.	HyperCard's Architecture for Hypermedia Browsing.....	71
iii.	HyperCard's Architecture for Hypermedia Authoring.....	72
iv.	HyperTalk Scripting Language .....	72
v.	XCMDs, XFCNs and Mac-specific Issues.....	73
vi.	VideoWorks and Director.....	74
vii.	QuickTime .....	76
viii.	HyperCard in Networked Environments.....	79
V.	World Wide Web Browsers That Existed By 1993 - 1994.....	80
1.	The Internet .....	80
a.	Background - WWW .....	80
b.	Protocols: HTTP, HTML and W3C.....	81
2.	Mosaic.....	82
a.	Ordinary Operation .....	82
i.	Overview .....	82
ii.	Helper Applications and MIME .....	85
iii.	Widgets .....	85
iv.	Distributed Applications .....	86
b.	www-talk .....	87
i.	Suggestions on Inline Viewing.....	87
ii.	IMG Tag .....	89
iii.	Inline xv.....	90
iv.	Interactive Widgets.....	92
3.	Viola and ViolaWWW.....	93

- a. Viola Toolkit/Language System.....93
  - i. Overview .....93
  - ii. Object-oriented Architecture.....93
  - iii. Scripting Language.....94
  - iv. Application Support Libraries .....95
- b. Viola in a Hypermedia Networked Environment.....96
  - i. HTTP versioning .....96
  - ii. [Viola-5/27/93] download location.....97
  - iii. Alpha Release's interoperability with HTTP .....98
- c. ViolaWWW: Web Browser Based on Viola Toolkit/Language System 98
  - i. Overview .....98
  - ii. Embedded, Interactive Objects .....100
  - iii. Further Development.....101
- d. Additional ViolaWWW Architectural Issues .....104
  - i. HMML and HTML Documents .....104
  - ii. HMML and HTML Document Parsers.....105
  - iii. Tags for Embedded Objects .....106
  - iv. LINK Tag in HTML .....107
  - v. Examples of Embedded, Interactive Objects .....108
  - vi. Security flag for embedding in beta versions .....109
  - vii. Distributed Applications .....111
- VI. Software development during early 1990s .....114
- VII. The patents-in-suit .....116
  - 1. The patent specification and asserted claims .....116
  - 2. Claim construction .....122
  - 3. The prosecution histories of the patents in suit .....135
    - a. The '906 Patent.....135
      - i. The Original Patent Prosecution Of The '906 Patent.....135
      - ii. The Director-Ordered Reexamination Of The '906 Patent.....142
      - iii. The Second Reexamination Proceeding .....149
      - iv. The "Koppolu" Interference.....155

b.	The '985 Patent.....	155
i.	The Original Patent Prosecution Of The '985 Patent.....	155
VIII.	Legal Standards.....	157
1.	Anticipation.....	159
2.	Obviousness .....	161
3.	Section 112 Requirements.....	165
4.	Level of Ordinary Skill in the Art.....	166
5.	Materiality.....	167
IX.	My Opinions Relating To This Case .....	168
1.	Viola.....	168
a.	Analyzed source code.....	168
b.	Vplot.....	169
c.	Viola/5/12/93 .....	171
d.	Viola-5/27/93.....	<b>Error! Bookmark not defined.</b>
e.	Alpha release.....	179
f.	February beta release .....	182
g.	March beta release .....	184
h.	Other codebases and materials .....	187
i.	Conferences and demonstrations.....	188
j.	Pei Wei's August 1994 paper.....	190
k.	Claim construction.....	198
2.	MediaView.....	200
3.	MediaView and Mosaic .....	205
a.	Mosaic prior art .....	205
b.	Obviousness based on MediaView and Mosaic .....	207
4.	Mosaic, HTML+, and disclosure and testimony of Bill Janssen .....	215
a.	HTML+ and disclosure and testimony of Bill Janssen as prior art.....	215
b.	Obviousness based on these references.....	216
5.	Mosaic in combination with Chris McRae's posting to www-talk.....	222
a.	McRae's posting as prior art .....	222
b.	Obviousness based on these references.....	223

- 6. Mosaic in combination with Adobe PDF related postings to www-talk. 225
  - a. Adobe PDF is prior art .....225
  - b. Obviousness based on Mosaic with Acrobat and PDF www-talk postings.....226
- 7. HyperCard (including Director and Quicktime) and Viola .....228
  - a. HyperCard as prior art.....228
  - b. QuickTime as prior art .....230
  - c. Obviousness based on HyperCard/QuickTime and Viola.....233
  - d. Obviousness based on HyperCard/QuickTime and Mosaic .....239
  - e. Director as prior art.....239
  - f. Obviousness based on HyperCard/Director and Mosaic .....241
  - g. Obviousness based on HyperCard/Director and Viola .....246
  - h. Claim construction issues related to HyperCard .....247
- 8. Viola and Cohen .....247
- 9. Any of the above-identified prior art, and distributed applications 255
  - a. VIS .....255
  - b. Collage.....257
  - c. Scientific Visualization Workbench.....259
  - d. Pei Wei's August 1994 paper [Wei94].....261
  - e. Mathematica.....262
  - f. RenderMan .....264
  - g. Motivations to combine .....265
  - h. Claim construction.....267
- 10. Invention date .....267
  - a. Opinions .....267
    - i. Overview .....267
    - ii. Opinion related to type information.....269
    - iii. Opinion related to enabling an end-user to directly interact .....270
  - b. The '906 patent and appendices .....270
  - c. Other documents upon which Eolas relies as evidence of conception and reduction to practice .....271

- d. Conclusions .....282
- X. Section 112 .....283
  - a. The Specification Lacks An Adequate Written Description Of An "Embed Text Format" Other Than One Implemented Using Special Tags .283
  - b. The Specification Lacks An Adequate Written Description Of An "Embed Text Format" Other Than One Located At The "First Location" ....287
  - c. The Specification Lacks An Adequate Written Description Of An "Embed Text Format" That Is Identified By A Method Other Than Parsing  
289
  - d. The Specification Lacks An Adequate Written Description Of "Executable Application" .....290
- XI. Materiality of prior art .....292
  - 1. ViolaWWW and Vplot .....292
  - 2. Acrobat .....298
  - 3. MediaView .....299
- XII. Secondary considerations of non-obviousness .....299
  - 1. No Evidence of Commercial Success .....300
  - 2. No Evidence of Inability to Design Around the Claimed Invention  
303
  - 3. No Evidence of Long Felt Need and Skepticism of Experts .....304
  - 4. No evidence of "Copying by others" .....305
- My Compensation .....306
- Prior Expert Testimony .....307

## Appendices

### I. Appendix A: CV

Appendix A is my CV, which provides information about my education, employment, and publications. A summary is also provided below.

### II. Appendix B: Claim chart exhibits

Appendix B sets forth a series of Invalidity Claim Chart Exhibits.

- **Claim Chart Exhibit 1: "Viola-5/12/93"**

[Viola-5/12/93], dated May 12, 1993, and located at [PA-NAT-78\viola\1993-05-12 DX34 - Ex A to Inv Contentions\software\viola930512.tar.gz]. This is prior art under 35 U.S.C. §§ 102(a), (b), (g) and 103.

I note that this file is not the same as "DX34" that was submitted as a trial exhibit during the *Eolas v. Microsoft* litigation. [Viola 5/12/93] is a single zip file contained in the larger directory of files included in the DX34 trial exhibit. I further note that, according to the testimony of Scott Silvey and Pei Wei, [Viola-5/12/93] is, insofar as is relevant here, effectively identical to a version of Viola that was demonstrated to engineers at Sun Microsystems, Inc. on May 7, 1993. The complete trial exhibit is contained at PA-NAT-78\viola\1993-05-12 DX34 - Ex A to Inv Contentions, and includes all files and folders therein (including sample images showing embedded interactive images from doodle and testplot). DX34 was also produced by Eolas at EOLASTX-000186973.

My opinions about [Viola-5/12/93] are further informed by the full body of Viola-related prior art and testimony regarding the same, and I reserve the right to opine on [Viola-5/12/93] based on other Viola prior art.

- **Claim Chart Exhibit 2: "Viola-5/27/93 "**

Based on [Viola-5/27/93], dated May 27, 1993, and located at [PA-NAT-78\viola\1993-05-27 DX37 - Ex AE to Inv Contentions/violaTOGO.tar.z]. This is prior art under 35 U.S.C. §§ 102(a), (b), (g) and 103.

I note that this file is not the same DX37 that was submitted as a trial exhibit during the *Eolas v. Microsoft* litigation. [Viola-5/27/93] is a single zip file contained in the larger directory of files included in the DX37 trial exhibit. The complete trial exhibit is contained at EOLASTX-000186966, and includes all files and folders therein.

My opinions about [Viola-5/27/93] are further informed by the full body of Viola-related prior art and testimony regarding the same, and I reserve the right to opine on [Viola-5/27/93] based on other Viola prior art.

- **Claim Chart Exhibit 3: "Viola Alpha"**

Based on [Viola Alpha], dated October 16, 1993 [PA-NAT-78\viola\1993-10-16 Alpha release - Ex AF to Inv Contentions\violaTOGO.tar.z]. This is prior art under 35 U.S.C. §§ 102(a), (b), (g) and 103.

My opinions about [Viola-Alpha] are further informed by the full body of Viola-related prior art," identified below, and testimony regarding the same and I reserve the right to opine on [Viola-alpha] based on other Viola prior art.

- **Claim Chart Exhibit 4: "Viola February Beta"**



[Viola February Beta], dated February 23, 1994 [PA-NAT-78\viola\1994-02-23 February beta release - Ex AG to Inv Contentions]. This is prior art under 35 U.S.C. §§ 102(a), (b), (g) and 103.

My opinions about [Viola-Feb Beta] are further informed by the full body of Viola-related prior art and testimony regarding the same, and I reserve the right to opine on [Viola-Feb Beta] based on other Viola prior art.

- **Claim Chart Exhibit 5: "Viola March Beta"**

[Viola March Beta], dated March 23, 1994 [PA-NAT-78\viola\1994-03-23 March beta release - Ex AH to Inv Contentions]

My opinions about [Viola-March Beta] are further informed by the full body of Viola-related prior art and testimony regarding the same, and I reserve the right to opine on [Viola-March Beta] based on other Viola prior art.

- **Claim Chart Exhibit 6: "Pei Wei's August 1994 paper"**

Based on Pei Wei's publication "A Brief Overview of the Viola Engine, and its Applications" bearing the date August 16, 1994, available at [PA-00318355]. Also available at [PA-00318385] through [PA-00318392]. This is prior art under 35 U.S.C. §§ 102(a), (b), (g) and 103.

My opinions about [Wei94] are further informed by the full body of Viola-related prior art and testimony regarding the same, and I reserve the right to opine on [Wei94] based on other Viola prior art.

- **Claim Chart Exhibit 7: "MediaView"**

Based on each, individually, and in combination: (1) An Interpersonal Multimedia Visualization System, IEEE Computer Graphics & Applications, May, 1991 [PA-00273175] [Phillips91a]; (2) MediaView: An Editable Multimedia Publishing System Developed with An Object-Oriented Toolkit, Proc. USENIX Summer Conf., Nashville, TN, June, 1991 [PA-00327398] [Phillips91b]; (3) MediaView: A General Multimedia Digital Publication System, Comm. ACM, Vol. 34, No. 7, July, 1991 [PA-00273194] [Phillips91c]; (4) MediaView, Videotaped Demonstration of MediaView Capabilities by Los Alamos National Laboratory, VIDEO-93-121, March 20, 1993 [LANL93]; and (5) materials on the NeXT computer with MediaView related materials on it maintained in my possession since prior to the date of publication of the above references regarding public demonstrations and uses of the MediaView system, by me, embodied in these materials and the referenced and cited below and the accompanying charts. This is prior art under 35 U.S.C. §§ 102(a), (b), (g) and 103 (in combination with any prior browser that handles HTML tags, e.g. Mosaic, CERN web browser and Viola).

My opinions about MediaView are further informed by the full body of MediaView-related prior art and testimony regarding the same, including my personal experience and knowledge, and I reserve the right to opine on MediaView based on other MediaView prior art.

- **Claim Chart Exhibit 8: "Mosaic"**

Based on "NCSA Mosaic for X 2.0 available", WWW-Talk, Oct-Dec, 1993 [PA-00292659] [Andreessen93a]; NCSA Mosaic Technical Summary [PA-00292824] [Andreessen 93b]; NCSA Collage for the Macintosh version 1.0, October 1992 [PA-00292677] [Collage92]; Mosaic software including the codebases found at [PA-NAT-00000044] – [PA-NAT-00000046]; Bina Tr. and Exs. 4 and 7; McRae Tr. and Ex. 37 (messages from April-June 1993); Martin Tr. including 425:13-426:7 and 532:10-22; my personal experience with the Mosaic browser. This is prior art under 35 U.S.C. § 103.

My opinions about Mosaic are further informed by the full body of Mosaic-related prior art and testimony regarding the same, and I reserve the right to opine on Mosaic based on other Mosaic prior art.

- **Claim Chart Exhibit 9: "Mosaic, HTML+, and disclosure and testimony of Bill Janssen"**

"NCSA Mosaic for X 2.0 available", WWW-Talk, Oct-Dec, 1993 [PA-00292659] [Andreessen93a]; NCSA Mosaic Technical Summary [PA-00292824] [Andreessen 93b]; NCSA Collage for the Macintosh version 1.0, October 1992 [PA-00292677] [Collage92]; Mosaic software including the codebases found at [PA-NAT-00000044] – [PA-NAT-00000046]; My personal experience with the Mosaic browser; Video: The National Center for Supercomputing Applications Software Development Group presents NCSA Mosaic [Hardin 93]; "HTML+ (Hypertext markup language), Hewlett-Packard, 1993 [Raggett93a] [PA-00321233]; Deposition of William Janssen (May 11, 2011) [Janssen Dep.]; Exhibits to [Janssen Dep.], including Exhibit 6 [PH\_001\_0000598210], Exhibit 8 [PH\_001\_0000598248], Exhibit 9 [PA-00306624], Exhibit 10 [PH\_001\_0000588858], and Exhibit 11 [PA-00333547]. This is prior art under 35 U.S.C. § 103.

My opinions about Mosaic are further informed by the full body of Mosaic-related prior art and testimony regarding the same, and I reserve the right to opine on Mosaic based on other Mosaic prior art.

- **Claim Chart Exhibit 10: "Mosaic and Chris McRae's June 26, 1993 posting to www-talk, "Re: Xmosaic and Xv"**

Based on "NCSA Mosaic for X 2.0 available", WWW-Talk, Oct-Dec, 1993 [PA-00292659] [Andreessen93a]; NCSA Mosaic Technical Summary [PA-00292824] [Andreessen 93b]; NCSA Collage for the Macintosh version 1.0, October 1992 [PA-00292677] [Collage92]; Mosaic software including the codebases found at [PA-NAT-00000044] – [PA-NAT-00000046]; My personal experience with the Mosaic browser, and Chris McRae's June 26, 1993 posting entitled "Re: Xmosaic and Xv" [www-talk-00293020]; McRae Exhibit 27 and Bina Exhibits 4 and 7. This is prior art under 35 U.S.C. § 103.

My opinions about Mosaic are further informed by the full body of Mosaic-related prior art and testimony regarding the same, and I reserve the right to opine on Mosaic based on other Mosaic prior art.

- **Claim Chart Exhibit 11: "HyperCard and QuickTime: QTMovie XCMD"**

The Complete HyperCard 2.0 Handbook, 3<sup>rd</sup> Edition [PA-00288603] [Goodman90]; Eric Lease Morgan, "Implementing TCP/IP Communications with HyperCard" Information Technology and Libraries, Dec. 1992; 11, 4; ABI/INFORM Global. pp. 421-432 [Morgan92] [PA-00290482]; John R. Powers, III, "Mac to Mainframe with HyperCard." MacTutor, June 1990 [PA-00288589] [Powers90]; Declaration of Dan Sadowski (Appendix D to this report) [Sadowski11]; David L. Drucker and Michael D. Murie, "QuickTime Handbook: The Complete Guide to Mac Movie Making." August 1992. [Drucker92] [PA-00289694]; Jerry Borrell et al., "Mastering the World of Quicktime," First Edition, Random House, 1993 [Borrell93] [PA-00328099]. This is prior art under 35 U.S.C. § 103 (due to the Court's analysis of the "location" issue).<sup>1</sup>

My opinions about HyperCard are further informed by the full body of HyperCard-related prior art, including [HyperCard] software, and testimony regarding the same, and I reserve the right to opine on HyperCard based on other HyperCard prior art. My opinions about QuickTime are further informed by the full body of Quicktime-related prior art, including [Quicktime] software, and testimony regarding the same, and I reserve the right to opine on Quicktime based on other Quicktime prior art.

- **Claim Chart Exhibit 12: "HyperCard and QuickTime: Movie XCMD"**

Based on The Complete HyperCard 2.0 Handbook, 3<sup>rd</sup> Edition [PA-00288603] [Goodman90]; Eric Lease Morgan, "Implementing TCP/IP Communications with HyperCard," Information Technology and Libraries, Dec. 1992; 11, 4; ABI/INFORM Global. pp. 421-432. [Morgan92] [PA-00290482]; John R. Powers, III, "Mac to Mainframe with HyperCard," MacTutor, June 1990 [PA-00288589] [Powers90]; Declaration of Dan Sadowski (Appendix D to this report) [Sadowski11]; David L. Drucker and Michael D. Murie, "QuickTime Handbook: The Complete Guide to Mac Movie Making." August 1992. [Drucker92] [PA-00289694]; Jerry Borrell et al., "Mastering the World of Quicktime," First Edition, Random House, 1993 [Borrell93] [PA-00328099]. This is prior art under 35 U.S.C. § 103 (due to the Court's analysis of the "location" issue).

My opinions about HyperCard are further informed by the full body of HyperCard-related prior art, including [HyperCard] software, and testimony regarding the same, and I reserve the right to opine on HyperCard based on other HyperCard prior art. My opinions about QuickTime are further informed by the full body of Quicktime-

---

<sup>1</sup> There is a difference between [Martin11]'s application on the location issue and mine. Under my understanding and application of the construction, the location of the embed text format in the hypermedia document is specific relative to the other parts of the document. Under [Martin11], the location is not specific relative to the other parts of the document. Using [Martin11]'s application, each non-patent reference identified as a 103 reference under this application would be a 102(a), (b), and (g) reference too, and the Cohen patent would be a 102(e) reference too.

related prior art, and testimony regarding the same, including [Quicktime] software, and I reserve the right to opine on Quicktime based on other Quicktime prior art.

- **Claim Chart Exhibit 13: "HyperCard and Director"**

Danny Goodman, *The Complete HyperCard 2.0 Handbook*, 3<sup>rd</sup> Edition, Bantam Books, Inc., August 1990. [PA-00288603] [Goodman90]; HyperCard software (e.g., versions 2.0, 2.1, or 2.2); Eric Lease Morgan, "Implementing TCP/IP Communications with HyperCard," *Information Technology and Libraries*, Dec. 1992; 11, 4; ABI/INFORM Global. pp. 421-432. [Morgan92] [PA-00290482]; John R. Powers, III, "Mac to Mainframe with HyperCard," *MacTutor*, June 1990 [PA-00288589] [Powers90]; Declaration of Dan Sadowski (Appendix D to this report) [Sadowski11]; MacroMind Player Manual distributed with Director 3.1.3. This is prior art under 35 U.S.C. § 103 (due to the Court's analysis of the "location" issue).

As noted above, my opinions about HyperCard, as well as Director and the HyperCard user manual for Director are informed by the full body of HyperCard and Director prior art and testimony regarding the same, and reserve the right to opinion on HyperCard and Director based on other prior art and testimony regarding the same.

- **Claim Chart Exhibit 14: "Cohen"**

US Patent 5,367,621 to Cohen et al. This is prior art under 35 U.S.C. § 103 (due to the Court's analysis of the "location" issue).

My opinions about Cohen are further informed by Prosecution History documents involving this reference from Reexamination No. 90/007,858, and I reserve the right to opine on Cohen based on my understanding of these documents. These documents include: Office Action Rejection (July 30, 2007) [PH\_001\_0000786943]; Applicant Remarks (September 27, 2007) [PH\_001\_0000787028]; Office Action (Final) Rejection (April 18, 2008) [PH\_001\_0000787208], and Applicant Amendment and Remarks (June 23, 2008) [PH\_001\_0000787257].

### **III. Appendix C: Video Exhibits**

Appendix C sets forth an index of video demonstration exhibits. These videos were provided on a DVD in the directory "Appendix C - Video Demonstrations" with my July 20, 2011 report with the exception of [LANL93]. The directory organizes my videos into subdirectories based on the prior art systems being demonstrated: "Viola," "HyperCard," "Mosaic," "SunOS," "MediaView," "Standalone

applications," and "OLE." Also included in the DVD is a subdirectory labeled "Related materials," which includes other files related to my video demonstrations.

As is noted above, I recently discovered from a list compiled for a Freedom of Information Act request that Los Alamos National Laboratory ("LANL") recorded a demonstration of MediaView. [LANL93] The date indicated on the media for that video is March 20, 1993. See [ADBE0196693, October 31, 2010] I have received a copy of that video from LANL and will rely on it as prior art evidence for MediaView, demonstrative evidence of MediaView and as representative of the state of the art at the time of the alleged invention.

I intend to rely on these video demonstration exhibits to show or explain the functionality and operation of prior art systems, to support my opinions of anticipation and obviousness in connection with these prior art systems, and to show what was known to persons of ordinary skill in the art during the time of the prior art. In certain locations in my report or claim charts, I make citations to certain video demonstrations. Those citations are exemplary only, and I may use some or all of these video demonstration exhibits to support my opinions of anticipation or obviousness for the prior art systems to which they are directed, and to show what was known to persons of ordinary skill in the art.

#### **IV. Appendix D: Declaration of Dan Sadowski**

Appendix D, which was served on July 20, 2011 with my initial report, includes a Declaration of Dan Sadowski ([Sadowski11]) and exhibits, which I reviewed in preparation of this report. I incorporate Appendix D as if it were a part of this report too.

## Expert report by Richard L. Phillips on Invalidity

### I. Introduction

1. I have been retained by the Defendants<sup>2</sup> as an expert witness in the above entitled action. I have been asked to render an opinion regarding the validity of the asserted claims in U.S. Patent Nos. 5,838,906 and 7,599,985. The following is my written report, submitted pursuant to Federal Rule 26(a)(2) detailing the subject matter areas and opinions about which I may testify in this litigation if called upon to do so.

2. I understand that the following claims of the '906 patent are being asserted: Claim 1 (and dependent claims 2, 3, and 11), and claim 6 (and dependent claims 7, 8, and 13).

3. I understand that the following claims of the '985 patent are being asserted: Claim 1 (and dependent claims 2-11), claim 16 (and dependent claims 17-19), claim 20 (and dependent claims 21-23), claim 24 (and dependent claims 25-27), claim 28, claim 36 (and dependent claims 37-39), and claim 40 (and dependent claims 41-43).

#### 1. EXPERT QUALIFICATIONS

4. Appendix A is my CV, which provides information about my education, employment, and publications. Below is a summary.

#### 2. GENERAL TECHNICAL EXPERIENCE

5. I have spent a career in teaching and research that spans nearly 50 years. In that period I have authored over 45 technical articles and symposium proceedings. I began my academic career in 1961 at the University of Michigan in the Aerospace

---

<sup>2</sup> The Defendants are Adobe Systems Incorporated, Amazon.com, Inc., CDW LLC, Citigroup Inc., Frito-Lay, Inc., The Go Daddy Group, Inc., Google Inc., J.C. Penney Corporation, Inc., Staples, Inc., Yahoo! Inc., and YouTube, LLC.

Engineering Department. In that position I taught courses in aircraft and space propulsion and aerodynamics.

6. In 1974, because of my growing involvement in computer science, I accepted a joint appointment in the Department of Electrical Engineering and Computer Science. While continuing to teach in Aerospace Engineering, I developed and taught courses in computer graphics and computer-aided design. I had the principal responsibility for writing *Introduction to Computer Graphics*, an undergraduate adaptation of the well-known book *Computer Graphics: Principles and Practice* by Foley, van Dam, Feiner and Hughes.

7. I retired from the University of Michigan in 1987 and joined the Computer Division of Los Alamos National Laboratory as a staff researcher. There I did research on user interface design and development, scientific visualization, multimedia systems and Internet-based telemedicine systems.

8. I have been active in many professional organizations, especially SIGGRAPH, the premiere scientific organization for computer graphics professionals. Besides contributing to the SIGGRAPH technical program over the years, I have held positions on the SIGGRAPH Executive Committee and on several SIGGRAPH Conference Committees. For example, I was a member of the Papers Selection Committee for SIGGRAPH '77 in San Jose and I was the technical program chair for SIGGRAPH '78 in Atlanta. I was also the panels' chair for both SIGGRAPH '86 in Dallas and SIGGRAPH '88 in Atlanta. In addition, I developed the introductory "Computer Graphics Fundamentals" course and presented it at SIGGRAPH '87 in Anaheim, SIGGRAPH '88 in Atlanta and SIGGRAPH '89 in Boston. That course became a regular event at succeeding SIGGRAPH conferences.

9. In the field of computer graphics, I have also conducted research in areas including graphics systems software development, distributed scientific visualization, computer animated film production, interactive environmental

mapping, computer aided design, database management systems, interactive query languages, low cost computer graphics systems, raster graphics algorithms, control console displays for teleconferencing sites, business presentation display systems, interactive painting systems, distributed graphics systems, graphics-based user interface management systems, window systems, multimedia digital publication, and interactive medical record systems.

### **3. SOFTWARE DEVELOPMENT EXPERIENCE**

10. I have been a hands-on programmer throughout my career and have participated in the development of a wide variety of large scale software systems. Since my career began there has seldom been a time that I have not been involved in a programming project. I'll highlight just a few of those projects to give an idea of the diversity of programming languages and operating systems I've worked with over the years.

11. My experience in the design and programming of large scale, complex computer systems began in 1963, during the course of my dissertation research. At that time I programmed in MAD (Michigan Algorithmic Decoder), one of the first block structured languages. As part of that project I developed an early technique for producing computer animated films.

12. In later work, I programmed in FORTRAN and IBM 360 assembly language to produce a specialized interpretive language for time series analysis and graphical display of environmental data, such as air and water quality. That system comprised over 300,000 lines of FORTRAN. Another large system written in FORTRAN was developed by me for the US Geological Survey to maintain a database of off-shore oil leases. The system featured an interactive query language and a thematic mapping facility.

13. In 1985, I designed and programmed a system to allow Macintosh computers to function as satellite terminals to Apollo workstations. That system,



programmed in Pascal and running under UNIX, featured a graphics interpreter running on a Macintosh that displayed graphical information arising from a computation running on the (then) more powerful Apollo workstation [Phillips86].

14. In [Phillips89] I reported on the Scientist's Workbench, a distributed scientific visualization system that I wrote which featured a high speed connection between a Cray computer and a Sun workstation. Using the facilities of Sun's NeWS window system, the Scientist's Workbench allowed a user to view an evolving Cray-based simulation on a remote workstation. Because NeWS was Postscript-based, much of the Scientist's Workbench was written in about 10,000 lines of Postscript.

15. Beginning in 1990, I designed and developed a large system which presaged Apple's OpenDoc technology (See, e.g., [OpenDoc97]). The system was called MediaView [Phillips91] and comprised over 50,000 lines of Objective C code running on a Mach/UNIX platform.

16. Then, in the 1994 - 1995 timeframe I was the chief architect and senior programmer for a multimedia distributed telemedicine system [Phillips97]. The system, called TeleMed, featured access to object oriented databases through CORBA-compliant object request brokers. The system was programmed in Java and C++ and ran on UNIX and Windows platforms.

17. Finally, throughout my career, I've worked in not just high level programming languages, but in assembly languages as well. I'll summarize much of that experience by describing one recurring project, a Tektronix terminal emulator for several raster graphics systems. I first wrote that system for an Apple II in 1978. It comprised 10,000 lines of assembly code for a Motorola 6502 processor. Subsequent rewritings of the emulator were for a Zilog Z8000, an Intel 8088 and a Texas Instruments TI 7700.

18. In summary, I've had over 40 years of programming experience, both in high level and low level languages, on a wide variety of operating systems.

## II. Information Considered In Forming My Opinions

19. In forming my opinions for this case, I considered the following information:<sup>3</sup>

### 1. PATENTS IN SUIT AND DECLARATIONS

- [Doyle-906] Doyle, Michael D., Martin, David C., and Ang, Cheong S., US Patent 5,838,906, Distributed Hypermedia Method and System for Automatically Invoking External Application Providing Interaction and Display of Embedded Objects Within a Hypermedia Document, November 17, 1998. Also referred to as "'906" or "'906 patent."
- [Doyle-985] Doyle, Michael, Martin, David, and Ang, Cheong, US Patent 7,599,985 B2, Distributed Hypermedia Method and System for Automatically Invoking External Application Providing Interaction and Display of Embedded Objects Within a Hypermedia Document, October 6, 2009. Also referred to as "'985" or "'985 patent."
- [Sadowski11] Declaration of Daniel Sadowski, Appendix D to my earlier report.

### 2. USPTO PROSECUTION HISTORIES

- ['906-PH] '906 History  
PH\_001\_0000775641 - PH\_001\_0000776910; see also PH\_001\_0000783777 - PH\_001\_0000784194
- ['985-PH] '985 History  
PH\_001\_0000000001 - PH\_001\_0000588348; see also PH\_001\_0000784195 - PH\_001\_0000784758
- ['831-PH] '831 (Director's) Reexamination History  
PH\_001\_0000779756 - PH\_001\_0000782282; see also PH\_001\_0000784759 - PH\_001\_0000786088

---

<sup>3</sup> I have tried to include in this listing all the materials that I considered in preparing this report. However, other materials not listed in this section but cited elsewhere in this report and its accompanying materials were also considered.

['858-PH] '858 (Third Party) Reexamination History  
PH\_001\_0000588349 - PH\_001\_0000775640; see also PH\_001\_0000786089 -  
PH\_001\_0000787424

['563-PH] '563 (Koppolu) Interference  
PH\_001\_0000776911 - PH\_001\_0000779016; see also PH\_001\_0000787425 -  
PH\_001\_0000787692

### 3. PUBLICATIONS AND OTHER REFERENCES

- [Foley90] Foley, J., A. van Dam, S. Feiner and J. Hughes, *Computer Graphics: Principles and Practice, Second Edition*, Addison-Wesley, Reading, MA, 1990.
- [Foley93] Foley, J., A. van Dam, S. Feiner, J. Hughes and R. Phillips, *Introduction to Computer Graphics*, Addison-Wesley, Reading, MA, 1993.
- [Forslund92] High-Speed Networks, Visualization, and Massive Parallelism in the Advanced Computing Laboratory, *Computing Systems in Engineering*, Vol. 3, Nos. 1-4, pp. 521-524, 1992. [ADBE0196687]
- [Phillips77] "A Query Language for a Network Database with Graphical Entities", presented at the 4th Annual SIGGRAPH (ACM) Conference on Computer Graphics, San Jose, CA, July 1977. [PA-00334339]
- [Phillips86] A Bridge from Full-function to Reduced-function Workstations, with co-authors, *IEEE Computer Graphics & Applications*, May, 1986. [PA-00273154]
- [Phillips88] A Scientific Visualization Workbench, *Proc. Supercomputing'88*, Orlando, FL, November 1988. [PA-00273159]
- [Phillips89a] Distributed Visualization at Los Alamos National Laboratory, *Computer*, August, 1989, 70-77. [PA-00273167]
- [Phillips89b] Interactive SIGGRAPH Proceedings: A New Form of Publication, Presented at EDUCOM'89, Ann Arbor, MI, October 15-18, 1989. Also reprinted in *Computer Graphics*, Vol. 24, January, 1990. [PA-00334346]

- [Phillips90a] Interactive SIGGRAPH Proceedings: A New Form of Publication, Computer Graphics, Vol. 24, No. 1, January 1990, pp. 60-61 (presented at EDUCOM '89). [ADBE0183270]
- [Phillips90b] An Interpersonal Multimedia Visualization System, Submitted to Visualization '90, San Francisco, CA, October 22-26, 1990. [ADBE 0196690]; see also An Interpersonal Multimedia Visualization System, VIS '90 Proceedings of the 1<sup>st</sup> Conference on Visualization '90, IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 338-341. [ADBE0196707 and ADBE0196688 (re-publication from Department of Energy)]
- [Phillips90c] Digital Publication: Status, Opportunities and Problems, SIGGRAPH '90, August 6-10, Dallas, TX, 16-1-16-22 (demonstration of MediaView at 16-19) [RLP 1 (SG'90 dig pub.pdf)]
- [Phillips91a] An Interpersonal Multimedia Visualization System, IEEE Computer Graphics & Applications, May, 1991. [PA-00273175 also at RLP 1]
- [Phillips91b] MediaView: An Editable Multimedia Publishing System Developed with An Object-Oriented Toolkit, Proc. USENIX Summer Conf., Nashville, TN, June, 1991. [PA-00327398 also at RLP 1]
- [Phillips91c] MediaView: A General Multimedia Digital Publication System, Comm. ACM, Vol. 34, No. 7, July, 1991. [PA-00273194 also at RLP 1]
- [Phillips91d] "How to Write Custom Modules for MediaView", distributed by request to author, 1991 and included with [LANL92]. [PA-00327441 also at RLP 1]
- [Phillips92] A Fast and Accurate Light Reflection Model, with co-authors, Computer Graphics, Vol. 26, No. 2, July, 1992. [ADBE0195786]
- [He91] A Comprehensive Physical Model for Light Reflection, Computer Graphics, Vol. 25, No. 4, July 1991, pp. 175-186 [ADBE0196695]
- [LANL93] MediaView IS-9 Production, 3/20/93, VIDEO-93-121 VT. [RLP 7]. See also Department of Energy, National Nuclear Security Administration, Service Center letter October 28, 2010 regarding FOIA Request dated May 28, 2006 (Films Report – Summary of all Films at page 112 of 220, item 1735). [ADBE0196693]
- [Umar93] Distributed Computing: A Practical Synthesis, Prentice Hall, New York, 1993, pp. 736, ISBN 0-13-036252-2. [ADBE0196711 (previously produced as ADBE0196688)]
- [Studt94] Multimedia Allows Researchers to Interact With Their Data, R&D Magazine, Vol. 36, No. 8, July 1994, pp. 35-36 and cover. [RLP 2-6]
- [Phillips94] A Network-Based Distributed, Media-Rich Computing and Information Environment, Supercomputing '94 Conference, Digital

- Media & Electronic Publishing Conference, Leeds, United Kingdom, December 6-8, 1994. [ADBE0183250]
- [Earnshaw96] Digital Media and Electronic Publishing, Academic Press Inc., San Diego, CA, 1996, pp. 219. [ADBE0196687 and RLP 9]
- [Heller91] Motif Programming Manual, O'Reilly & Assoc., 1991. [PA-00285473]
- [Stevens90] UNIX Network Programming, Prentice-Hall, 1990. [PA-00282885.]
- [Sun91] Introduction to the ToolTalk Service, Sun Microsystems, 1991. [PA-00195040]
- [Swaine91] "Applications Are Talking Too", MacUser, May, 1991. [PA-00289538]
- [Microsoft92a] Object Linking and Embedding Programmer's Reference, Microsoft Press, 1992. [PA-00274385]
- [Eckerson93] "Microsoft brings object technology to the mainstream" [PA-00277077]
- [Microsoft92b] Rich Text Format (RTF) Specification, Application Note, June, 1992 [PA-00260440]
- [Microsoft93a] "Microsoft OLE Today and Tomorrow", Microsoft, December 1993. [PA-00329002]
- [Koppolu97] U.S. Patent No. 5,613,058 to Koppolu et al. [PA-00278837]
- [Quercia91] X Window System User's Guide, O'Reilly & Assoc., 1991. [PA-00284769]
- [Nye93] X Toolkit Intrinsics Programming Manual, O'Reilly & Assoc., 1993. [PA-00286486]
- [Adie93] "Network Access to Multimedia Information," RARE, 1993. [PA-00329903]
- [Raggett93a] "HTML+ (Hypertext markup language), Hewlett-Packard, 1993 [PA-00321233]
- [Raggett93b] "HTML+ support for eqn & Postscript," email message to www-talk dated June 14, 1993. [PA-00321230]
- [Janssen93] "HTML+ support for eqn & Postscript," email message to www-talk dated June 14, 1993. [PA-00321229]
- [Shneiderman92] Hyperties Author's Guide, Cognetics, 1992. [PA-00270936]
- [Shneiderman91] Designing to Facilitate Browsing: A Look Back at the Hyperties Workstation Browser. [PA-00270916]

- [Cohen94] Cohen et al., US Patent 5,367,621, Data Processing Method To Provide A Generalized Link From A Reference Point In An On-Line Book To An Arbitrary Multimedia Object Which Can Be Dynamically Updated, November 22, 1994. [PA-00319424]
- [NeXT93] "Distributed Objects," Develop Documentation Manuals, NeXT Computer, 1993. [ADBE0195785]
- [NeXT89] The NeXT System Reference Manual, Part 2: Reference, Product #N6007, NeXT, Computer, 1989. Produced at [ADBE0195787].
- [NeXT92] Questions and Answers Regarding NeXTSTEP Release 3.0 (downloadable from [http://www.kevra.org/TheBestOfNext/NeXTProducts/NeXTSoftware/QandA-NSRelease3/files/page621\\_1.pdf](http://www.kevra.org/TheBestOfNext/NeXTProducts/NeXTSoftware/QandA-NSRelease3/files/page621_1.pdf)) [PA-00334325]
- [NeXT93a] 3D Graphics Kit Overviews [ADBE0195784]
- [Andreessen93a] "NCSA Mosaic for X 2.0 available", WWW-Talk, Oct-Dec, 1993 [PA-00292659]
- [Andreessen93b] NCSA Mosaic Technical Summary [PA-00292824]
- [Wolfram88] Wolfram, S., *Mathematica: A System for Doing Mathematics by Computer*, Addison-Wesley Reading Mass, 1988. [PA-00333548]
- [Ang94] Ang, C., Martin, D., Doyle, M., "Integrated Control of Distributed Volume Visualization Through the World-Wide-Web", IEEE Vis94, 1994. [PA-00300269]
- [Lin92] Lin, Jin-Kun, MediaMosaic – A Multimedia Editing Environment, UIST'92, November, 1992. [PA-00195287]
- [Collage92] NCSA Collage for the Macintosh version 1.0, October 1992 [PA-00292677]
- [Wei94] Pei Wei's publication "A Brief Overview of the Viola Engine, and its Applications" bearing the date August 16, 1994. Versions of this paper are available at [PA-00318355] and at [PA-00318383]. In addition, a version of this paper is publically available at <http://www.viola.org/viola/violaIntro.html>. In addition, copies of other publications of this paper appear at PA-NAT-78\1993-05-27 DX37 - Ex AE to Inv Contentions\violaTOGO.tar.z \violaTOGO\docs; [Viola-alpha] at violaTOGO\docs\viola; [Viola-Feb Beta] at viola\violaDocs\vw; [Viola-March Beta] at \violaDocs\vw.
- [Goodman90] The Complete HyperCard 2.0 Handbook, 3<sup>rd</sup> Edition [PA-00288603]
- [Powers90] John R. Powers, III, "Mac to Mainframe with HyperCard." MacTutor, June 1990 [PA-00288589]

- [Morgan92] Eric Lease Morgan, "Implementing TCP/IP Communications with HyperCard." Information Technology and Libraries, Dec. 1992; 11, 4; ABI/INFORM Global. pp. 421-432. [PA-00290482]
- [Drucker92] David L. Drucker and Michael D. Murie, "QuickTime Handbook: The Complete Guide to Mac Movie Making." August 1992. [PA-00289694]
- [Borrell93] Jerry Borrell et al., "Mastering the World of Quicktime." First Edition, Random House, 1993. [PA-00328099]
- [Thomas85] Diamond: A Multimedia Message System Build on a Distributed Architecture [PA-00332212]
- [Cailliau00] How the Web was Born [PA-00320294]
- [Branwyn94] Mosaic Quick Tour for Windows [PA-00320977]
- [Bradley92] Interactive Image Display for the X Window System – Technical Report [PA-00321299].
- [Wei94b] Stanford Computer Forum WWW Workshop, September 20-21, 1994 [PA-00293130]
- [Wei-00318312] "weekly viola status report" [PA-00318312]
- [FTP92] Log of FTP activity. [PA-00313878.pdf]
- [Doyle93] Michael D. Doyle, Cheong Ang, Rakesh Raju, Gary Klein, Betsey S. Williams, Thomas DeFanti, Arsdeshir Goshtasby, Robert Grzesczuk, and Adrienne Noe. "Processing of Cross-Sectional Image Data for Reconstruction of Human Developmental Anatomy from Museum Specimens." SIGBIO Newsletter. pp. 9-15. February, 1993 [PH\_001\_0000041528]
- [Doyle94] Medicine Meets Virtual Reality II: Interactive Technology & Healthcare. The Virtual Embryo: VR Applications in Human Developmental Anatomy. [PH\_001\_0000759332]
- [W3C-overview] An Overview of Hypertext and IR systems and applications. From <http://www.w3.org/History/19921103-hypertext/hypertext/Products/>. [PA-00334369]
- [W3C-Index] Index of /History/19921103-hypertext/hypertext. From <http://www.w3.org/History/19921103-hypertext/hypertext/>. [PA-00334365]
- [MediaView-Purdue] MediaView download instructions. From <http://www.w3.org/History/19921103-hypertext/hypertext/Products/MediaView/Mail.html>. [PA-00334323]

- [Viola-Montage] Montage of 1991 Viola releases, from <http://scam.xcf.berkeley.edu/~wei/viola/vintage/montage.html>. [PA-00313845]
- [Hoffert91] Eric M. Hoffert and Greg Gretsch, "The Digital News System at EDUCOM: A Convergence of interactive Computing, Newspapers, Television and High-Speed Networks." Communications of the ACT, Vol. 34, No. 4, April 1991, pp. 113-116. [PA-00289534].
- [KenDoyle94] "The QuickTime XCMDs." QuickTime Software Group, Apple Computer, Inc., April 26, 1994. [PA-00333243].
- [Adobe93] Tim Bienz and Richard Cohn, Adobe Systems Incorporated, "Portable Document Format Reference Manual," Addison-Wesley Publishing Co., 1993 [ADBE0195521]
- [HyperCardBasics90] HyperCard Basics, Apple Computer, Inc., 1990 [PA-00333508]
- [OpenDoc97] U.S. Patent No. 5,669,005 to Curbow et al., "System for Automatically Embedding or Incorporating Contents Added to a Document" [PA-00329032]
- [Reynolds92] Louis R. Reynolds and Steven J. DeRose, Electronic Books, BYTE, Vol. 17, No. 6, McGraw-Hill., New York, June 1992, pp. 263-268. [PA-00334349]
- [Haan92] IRIS Hypermedia Services, Comm. ACM, Vol. 35, No. 1, January 1992, pp. 36-51 [ADBE018751 (IRIS Hypermedia.pdf)]
- [Meyrowitz82] Interactive Editing Systems: Part II, ACM Computing Surveys, Vol. 14, No. 3, September 1982, pp. 353-414. [ADBE018751 (p353-meyrowitz.pdf)]
- [VanDam88] Hypertext '87 Keynote Address, Comm. ACM, Vol. 31, No. 7, July 1988, pp. 887-895. [ADBE018751 (p887-van\_dam.pdf)]
- [Ohtsu93] User Interface Management System Embedded in a Multimedia Document Editor Framework, EECS Dept., University of California, Berkeley, Technical Report No. UCB/CSD-93-773, October 1993, also Foundations of Data Organization and Algorithms, 4<sup>th</sup> Int'l Conference, FODO'93, Chicago, IL, October 13-15, Proceedings 1993.
- [Rowe92] A Continuous Media Player, Proc. 3<sup>rd</sup> Int. Workshop on Network and OS Support for Digital Audio and Video, San Diego, CA, November 1992.
- [Hindus93] Capturing, Structuring, and Representing Ubiquitous Audio, ACM Transactions of Information Systems, Vol. 11, No. 4, October 1993, pp. 376-400.



[Hindus92] Ubiquitous Audio: Capturing Spontaneous Collaboration, CSCW '92 Proceedings, ACM, November 1992, pp. 210-217.

Adobe Non-Disclosure Agreement, 1993 [ADBE0195776-777]

MacroMind Play HyperCard Manual, 1993 [ADBE0196003-061]

Bob Wulff presentation [ADBE0196062-071]

Adobe Acrobat Reader Program On-Line Guide, 1993 [ADBE0196072-113]

Adobe Acrobat Exchange Program On-Line Guide, 1993 [ADBE0196114-178]

#### 4. SOFTWARE

[SunOS 4.1.3] SunOS 4.1.3. [PA-NAT-00000029].

[NeXT] NeXTSTEP release 3.2, released in October 1993. Available for inspection by contacting counsel of record for Adobe Systems, Inc.

[Viola-5/12/93] From [PA-NAT-78\viola\1993-05-12 DX34 - Ex A to Inv Contentions \software\viola930512.tar.gz]

[Viola-5/27/93] From [PA-NAT-78\viola\1993-05-27 DX37 - Ex AE to Inv Contentions\violaTOGO.tar.z]

[Viola-Alpha] Viola alpha, dated October 16, 1993 [PA-NAT-78\viola\1993-10-16 Alpha release - Ex AF to Inv Contentions\violaTOGO.tar.z]

[Viola-Feb Beta] Viola February Beta, dated February 23, 1994 [PA-NAT-78\viola\1994-02-23 February beta release - Ex AG to Inv Contentions]

[Viola-March Beta] Viola March Beta, dated March 23, 1994 [PA-NAT-78\viola\1994-03-23 March beta release - Ex AH to Inv Contentions]

[Vplot-Sun] PA-NAT-78\vplot\Tar Secondaries\000007\_usr\_users\scott\vplot

[Vplot-petra] PA-NAT-78\vplot\MS\_SUPP\_1205\_001\petra

[Xplot] Xplot software obtained from 1 - TAR Secondaries\047263-000006\usr\work\xplot\xplot and dated May 6, 1993

[XV] XV, obtained from ftp://ftp.ncsa.uiuc.edu/Mosaic/Unix/viewers/xv-3.00.tar.Z and dated April 28, 1993.

[HyperCard] HyperCard version 2.1, obtained from [PA-NAT-00000074]

HyperCard version 2.1, including PowerTools with Resource Mover, obtained from [PA-NAT-00000010]

- See also* HyperCard 2.0 [PA-NAT-00000067]; HyperCard 2.2 [PA-NAT-00000009]
- [QuickTime] QuickTime version 1.0 software (*see, e.g.*, QuickTime 1.0 Developer's CD [PA-NAT-00000074])
- QuickTime version 2.0 software (*see, e.g.*, Quicktime: The Official Guide for Macintosh Users [PA-NAT-00000077])
- QTMovie XCMD
- See, e.g.*, Quicktime Version 1.0 for Developers [PA-NAT-00000074], at: Programming Stuff:XCMDs:QTMovie Stack, dated December 8, 1991.
- See, e.g.*, Apple Chronicle CD [PA-NAT-00000068; PA-NAT-00000069] at: Interactive Speech Systems:SE stack, dated March 19, 1992
- See, e.g.*, :Unsupported Stuff:XCMDs:QTMovie Stack in "QuickTime: The Official Guide for Macintosh Users," dated April 26, 1994. [PA-NAT-00000077]
- MOVIE XCMD
- See, e.g.*, Movie XCMD, obtained from QuickTime Tools in Claris HyperCard 2.1, from 1991 [PA-NAT-00000010]
- [Director] Director software referenced in [Sadowski11], including Director 3.1.3 software, and the Macromind Player stack documentation that shipped with Director 3.1.3. Available for inspection by contacting counsel of record for Adobe Systems Inc.
- [Mosaic] Mosaic version 2.4. *See, e.g.*, PA-NAT-00000044 ("Mosaic 2.4"), PA-NAT-00000054 ("Mosaic-Sun 2.4"), [PA-NAT-00000024] at 047263-000004\net\marble\usr\users\mcrae\src\www\mosaic\Mosaic-sun
- See also, e.g.*, PA-NAT-00000045 and PA-NAT-00000046 ("Mosaic 2 prerelease 5"); PA-NAT-00000047 ("Mosaic 2 prerelease 1"); PA-NAT-00000048 ("Mosaic 2 prerelease 2"); PA-NAT-00000049 ("Mosaic 2 prerelease 3"); PA-NAT-00000050 ("Mosaic 2 prerelease 4"); PA-NAT-00000051 ("Mosaic 1.2"); PA-NAT-00000052 ("dandelionpatch.mit.edu - Mosaic 2.1"); PA-NAT-00000054 ("DX227 Mosaic 2.4").
- [MediaView] MediaView version 3.0 [PA-NAT-00000071] and subsequent production and materials on NeXT cube (see below).
- [LANL92] "LANL Software and Visualization Sampler," Los Alamos National Laboratory. Ed. Richard L. Phillips. [DS 1]

- [VIS] VIS executable obtained from [PA-NAT-00000060]
- [OLE] Microsoft Word Version 6.0a © 1983-1994 Microsoft Corporation and Microsoft Excel Ver. 5.0a © 1985-1993 Microsoft Corporation. Available for inspection by contacting counsel of record for Adobe Systems Inc., Apple Computer, Inc., and Google, Inc.
- [MMPEG] MMPEG software, obtained from PA-NAT-00000060 and dated 10/31/1994

## 5. COMPUTERS

Three Sun Sparcstations (model 10), dated 1992, each running SunOS version 4.1.3, dated 1992, and connected using Ethernet networking. Available for inspection by contacting counsel of record for Apple, Inc., Adobe Systems Inc., or Google Inc.

Two Apple Macintosh Model IISI computers, dated 1990, running System 7.1, dated 1992, connected using Ethertalk networking. Available for inspection by contacting counsel of record for Apple, Inc., Adobe Systems Inc., or Google Inc.

One "NeXT cube" computer, dated 1992, with a Motorola 68040 cpu, running NeXTSTEP release 3.2, dated October 1993. Available for inspection by contacting counsel of record for Adobe Systems Inc. or Google Inc.

## 6. WWW-TALK PUBLICATIONS AND OTHER EMAILS

www-talk-00293029 1991-12-13 1145 Wei Email to www-talk re: X Browser (PA-00293029)

www-talk-00318887 1993-01-30 1656 Andreessen Email to www-talk re: Solicitation for Widgets (PA-00318887)

www-talk-00293041 1993-02-08 1723 Wei Email to marca@ncsa.uiuc.edu re: Stuff in New violaWWW (PA-00293041)

www-talk-00293039 1993-02-08 1755 Andreessen Email to Wei re: Stuff in New violaWWW (PA-00293039 - 040)

www-talk-00292995 1993-02-26 1028 Weber Email to Andreessen re: Proposed New Tag: IMG (PA-00292995)

www-talk-00292991 1993-02-26 1404 Lee Email to Andreessen re: Proposed New Tag: IMG (PA-00292991)

www-talk-00293002 1993-03-12 2232 Andreessen Email to Lee re: Proposed New Tag: IMG (PA-00293002 - 003)

www-talk-00293011 1993-04-29 1217 Janssen Email to Sanders re: Standardizing New HTML Features (PA-00293011)

www-talk-00293014 1993-04-29 1506 Janssen Email to Sanders re: Standardizing New HTML Features (PA-00293014)

www-talk-00293059 1993-05-10 1855 Wei Email to dsr@hplb.hpl.hp.com re: HMML (PA-00293059 - 069)

www-talk-00293081 1993-06-19 announcement by Dale Dougherty re: WWW Developer's Conference (PA-00293081)

www-talk-00293020 1993-06-26 1309 McRae Email to Raisch re: Xmosaic and Xv (PA-00293020 - 021)

www-talk-00292656 1993-10-10 2245 Andreessen to www-talk re: NCSA Mosaic for X 2.0 Prerelease 5 available (PA-00292656)

www-talk-00028138 1994-01-24 1156 McRae Email to Albilis and www-talk re: Modifying Xmosaic (PA-00028138)

www-talk-00028162 1994-01-25 1229 Martin Email to Mittelhauser and www-talk re: Inlined Image Format (PA-00028162 - 163)

www-talk-00028157 1994-01-25 1445 Ellson Email to www-talk re: Inlined Image

Format (PA-00028157)

www-talk-00028167 1994-01-25 1620 Ellson Email to www-talk re: Inlined Image Format (PA-00028167 - 168)

www-talk-00293096 1994-01-28 0802 Wei Email to www-talk re: Universal Network Graphics Language (PA-00293096 - 097)

www-talk-00318314 1994-01-28 0809 Wei Email to www-talk re: Universal Network Graphics Language (PA-00318314 - 316)

www-talk-00293098 1994-02-25 1313 Wei to www-talk re: ViolaWWW Beta Release is Available (PA-00293098 - 099)

www-talk-00293100 1994-03-24 Wei Email to www-talk re: ViolaWWW Release (PA-00293100)

www-talk-00293120 1994-08-31 1347 Doyle Email to Doyle re: violaWWW Embedding in Docs for Months (PA-00293120 - 121)

www-talk-00293122 1994-08-31 1852 Wei Email to www-vrml@wired.com re: violaWWW Embedding in Docs for Months (PA-00293122)

www-talk-00293119 1994-08-31 2106 Doyle Email to Wei re ViolaWWW Embedding in Docs for Months (PA-00293119)

www-talk-00293126 1994-08-31 2316 Wei Email to Doyle re: violaWWW Embedding in Docs for Months (PA-00293126 - 127)

www-talk-00293117 1994-08-31 2336 Doyle Email to Wei re: Publish Viola Embedding Objects Findings (PA-00293117 - 118)

www-talk-00293124 1994-09-01 0438 Doyle Email to Wei re: Modified X Mosaic Clients for Sun and SGI (PA-00293124 - 125)

www-talk-00293128 1994-09-01 0819 Wei Email to Doyle re: Viola vs. Embedded Program Objects (PA-00293128 - 129)

www-talk-00293123 1994-09-01 1658 Doyle Email to Foster re: VRML (PA-00293123)

www-talk-00318682 1994-09-14 Stanford Computer Forum Sept. 20-21, 1994 Meeting Agenda (PA-00318682 - 683)

www-talk-00293140 1995-08-21 1602 Wei Email to www.talk re: Eolas Acquires Milestone Internet Software Patent (PA-00293140)

www-talk-00293138 1995-08-21 1609 Wei Email to Doyle re: Eolas Acquires Milestone Internet Software Patent (PA-00293138 - 139)

www-talk-00293130 1998-05-28 Wei Stanford Computer Forum WWW Workshop - September 20-21 re: WWW Browsers Extensibility Issues (PA-00293130 - 137)

ora-00293042 1993-05-04 1700 Dougherty Email to Wei re: Viola Demonstration (PA-00293042)

ora-00293056 1993-05-08 1321 Dougherty Email to Wei re: DMG Status Report (PA-00293056 - 058)

ora-00293071 1993-05-31 1604 Wei Email to Kempf re: Viola Tar (PA-00293071 - 072)

ora-00293070 1993-05-31 1623 Wei Email to Kempf re: Viola.tar.z for FTPing (PA-00293070)

ora-00293074 1993-06-03 1022 Dougherty Email to Wei re: Viola.tar for SUN (PA-00293074 - 080)

ora-00318281 1993-06-17 1601 Wei email to hackers@ora.com re: Viola status report.

ora-00293084 1993-10-14 1824 Wei Email to Dougherty re: Viola Alpha Status (PA-00293084 - 085)

ora-00293086 1993-10-15 1259 Silvey Email to Wei re: First Alpha Release to Go Around ORA (PA-00293086)

ora-00293088 1993-10-16 1009 Wei Email to hackers@ora.com re: Viola Alpha Update (PA-00293088)

ora-00293089 1993-10-17 0615 Wei Email to weber@eit.com re: violaWWW (PA-00293089)

ora-00293090 1993-10-18 1652 Bluming Email to Wei re: Viola Questions and Praise (PA-00293090 - 091)

[McRae93] Email from Christopher Rae to David Martin, both of UCSF, re: More Hypertext Systems (September 8, 1993) [PA-00334406]

Email from Christopher McRae to John Dawes et al. re: Acrobat and UCSF (May 14, 1993) [PA-00334379]

Email from John Dawes to Adobe et al. re: Acrobat and UCSF (May 17, 1993) [PA-00334381]

Email from Christopher McRae to David Martin et al. re: Adobe Acrobat Beta Agreement (June 2, 1993) ("1993-06-02 1004 email from McRae-McRae") [PA-00334383]

Email from Christopher McRae to David Martin, CC: Michael Doyle al. re: Adobe Acrobat Beta Agreement (June 18, 1993) ("1993-06-18 1355 email from McRae-McRae") [PA-00334384]

Email from David Martin to Christopher McRae re: Adobe Acrobat Beta Agreement (June 21, 1993) ("1993-06-21 0920 email from Martin-McRae") [PA-00334385]

Email from Christopher McRae to www-talk re: HTML Spec (June 21, 1993) ("1993-06-21 1557 email from McRae-McRae") [PA-00334386]

Email from Robert Cailliau to www-talk et al. re: WWWWorkshop (July 16, 1993) ("1993-07-16 0950 email from Cailliau-McRae") [PA-00334387]

Email from Daniel Kehoe to www-talk et al. re: Adobe's PDF (July 16, 1993) ("1993-07-16 1812 email from Kehoe-McRae") [PA-00334389]

Email from Kevin Altis to www-talk et al. re: Adobe's PDF (July 19, 1993) ("1993-07-19 1107 email from Altis-McRae") [PA-00334392]

Email from Daniel Kehoe to www-talk et al. re: Adobe's PDF (July 19, 1993) ("1993-07-19 1309 email from Kehoe-McRae") [PA-00334395]; [PA-00334397]

Email from David Martin to Marc Solomon, CC: Michael Doyle et al. re: Adobe's PDF (July 19, 1993) ("1993-07-19 0855 email from Martin-McRae")

Email from Kevin Altis to www-talk et al. re: Adobe's PDF (July 19, 1993) ("1993-07-19 1107 email from Altis-McRae") [PA-00334396]

Email from Bill Janssen to www-talk et al. re: Adobe's PDF (July 19, 1993) ("1993-07-19 1552 email from Janssen-McRae") [PA-00334394]

Email from Daniel Kehoe to www-talk et al. re: Adobe's PDF (July 19, 1993) ("1993-07-19 1309 email from Kehoe-McRae") [PA-00334397]; [PA-00334395]

Email from Steve Heaney to www-talk re: HTML+ Comments (July 20, 1993) ("1993-07-20 1848 email from Heaney-McRae") [PA-00334399]

Email from David Martin to www-talk et al. re: Adobe's PDF (July 20, 1993) ("1993-07-20 1314 email from Martin-McRae") [PA-00334402]

Email from David Martin to UCSF re: Agenda for 08-19 Meeting (August 13, 1993) ("1993-08-13 1905 email from Martin-McRae") [PA-00334404]

Email from Marc Andreessen to www-talk et al. re: MIF and PDF Mime Types (October 23, 1993) ("1993-10-23-1646 email from Andreessen-McRae") [PA-00334405]

Matthey Grey's post, dated August 19, 1993, and available at <http://ksi.cpsc.ucalgary.ca/archives/WWW-TALK/www-talk-1993q3.messages/736.html>.)

## 7. VIDEOS

[News Navigator] Video and presentation of 1990 EDUCOM conference in Atlanta, [PA-NAT-00000036] and [PA-NAT-00000035]

[Hardin93] The National Center for Supercomputing Applications Software Development Group presents NCSA Mosaic.

## 8. LITIGATION MATERIALS

Trial testimony from Pei Wei, Scott Silvey, and Dale Dougherty, *Microsoft Corp.*, No. 99-C-626 (N. D. Ill. Jul. 8, 2003 – Aug. 7, 2003) [EOLASTX-E-0000000644]

Excerpt of Deposition Testimony of Karl Jacob from October 4, 2001 in connection with *Eolas Technologies, Inc. v. Microsoft Corp.*, No. 99-C-626, Document # 816-6.

Deposition transcript of William Janssen (May 11, 2011) and exhibits

Eolas's Responses and Objections to Defendants' Third Set of Common Interrogatories or Eolas Technologies Incorporated (No. 6.) (related to secondary considerations of obviousness)

Deposition transcript of Charles Krueger (June 14, 2011), and exhibits.

Deposition transcripts of Michael Doyle and exhibits.

Deposition transcript of Dave Raggett (July 15, 2011) and exhibits.

Deposition transcripts of Cheong Ang and exhibits.

Deposition transcript of Eric Bina (August 2, 2011) and exhibits.

Deposition transcripts of David C. Martin and exhibits.

Deposition transcript of Scott Silvey (September 15, 2011) and exhibits.

Deposition transcript of Christopher McRae (September 19, 2011) and exhibits.

Deposition transcript of Pei (Perry) Wei (October 5, 2011) and exhibits.

Eolas's Local P. R. 3-2 exemplary infringement contentions served on Defendants, including "906 - Adobe - PDF - Authoring Tools and Players (Final).pdf," "985 - Adobe - PDF - Authoring Tools and Players (Final).pdf," "906 - Adobe - Authoring Tools and Players (Final).pdf," "985 - Adobe - Authoring Tools and Players (Final).pdf," "906 - Apple - iTunes (Final).pdf," "985 - Apple - iTunes (Final).pdf," "906 - Apple - Quicktime - Authoring Tools and Players (Final).pdf," and "985 - Apple - Quicktime - Authoring Tools and Players (Final).pdf"

*Eolas Techs, Inc. v. Microsoft Corp.*, 399 F.3d 1325 (Fed. Cir. 2005)

Infringement Report by David Martin, Ph.D. [Martin11] and exhibits and supplemental statement regarding [Martin11]



## 9. OTHER

[Viola Stuff] [PA-00318733]

[Chronicle] Apple Chronicle CD-Rom containing exemplary HyperCard content, also cited above in connection with [Quicktime] software. [PA-00000068] and [PA-00000069]

[www-conference-photos] Photos of attendees from the World-Wide Web Wizards Workshop ("WWW Wizards Conference"), which took place from Wednesday July 28, 1993 – Friday July 30, 1993, in Cambridge, Massachusetts. [PA-00293083]

[Stanford-schedule] Schedule for StanfordWWW workshop, which took place on September 20-21, 1994 [PA-00318682]

[Smithsonian91a] Responses to a series of questions from Smithsonian that I answered in 1991. [PA-00334357]

[Smithsonian91b] A scan of a formal document I received in recognition for my work in connection with MediaView in 1991 from the National Museum of American History, at the Smithsonian Institution. [PA-00334364]

20. I may rely upon these materials and/or additional materials to rebut arguments raised by Plaintiff. I have also relied on my years of education, research, and experience described above. Further, I may also consider additional documents and information in forming any necessary opinions – including documents that may not yet have been produced. Accordingly, I reserve the right to supplement my report with opinions based on newly discovered documents or any new rulings by the Court.

21. I also reserve the right to rely on other prior art that I understand was produced by Defendants individually and as part of its prior art productions, including [PA-00000001] through [PA-00333507], [PA-NAT-1] through [PA-NAT-78], the RLP productions, and productions from various party and third party prior art defendants including Messrs. Bina, Wei, Silvey, Dougherty, Berners-Lee and any other witnesses who produce materials, which of course include versions of materials cited directly in this report and additional materials supporting those items (e.g. materials related to Viola, MediaView, and the Mosaic and CERN browsers). I also

reserve the right to review and rely upon additional materials produced by the Regents of the University of California, who I understand have not completed their production of documents or witnesses as of the date of this report.

22. The following is a summary of some of the information I considered.

### **III. Tools Available To A Person of Ordinary Skill By October'93**

#### **1. Workstations by Sun, Apollo, SGI and NeXT**

23. Several workstation-class computers were available to web developers in the early to mid-1990's. Among the vendors were Sun Microsystems, Apollo Computer Inc., Silicon Graphics Inc. (SGI) and NeXT Inc. Sun's system was open in the sense they used widely accepted standard components such as BSD UNIX software and Ethernet networking. Apollo, however, elected to develop a UNIX-like operating system and used a proprietary token ring network called Domain.

24. One workstation made available by Sun was called the Sun Sparcstation. By 1992, Sun had released Model 10 of its Sparcstation. The Sun Sparcstation ran an operating system called SunOS, and by 1992, Sun had released version 4.1.3 of this operating system. As part of my review of the prior art in preparation for this expert report, I used a Sun Sparcstation Model 10 running SunOS version 4.1.3. [SunOS 4.1.3.]

25. SGI's workstations featured high performance graphics engines, which allowed real-time animation of complex datasets. The operating system was based on ATT System V and used conventional Ethernet networking. A NeXT computer, augmented with a NeXTDIMENSION high performance color subsystem was capable of rapid display of complex objects using 32 bit/pixel color. A video subsystem allowed real time input and output of NTSC video. The operating system was Mach, a UNIX look-alike and Ethernet networking. NeXT also provided a flexible, powerful software development system that allowed complex applications

to be developed in an unusually short time. Tim Berners-Lee, then of CERN, used a NeXT to develop one of the first GUI-based web browsers.

26. As part of my review of the prior art in preparation for this expert report, I used a "NeXT cube" computer from 1992, with a Motorola 68040 CPU. I used the NeXTSTEP release version 3.2, which came out in October 1993. [NeXT.]

## 2. UNIX Operating System

### a. Processes

27. On UNIX systems, each system and end-user task was contained within a process. The system created new processes all the time and processes died when a task finished or something unexpected happened. UNIX ran many tasks seemingly at the same time because each process received a little slice of CPU time in a (conceptually) round-robin fashion.

28. A process is something of a container, bundling a running application, its environment variables, the state of the application's input and output, and the state of the process, including its priority and accumulated resource usage. Finally, each process has *privileges*. Typically, a process's privileges are commensurate with those of its owner. (For instance, if you can't access a particular file from your command-line shell, programs you launch from the shell inherit the same limitation.)

29. Processes can launch other processes, giving rise to the notion of parent and child processes.

30. UNIX provided a number of technologies for *interprocess communication*. Some techniques provided for communication on the same host computer, while others facilitate host-to-host (i.e., computer-to-computer) exchanges. Among them are technologies called pipes and named pipes, for local communication and sockets and remote procedure calls for both local and remote communication.

31. An exemplary prior art reference that described UNIX processes is [Stevens90].

**b. Interprocess Communication Techniques****i. UNIX IPC Techniques**

32. Pipes were provided with all flavors of UNIX. The pipe and named pipe were simple mechanisms. Both used two standard file descriptors on each end of the connection – one for read and another for write operations. The named pipe was another choice for data exchange on the same system. Each operated as a first-in, first-out (FIFO) device.

33. A socket functioned much like a named pipe but could span hosts. *Local sockets* (also called *UNIX sockets*) were restricted to local (same host) connectivity. Sockets accepted remote connections (and local connections via the local machine's Internet addressing). The socket was a common and practical choice for use with any networking application, such as distributed processing or a web browser, during the time of the prior art.

34. Remote Procedure call (RPC) programming builds on the techniques described above and was one of the most powerful and efficient ways to ensure communication between client and server entities. It formed the basis for almost any application running on distributed computing environments. The RPC runtime library provided a standard set of runtime routines to support the RPC applications. In a general application, a called procedure ran in the same address space and results were returned to the calling procedure. When one considers a distributed environment with a client and server running on different machines, the client side calls a procedure which runs on the server side and the results are sent back to the client. This is called a remote procedure call (RPC) and formed the basis for RPC programming. One example of a function that provided for process execution on a remote server was *rexec*.

35. One exemplary reference that described UNIX interprocess communication techniques is [Stevens90].

## ii. ToolTalk

36. ToolTalk was an interapplication communications system developed by Sun Microsystems (SunSoft) in order to allow applications to communicate with each other at runtime. Applications supporting ToolTalk could construct "high-level" messages and hand them off to the system's ToolTalk server, which determined the proper recipients and (after applying permission checks) forwarded the message to them. Although originally available only on SunOS and Solaris, ToolTalk was chosen as the application framework for the Common Desktop Environment and thus became part of a number of Unix distributions as well as OpenVS.

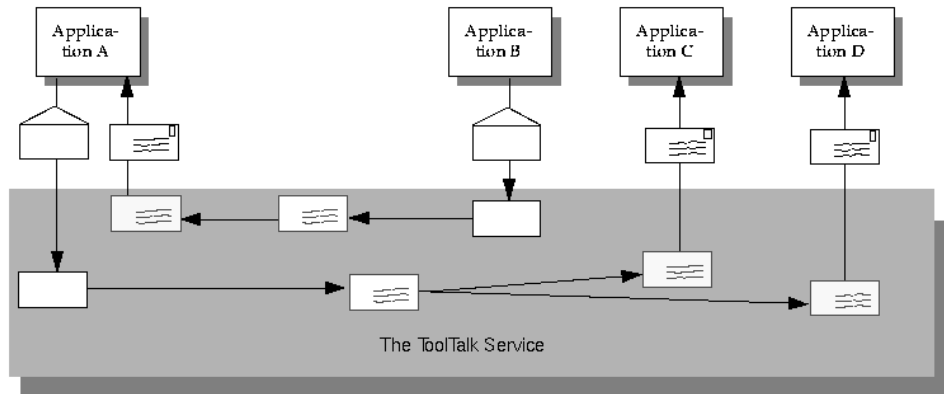
37. The ToolTalk service enabled independent applications to communicate with each other without having direct knowledge of each other. Applications created and sent ToolTalk messages to communicate with each other. The ToolTalk service received these messages, determined the recipients, and then delivered the messages to the appropriate applications, as shown in Figure 1<sup>4</sup>. The '906 patent specifies ToolTalk was used to enable communication between "Panel" and "VIS."<sup>5</sup>

38. One exemplary prior art reference describing ToolTalk is [Sun91].

---

<sup>4</sup> See also [Sun91] at Figure 1.

<sup>5</sup> Col. 16, 26-28.



**Figure 1. Overview of the ToolTalk Service**

### iii. IPC in Other Operating Systems

39. During the time of the prior art, operating systems besides UNIX typically had some form of interprocess communication. Among them was Microsoft's Object Linking and Embedding (OLE) protocol.

40. OLE was a set of extensible application protocols that generally enabled one application to use another application in a nearly-seamless manner. Applications that followed the OLE protocols could create documents that contained linked or embedded information from documents created by other applications; such a document was referred to as a container document. A container document had objects that were still connected to the original application that created them. For example, a user could link or embed a spreadsheet in a word-processing application, and keep the connection to the spreadsheet open. Other examples included interactive animations that opened within slide shows. Unlike normal cutting and pasting, the data in an OLE object could be:

- linked (where the data resides in another, separate document), or
- embedded (where the data resides in the container document).

41. In either case, the object was managed by the application that created it, not the application whose document it resided in. The applications interacted with

each other through inter-application interfaces that were well-documented in OLE programmer's manuals that persons of ordinary skill in the art could consult.

42. Objects in OLE could include text, charts, spreadsheets, bitmap pictures, vector drawings, sounds, video clips, and anything else that could be displayed or controlled by an application.

43. Exemplary prior art references describing OLE and its facility for interapplication communication and processing include [Microsoft93], [Microsoft92a], and [Eckerson93].

44. In addition, version System 7 of Apple's operating system included Interapplication Communication (IAC), which let applications communicate among themselves. IAC facilitated collaborative computing.

45. System 7.0 satisfied collaborative computing by allowing documents to incorporate links to shared chunks of information. The prior art recognized that there were different ways to implement such links. For example, Apple's model was Publish and Subscribe. In the Publish and Subscribe model, a chunk of information in one document could be marked as a "publication" and many documents could "subscribe" to it. The publication could be incorporated smoothly into each subscribing document as though it had been generated there, and when the publication was changed, it changed in each subscribing document. These changes occurred because the subscriber had a link to the original, not to the original itself.

46. An exemplary prior art reference describing inter-application communication on Macintosh System 7 was [Swaine91].

#### **iv. NeXTSTEP Distributed Objects**

47. NeXTSTEP Release 3.0 (1992) included a Distributed Objects system which provided a relatively simple way for applications to communicate with one another by allowing them to share Objective C objects, even amongst applications running on different machines across a network. They were useful for implementing client-

server and cooperative applications. The Distributed Objects system subsumed the network aspects of typical remote procedure call (RPC) programming, and allowed an application to send messages to remote objects using ordinary Objective C syntax.

48. The Distributed Objects system took the form of two classes, NXConnection and NXProxy. NXConnection objects were primarily bookkeepers that managed resources passed between applications. NXProxy objects were local objects that represented remote objects. When a remote object was passed to an application, it was passed in the form of a proxy that stood in for the remote object; messages to the proxy were forwarded to the remote object, so for most intents and purposes the proxy could be treated as though it were the object itself.

49. Interprocess communication was one tool, known and routinely used in the ordinary course of development by persons of skill in the art during the early 1990s, that was used by the prior art systems I describe below, including Mosaic, Viola, MediaView, HyperCard, and others.

### **3. X-Windows**

50. The X Window system (also known as X11 or simply X) was a computer software system and network protocol that provided a basis for graphical user interfaces (GUI) for networked computers. It created a hardware abstraction layer where software was written to use a generalized set of commands, allowing for device independence and reuse of programs on any computer that implements X. The specific look and the interaction with a GUI were determined by the choice of a widget toolkit.

51. X was an architecture-independent system for display of graphical user interfaces which allowed many people to share the processing power of a time-sharing computer; and each person used a networked terminal that had the capability to draw on the screen and accept user input.



52. X provided the basic framework, or primitives, for building such GUI environments: drawing and moving windows on the screen and interacting with a mouse and keyboard. X did not mandate the user interface — individual client programs known as window managers handled this, though even the window manager was not necessary and programs could have used X's graphical capabilities with no user interface. As such, the visual styling of X-based environments varied greatly; different programs could present radically different interfaces. X was built as an additional (application) abstraction layer on top of the operating system kernel.

53. In X, each window had a unique identification number associated with it called the *window ID* (also called the resource ID). This number could be used as a command line argument with client applications. In addition, a window ID could be passed as a parameter to executable applications that managed interactive multimedia objects, such as MMPEG and XV. This enabled a developer to embed an instance of that application into an existing X window by passing to the executable application the window ID of the target X window. In the videos that I am submitting along with this report, I show examples in which a window ID is passed to applications such as MMPEG, XV, and VIS in order to embed instances of those applications within an X window being used by another application. In my videos, the X window into which the output of the executable application is drawn is used by a browser application called Viola, but the same approach could be used to render outputs of executable applications into windows for other X applications as well.

54. Another facility in X windows was called *reparenting*. Using standard X routines, one could alter the parent-child relationship among windows in an X environment. In one usage, one could reparent an application's top-level window into another so-called "decoration window."

55. Exemplary prior art references that described X windows include [Quercia91] and [Nye93], see also Bina Tr. at 132:21-143:15 (discussing X-Windows and related tools and functionality).

56. X-Windows and its programming techniques and capabilities were tools, known and routinely used in the ordinary course by persons of skill in the art during the early 1990s and used by the prior art systems I describe below. For example, Mosaic and Viola both operated in the X environment.

## **4. Motif**

### **a. Background**

57. In general, Motif referred both to the Motif Window Manager (MWM) and the Motif toolkit, where MWM was an X window manager based on the Motif toolkit. See Figure 2<sup>6</sup>. (Mr. Bina also testified about Motif and widgets too, see, e.g., Bina Tr. at 40:8-21, 120:14-124:9, 133:15-137:8.)

---

<sup>6</sup> See also, e.g., [Heller91] at Figures 1-1 and 3-20.

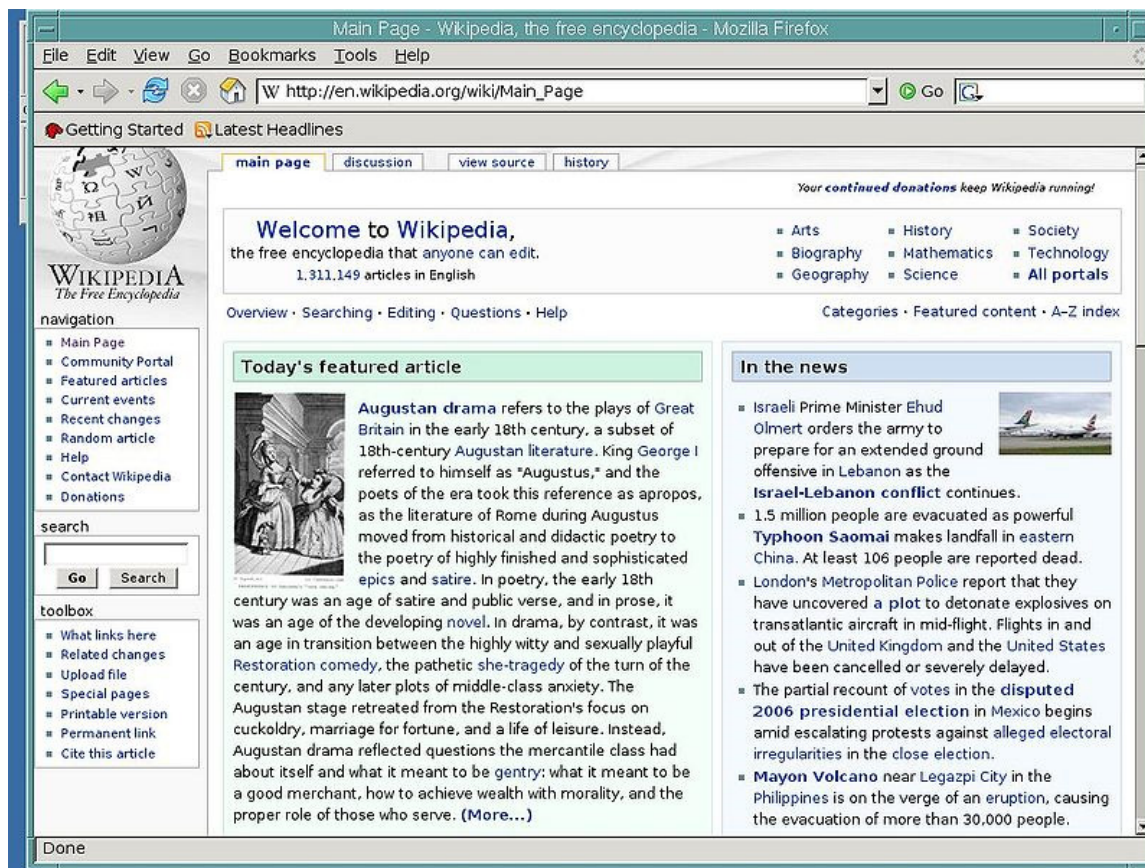


Figure 2. Motif Window Manager

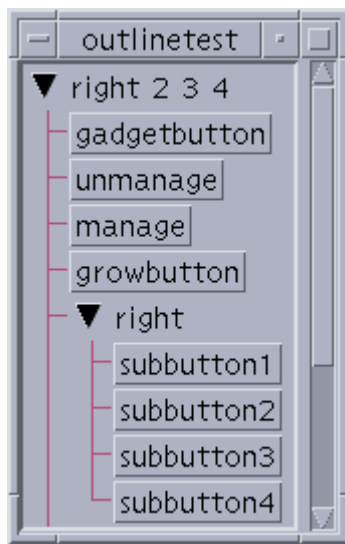
### b. Widgets

58. A widget (or control) was an element of a graphical user interface (GUI) that displayed an information arrangement changeable by the user, such as a window or a text box. A widget provided an interaction point for the direct manipulation of a given kind of data. In other words, widgets were basic visual building blocks which, combined in an application, held the data processed by applications and the available interactions on this data.

59. A widget toolkit, widget library, or GUI toolkit was a set of widgets for use in designing applications with graphical user interfaces (GUIs). The toolkit itself was a piece of software which was usually built on the top of an operating system, windowing system, or window manager and provided programs with an application programming interface (API), allowing them to make use of widgets. Each widget

facilitated user-computer interaction, and appeared as a visible part of the computer's GUI.

60. Widget toolkits also contained software to assist in the creation of window managers, as windows themselves were considered widgets. Some widgets supported interaction with the user, for example labels, buttons, check boxes, and mouse or keyboard event-handling. Others acted as containers that group the widgets added to them, for example windows, panels, and tabs.



**Figure 3. Typical Motif Widget Look**

61. The look and feel of the Motif toolkit was distinguished by its use of square, chiseled, three-dimensional effects for its various user interface elements – menus, buttons, sliders, text boxes, and the like. See Figure 3.<sup>7</sup>

62. There was a group of widgets that were extremely useful, easy to use and are worth discussing. Among these is the MainWindow widget. The MainWindow widget could be used to frame many types of applications. The MainWindow was a manager widget that provided a menubar, scrollable work area, and various other optional display and control areas. In general, a MainWindow widget was useful because one could use it to provide the overall layout to an application and then embed any kind of child widget within it. The MainWindow could perform simple widget positioning to control the sizes and positions of widget children.

63. The BulletinBoard widget was also useful because one could embed it simply by specifying the height, width, and X and Y coordinates of the embedded

<sup>7</sup> See also, e.g., [Heller91] at Figure 3-12.

widget. Thus, it would be easy to incorporate it into the aegis of a MainWindow widget.

64. Lastly, a DrawingArea widget was a valuable tool if one wanted to provide an interactive region that could accept and handle user inputs using basic X library drawing primitives. In short, the DrawingArea was a free-form widget that one could use for various interactive drawing routines or object placement. As a practical matter, this was a useful tool to support drawing from a companion application that produced interactive output from analysis of an arbitrary dataset. The '906 patent specifies that a DrawingArea widget was created to receive graphical output from an external application.<sup>8</sup>

65. An exemplary prior art reference that described Motif widgets is [Heller91].

#### **c. X Events, Callback Functions and user Interaction**

66. Once widgets have been created and configured, they can be hooked up to application functions. This could be done by means of widget resources known as callback resources. Before talking about callback resources and callback functions, though, X events will be discussed in more detail.

67. X Events were one of the major components of the X client-server protocol. In the X protocol, X clients sent *requests* to the X server. The server sent *replies*, *errors*, and *events* to its clients. The X protocol defined a large number of events. Some of the more important (to the application programmer) were:

- pointer button press events
- pointer motion events
- pointer enter and leave window events
- keyboard key press events

---

<sup>8</sup> Col. 15, 4-7.

- window configuration change (size, position) events
- window exposure events

68. Events could occur in any order, in any window, as the user moved the pointer, switched between the mouse and the keyboard, moved and resized windows, or invoked functions available through graphical user interface elements. Translations and actions allowed a widget class to define associations between events and widget functions. Some complex widgets, such as the Motif text widget, were almost applications in themselves. (Its actions provided a complete set of editing functions.)

69. The role of the callback function was to translate user interaction with a widget into a specific action in the application. As for the callback function itself, it was passed the widget, the client data, if any, and a third argument, generically referred to as "call data."

70. For example, for the event corresponding to a user pressing a button widget, the widget knew how to highlight itself but it was the data passed to the callback function that determined what application action occurred.

71. Likewise, for a text input widget, each keystroke was a separate event that passed data to its callback function. In that case, the application could elect to use each accumulated keystroke to try to match a partial word to some internal text list of certain terms. The partial match guess was displayed to the user allowing him to select the desired match and stop typing. The word thus selected would trigger an additional application action.

72. In the case of the DrawingArea widget described above, one would typically override that widget's translation table (the table that defines how a widget will respond to particular events) in order to specify existing, or add additional callback functions. These callback functions would control certain behaviors in response to various types of user interactions. One could have tracked pointer

motion and/or clicks in the DrawingArea widget and take appropriate action accordingly.

73. An exemplary prior art reference that described the facility for embedded Motif widgets to process user interaction is [Heller91].

**d. MediaMosaic, X Window Embedding**

74. In [Lin92] a technique is discussed for embedding multiple X client drawing surfaces (windows) into an X Virtual Screen. The Virtual Screen was implemented by designing a "pseudo-server" which intercepts and modifies X protocols between X clients and the X server. The modified protocol causes a re-parenting of a client's window to a specified window, instead of the root window of the real screen. The result is that all screen modification is done by the original client applications but the output appears in the Virtual Screen. This capability would be helpful for an X web browser application, where output from a program like XV (to be discussed below) would be embedded in the viewing window.

75. Motif, Motif widgets, and Motif programming techniques and capabilities were tools, known and routinely used by persons of skill in the art in the ordinary course of development during the early 1990s, used by the prior art systems I describe below. For example, I describe below that Mosaic supported Motif widgets. Also, Viola and MediaView used programming constructs very similar and analogous to the "widget" concept.

## **5. Programming Techniques**

**a. Source Code**

76. Human-readable programming instructions, or source code for computer languages, can be manipulated in several different ways. Depending on the language, code can be edited to form a *script*, which may not be further processed, or it can be partially compiled to form *bytecode*, or fully compiled to the binary code

accepted by a particular type of computer. These concepts are discussed in more detail below.

**i. Scripts**

77. Source code for a scripting language is never compiled but instead is interpreted on the fly to cause the target computer to execute the equivalent instructions in its binary code. Scripting languages such as JavaScript or TCL, allow for rapid experimentation because the code is easily changed and there is no need for a time-consuming compilation step.

**ii. Bytecode**

78. Bytecode is programming code that, once compiled, is run through a virtual machine before reaching the computer's processor. By using this approach, source code can be run on any platform once it has been compiled and run through the virtual machine.

79. For example, bytecode is the compiled format for Java programs. Once a Java program has been converted to bytecode, it can be transferred across a network and executed by a Java Virtual Machine (JVM).

**iii. Compiled Languages**

80. Compilation is the process that transforms a program written in a high-level programming language, such as C or C++, called source code into object code, or binary code native to the target computer. Source code must go through several steps before it becomes an executable program. The first step is to pass the source code through a compiler, which translates the high-level language instructions into object code.

81. The final step in producing an executable program – after the compiler has produced object code – is to pass the object code through a *linker*. The linker combines modules and gives real values to all symbolic addresses, thereby producing machine code.



**b. Structuring Documents****i. Background**

82. In the 1993 time frame there were many projects on multimedia document structuring, including but not limited to markup languages. Following are descriptions of some of those efforts.

83. There were a number of existing and emerging standards for structuring hypermedia applications. Examples include SGML, HyTime, MHEG, ODA, PREMO and Acrobat.

84. SGML (Standard Generalized Markup Language) was a markup language for delimiting the logical and semantic content of text documents. Because of its flexibility, it became an important tool in hypermedia systems.

85. Acrobat PDF was a format for representing multimedia (printable) documents in a portable, revisable form. It was based on PostScript and as early as 1993 it incorporated tags for embedding images in a PDF document. [Adobe93]

86. An exemplary prior art reference describing these standards for structuring hypermedia applications was [Adie93].

87. The Rich Text Format (RTF) standard was a method of encoding formatted text and graphics for easy transfer between MS-DOS, Windows, OS/2, and Apple Macintosh applications.

88. The RTF standard provided a format for text and graphics interchange that could be used with different output devices, operating environments, and operating systems. RTF used the ANSI, PC-8, Macintosh, or IBM PC character set to control the representation and formatting of a document, both on the screen and in print. With the RTF standard, one could transfer documents created under different operating systems and with different software applications among those operating systems and applications.

89. An RTF file consisted of unformatted text, control words (tags), control symbols, and groups.

90. A tag was a specially formatted command that RTF used to mark printer control codes and information that applications use to manage documents.

91. RTF was generally described in [Microsoft92b].

92. These mechanisms for structuring documents were tools, known and routinely used in the ordinary course of development by persons of skill in the art during the early 1990s, used by the prior art systems I describe below. For example, Mosaic and Viola both used HTML and drew on HTML+, and RTF was a tool used by both MediaView and OLE.

## ii. HTML

93. HTML stands for HyperText Markup Language. It was the authoring language used to create documents on the World Wide Web. HTML was used to define the structure and layout of a Web page, how a page looks and any special functions. As the viewer of a web page you don't generally see the HTML. It is hidden from your view. However, you do see the results.

94. HTML was invented by Tim Berners-Lee. While he was the primary author, he was assisted in its development by colleagues at CERN. In 1991-1992 Berners-Lee also wrote one of the first graphical HTML browser-editors, as seen in Figure 4. Berners-Lee developed this browser-editor on a NeXT workstation by providing a Hypertext subclass of the Text class provided by the NeXT development system.

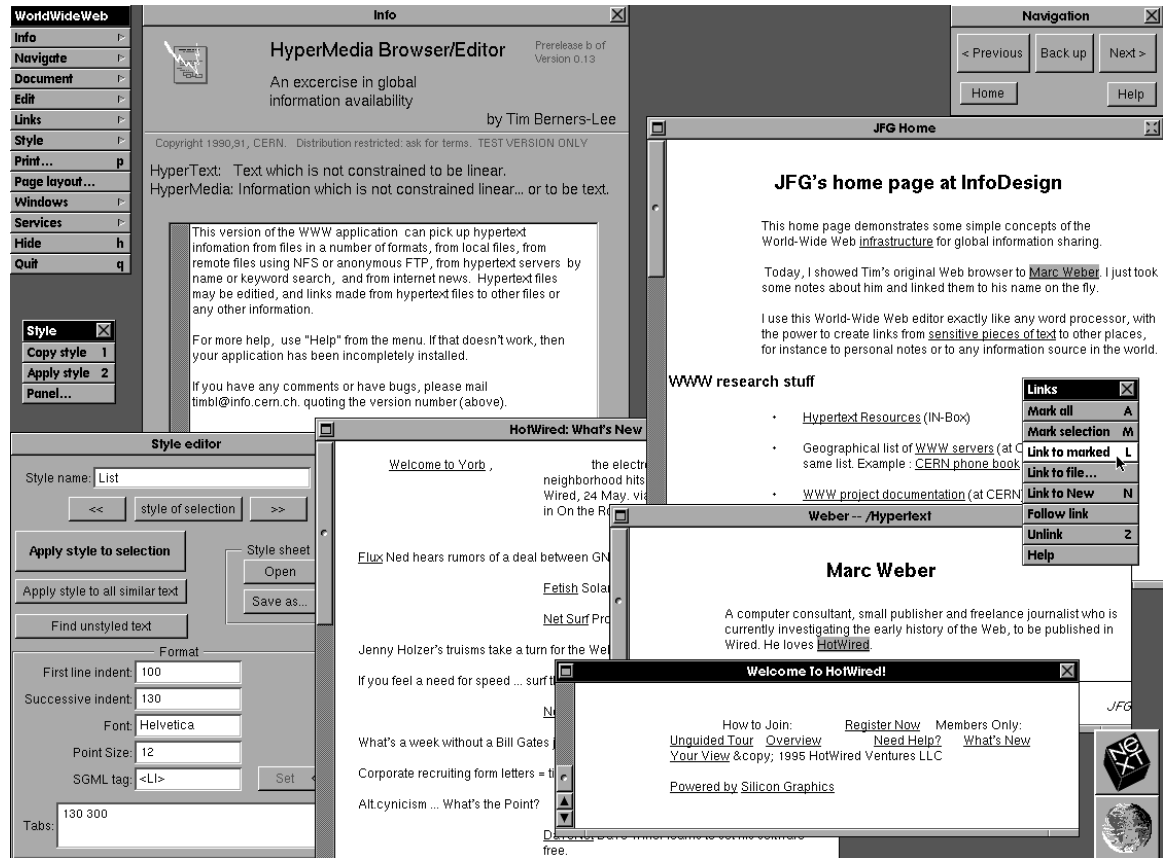


Figure 4. CERN Browser/Editor

95. HTML, being a markup language, used special characters for escaping non-contextual commands, known as *tags*. For example, the name of an HTML document was specified by:

```
<title>My first HTML document</title>
```

96. In general, the brackets < and > delimit tags in an HTML document. So, adding a heading to a document is specified by:

```
<h1>An important heading</h1>
```

97. Other content could be added by similar commands.

### iii. HTML+

98. Early versions of HTML were limited in the degree to which they could represent multimedia content, such as graphics, video and other arbitrary content. In order to rectify this situation several enhancements to HTML were proposed. One of

those was the HTML+ proposal by Raggett in [Raggett93a]. As he states, "HTML+ represents a substantial improvement over the existing format: HTML, offering nested lists, figures, embedded data in foreign formats for equations etc., tables with support for titles and column headings, change bars, entry forms for querying and updating information sources and for use as questionnaires for mailing. This document specifies the HTML+ format with guidelines on how it should be rendered by browsers."

99. He goes on to describe a simple example of HTML+ formatting with the following statements:

```
<title>A Simple HTML+ Document</title>
<h1 id="a1">This is a level one header</h1>
<p> This is some normal text which will wrap at the window margin.
You can emphasize <em>parts of the text</em> if you wish. </p>
```

100. The text of the document includes tags which are enclosed in <angle brackets>. Many tags have matching end tags for which the tag name is preceded by the "/" character. The tags are used to markup the document's logical elements, for example, the title, headers and paragraphs. Tags may also be accompanied by parameters, e.g. the "id" attribute in the header tag, which is used to define potential destinations for hypertext jumps.

#### iv. Tag Attributes

101. In addition to having simple tags such as <p>, which means a paragraph break, tags could also have associated attributes that extend their meaning. HTML+, for example, provided several ways of specifying emphasized text. They were:

```
<em b> bold text
<em i> italic text
<em U> underlined text
<em sup> superscript text
<em sub> subscript text
<em tt> type writer font (courier)
```

```
<em hv> sans serif font (helvetica)
<em tr> serif font (times roman)
```

102. HTML+ also introduced an EMBED tag with an associated *type* attribute. For example the use of `<embed type="application/eqn"> 2 pi int sin (omega t)dt </embed>` produced a proper graphical rendering of the specified equation.

103. The extensibility of the HTML+ EMBED tag with its *type* attribute was discussed further by Bill Janssen and Dave Ragget. In [Raggett93b], Raggett disclosed the use of MIME to identify the format of embedded data, and managed that data from separate programs using, e.g., UNIX pipes. Raggett proposed the use of X resources to bind the MIME type to the separate program.

104. In [Janssen93], Janssen disclosed that this could be achieved by creating an X sub-window and passing a window ID of that sub-window to the program managing the data embedded through use of the EMBED tag with *type* attribute.

#### (a) HyperTies HML Example

105. Hyperties allowed users to traverse textual, graphic or video information resources in an easy way. The system adopted the 'embedded menu' approach, in which links were represented by words or parts of images that appeared in the document itself. Users merely selected highlighted words or objects that interested them and a brief definition appeared at the bottom of the screen. Users could continue reading or ask for the full article (a node in the hypertext network) about the selected topic.

106. Multimedia supported by Hyperties included:

- Full screen or windowed video sequences using a video overlay board
- Video sequences played on a second monitor connected directly to the laser disk player
- CD audio

- Digital audio from SoundBlaster (VOC), MIDI (MID) or Windows (WAV) files

107. A developer of a Hyperties document used scripts to make links and to control behavior of multimedia components. A typical script for controlling a video window sequence was:

```

VIDEOWINDOW
    SIZE = x BY Y AT x,y
    SOURCE = x BY y AT x,y
    TRANS PARENTCOLOR = n
    [IMAGESIZE = n]
    [USERBUTTON = "button -name"]
    [INFOBOX = "button-name"]
ENDVIDEOWINDOW

```

108. Other multimedia specifications might be:

```

TYPE = VIDEODISC
STARTFRAME = 27824
ENDFRAME = 28440
FRAMEDISPLAY = OFF

```

109. where the TYPE field specifies the kind of multimedia content to be used.

110. HyperTies had versions available in both a UNIX workstation environment and an IBM PC environment.

111. Exemplary prior art references describing HyperTies include [Shneiderman92] and [Shneiderman91].

#### (b) Cohen GML Example

112. Cohen [Cohen94] describes an electronic book reading system that allows for linking locations in the book's text to a multimedia entity located elsewhere in the document. Thus, at the appropriate location a sound can be played or a video displayed that corresponds to the material being described in the text.

113. Multimedia entities in the book are described by a link description tag (LDESC) located in the prologue of the document. An example of an LDESC tag is:

**SUPPLEMENTAL EXPERT REPORT  
HIGHLY-CONFIDENTIAL – ATTORNEYS' EYES ONLY**

**RLP**

```

:LDESC ID=eleph_movie    OBJTYPE=video
                        OBJECT='family_clp.vid'
                        STORE=external
                        DATA='video.exe \ CD Video File Format A'
                        RUNTEXT=' African Elephant Family Video'
                        AUTOLAUNCH=no
                        VIEW=center
                        STYLE=2

                        ALT_OBJECT='family_clp.anm'
                        ALT-STORE=Internal
                        ALT_DATA='\Animation; 100 frames; B/W bitmap'
                        ALT_TEXT=' African Elephant Family Animation'
                        ALT_TEXT2='Video not available'

```

114. Multimedia objects are accessed from the current page by a Link Identification Tag (LID) that references the corresponding LDESC tag. An example of a page with an LID is:

**:P.Studies have been conducted over the last decade on the changing populations of wildlife around the world. Inroads made by civilization have drastically reduced the habitats for indigenous species. A typical population undergoing such changes is that of the elephant.**

**:L LID=eleph\_movie.Motion Picture of African Elephant family:eL**

115. **:P** denotes a paragraph of text on the page and **:L** denotes the presence of a link activation tag. When clicked by the user the elephant family video clip is played to completion in a window.

116. It's also important to note in the above example that the OBJTYPE attribute specifies the media type to be used (e.g., video), the DATA attribute specifies the executable program handler associated with the data, and the AUTOLAUNCH parameter specifies whether the object should be invoked

automatically or launched by user input. Thus, playing the above video sequence would have to be invoked by the user, but setting AUTOLAUNCH to "yes" would result in automatic invocation.

### c. Parsing

117. A Hypermedia document browser must be capable of parsing the document it opens in order to identify content and its layout. A browser that parses documents like HTML and its variants has to identify tags and other directives in order to extract a complete "sentence" and render it accordingly. Other document browsers parse other streams of data, also to identify and properly render a "sentence."

118. MediaView, a hypermedia document browser to be discussed later in more detail, parsed a stream of data in Rich Text Format (RTF) in order to discover the location of multimedia components embedded therein. It did this by looping through a series of "runs" of RTF data and searching for a "character" with an embedded flag that identifies it as a pointer to a multimedia component. From [Phillips91b], the Objective-C code that performed the parsing operation was:

```
- takeInventory
{
    int r , numChars;
    int numRuns = theRuns->chunk.used/sizeof(NXRun);

    if( [inventory count] ) [inventory empty];
    numChars = 0;
    for( r=0; r<numRuns; r++ ) {
        numChars += theRuns->runs[r].chars;
        if( theRuns->runs[r].rFlags.graphic ) {
            [inventory insertKey:(const void *)theRuns->runs[r].info
             value:(const void *)numChars - 1];
        }
    }
    return self;
}
```



119. In the statement, if( theRuns->runs[r].rFlags.graphic ) tests TRUE for a non-zero value of the graphic flag, the value of the multimedia component pointer info was stored in an inventory buffer. That data was subsequently used to activate and control the identified component.

120. While many types of web documents were parsed by a browser during document rendering, other types of documents were compiled into a binary file format and processed according to that format during rendering. Which technique to use was simply a design choice and both approaches were common and easily interchangeable. For example, the ViolaWWW browser could parse either a text-based HMML document, or process a binary HMMLB document. (See, e.g. [Viola-5/27/93] at \violaTOGO\docs) (showing exemplary documents of both types.)

121. It should also be noted that the types of parsers being discussed were readily available to persons of ordinary skill in the art. For example, one public correspondence of the era indicates that Tim Berners-Lee happily shared an SGML parser with whoever asked. [www-talk-00293029]. Also, the developer of the ViolaWWW browser noted that others could simply install an SGML parser in order to use his browser. [www-talk-00293088].

## **6. Networked Environments**

### **a. Client-Server Computing**

122. The client-server model of computing was a distributed application structure that partitioned tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicated over a computer network on separate hardware, but both client and server could reside in the same system. A server machine was a host that was running one or more server programs which share their resources with clients. A client requested a server's content or service function. Clients therefore initiated communication sessions with servers which awaited incoming requests.

123. One of many exemplary prior art references describing client-server computing is [Stevens90].

**b. Distributed Computing**

124. Distributed computing was a field of computer science that dealt with distributed systems. A distributed system consisted of multiple autonomous computers that communicated through a computer network. The computers interacted with each other in order to achieve a common goal. A computer program that ran in a distributed system was commonly called a distributed program, and distributed programming was commonly referred to as the process of writing such programs.

125. One example of a prior art system that employed distributed computing was NCSA Collage, which was used for scientific data analysis, visualization, and collaboration. [Collage92]. Other prior art systems using distributed computing include one of my projects called Scientific Visualization Workbench [Phillips 88], Mathematica [Wolfram88] and RenderMan[NeXT93a ](of which the latter two were used in conjunction with the MediaView system described below, [Phillips91c]). Applications were frequently configured to interoperate with applications implementing distributed computing, such as Mosaic (which interoperated with Collage, [Andreessen93b]) and Viola (whose author disclosed a distributed chess application, [Wei94]). Thus, distributed computing applications were tools, well-known by the early 1990s and routinely used in the ordinary course of development by persons of ordinary skill in the art, that other applications interoperated with.

126. Distributed computing generally referred to the use of distributed systems to solve computational problems. In distributed computing, a problem was divided into many tasks, each of which is solved by one or more computers.

127. The word *distributed* in terms such as "distributed system" and "distributed programming" generally referred to computer networks where individual computers were physically distributed over some geographical area.

128. Distributed computing resources were used to solve certain problems by performing what was known as parallel processing. Such problems could involve so much data or were so complex that obtaining a timely solution required dividing up computation between multiple distributed computers running at the same time. In this technique different parts of the problem were assigned to different machines and the final solution was obtained by "harvesting" sub-solutions from all computers involved. Such a technique was used by the named inventors of the patents-in-suit for their VIS rendering system, [Ang94] and [Doyle93].

## **IV. Hypermedia Browsers That Existed By 1993 – 1994**

### **1. Network-Enabled Compound Document Platforms**

#### **a. OLE**

129. OLE, discussed above, defined and implemented a mechanism that allowed applications to "connect" to each other as what OLE documentation referred to as software "objects" – collections of data and accompanying functions to manipulate the data. This connection mechanism and protocol was called the Component Object Model, a concept that placed special emphasis on its ability to provide a robust system model for seamless connections between software components running across multiple computers on a network.

130. Also, [Koppolu98], a patent involving OLE, discloses compound documents in which interactive objects of various types were embedded within documents. A contained object (containe) was displayed within a window environment of a container application. The containee object was managed by a server application.

131. OLE prior art disclosed the use of OLE technology in distributed networked environments, including the ability for users to retrieve compound documents across a network, for component objects to communicate across a network, and for component objects to be transmitted across a network. [Microsoft93a]; [Eckersion93].

132. Also, I prepared a video that demonstrates prior art OLE in use. (See Appendix C, at OLE.wmv.)

## **2. Hypermedia Browsers**

### **a. MediaView**

133. MediaView is a multimedia document browser whose characteristics are described in three major publications. [Phillips91c] provides a general user's overview of MediaView's capabilities and presents examples of several of its applications. [Phillips91a] focuses on MediaView's graphical and visualization features, while [Phillips91b] describes its design and internal structure. Figure 5 shows the MediaView browser window and its main interface features. For the most part, the other publications generally corroborate my public demonstration and use of MediaView and show screenshots for its use, as well as identify others involved in those demonstrations and use. Substantively, there is not a significant difference between the particular versions of MediaView for purposes of my analysis in this report, though obviously later versions of MediaView, as explained below, generally improved on earlier versions. All versions generally worked the same from an architectural perspective.

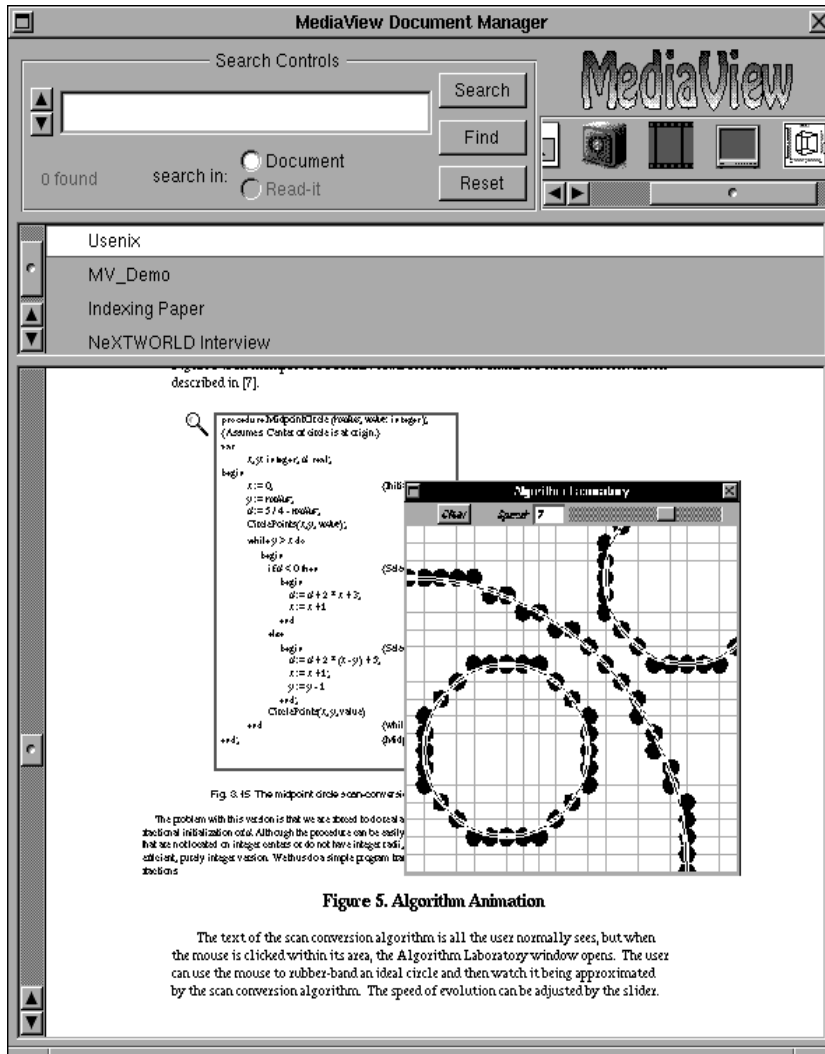


Figure 5. MediaView Browser

134. The large scrolling window (the content view) displays a portion of the document that has been selected in the scrolling summary view above, which in this case is the document titled *Usenix*. Directly above the summary view is a search panel with its controls. Then to the immediate right is an area called the icon well. Each icon represents a multimedia component such as sound, video, textual and graphical annotations and dataset browsing. The documents listed in the summary

view had been previously created by different authors and are being viewed in reading, or browsing mode. MediaView, however, has an editing mode as well where icons from the well can be dragged and dropped into a document at the current insertion point. Then, depending upon the icon, an image can appear or a sound recording widget pops up to allow insertion of audio into the document. The document can then be saved, complete with the content just added.

135. The icon well also facilitates MediaView's multimedia searching capability. After an icon is dragged into the search window, pressing the *Search* button causes all instances of the corresponding media type to be located and activated. Thus, sounds can be played or images can appear at the location where they appear in the document.

136. MediaView was designed and developed using the object-oriented NeXTSTEP development environment provided with the NeXT computer. The style of programming in NeXTSTEP is to subclass objects provided in the Application Kit and then to override default behavior or add new behavior. The heart of the application is a custom multimedia object, which is based on the NeXTSTEP Text object, [NeXT89]. Support was added to accommodate arbitrary NeXTSTEP views. This permitted the incorporation of custom multimedia control and display objects into a document. Several control panels were constructed with the NeXT Interface Builder and Objective-C code was written to produce the various user interface objects and to build the multimedia documents.

137. As stated above, the content view of a MediaView document is represented by a single large instance of a subclass (MediaViewText) of the Text class. The multimedia components of the document are each a subclass of the View class which has been inserted into the top-level MediaViewText object as a sort of special character. It is up to the View subclass to implement whatever behavior is appropriate for each component.

138. The internal representation of the text in a Text object is based on a subset of the Rich Text Format (RTF), described above. RTF is a markup language, i.e. a language where formatting characters are "escaped" to distinguish them from ordinary text. As an example, the following RTF code

```
{\rtf1\ansi{\fonttbl\f0\fswiss Helvetica;}\f0\pard
This is some {\b bold} text.\par
}
```

139. when read by a program that supports RTF, would be rendered as:

This is some **bold** text.

140. A Text object or any subclass allows a "graphic character" to be embedded in the text stream. As stated in NeXT's Text Class description:

"Each graphic is treated as a single character: The text's line height and character placement are adjusted to accommodate the graphic "character." Graphics are embedded in the text in either of two ways: programmatically or directly through user actions. In the programmatic approach, you add an object—generally a subclass of Cell—to the text. This object will manage the graphic image by drawing it when appropriate."

141. In MediaView "graphic characters" are subclasses of a ViewCell, whose address is the data contained in the info field described above in connection with "Parsing."

142. Original MediaView documents are created by the "drag and drop" paradigm. For MediaView this means dragging an icon that represents a text file from the NeXT file viewer and releasing it over the summary view. This action converts the file to MediaView internal format, adds its title to the summary view, and displays its text in the content view. Documents that can be processed by MediaView in this way are those produced by WriteNow, FrameMaker, and any file that is in the Rich Text Format (RTF). Several applications can produce RTF documents, including Microsoft Word and WordPerfect. Finally, even an ASCII text file can be processed in this way; a default font and format are assumed which the user can override with Media View formatting tools.

143. Once the textual structure has been produced, a user can begin to add multimedia components interactively. Annotations – textual, graphical, or aural – can be added using drag and drop icons, and TIFF or Encapsulated PostScript images can be pasted into the document at the current cursor location. Once finished, the document can be saved, complete with all multimedia components.

144. In addition to accommodating multimedia data that is represented by the standard items found in the icon well, MediaView also incorporates several functions that are in the data display and data manipulation category. One of these is the "live equation" feature. This feature takes advantage of the early bundling of *Mathematica* [Wolfram88] with all NeXT workstations. MediaView ran *Mathematica* either as a stand-alone application and used its user interface, or it accessed it as a server application using UNIX pipes. The live equation feature used the latter method. Figure 6 shows an example of the live equation viewer in action.



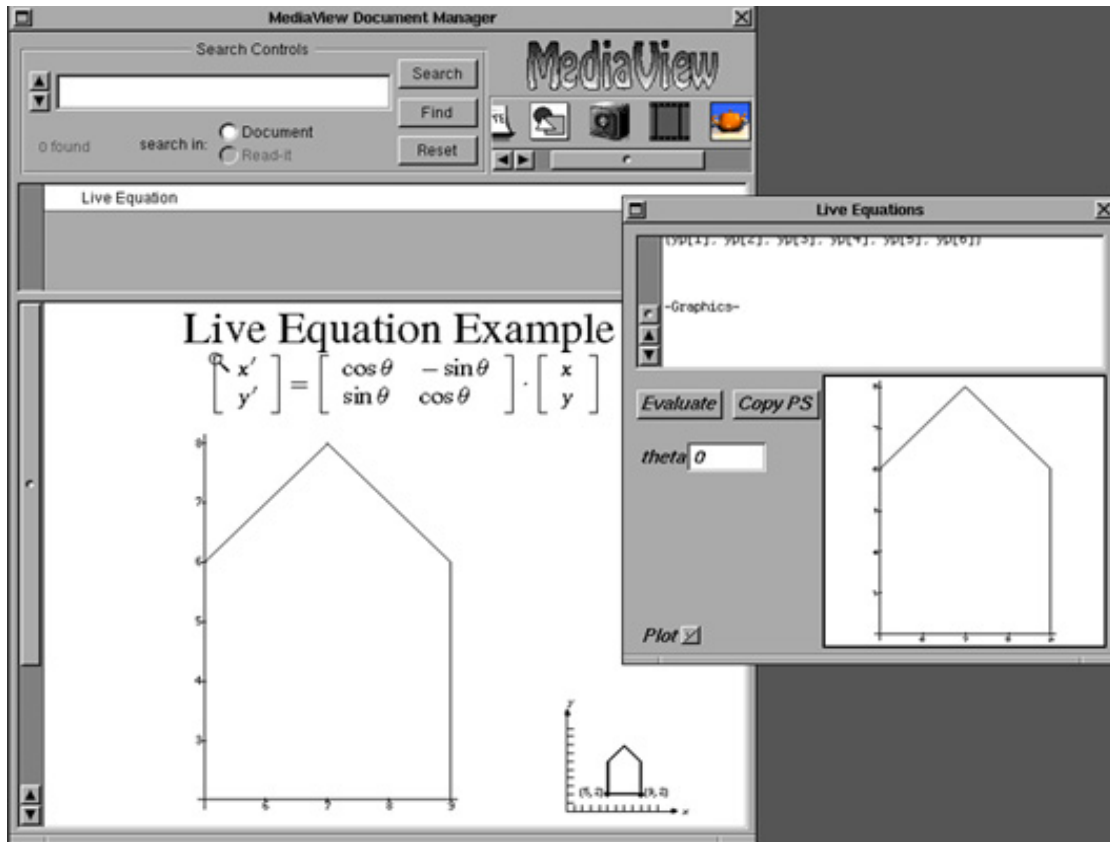


Figure 6. Live Equation Viewer

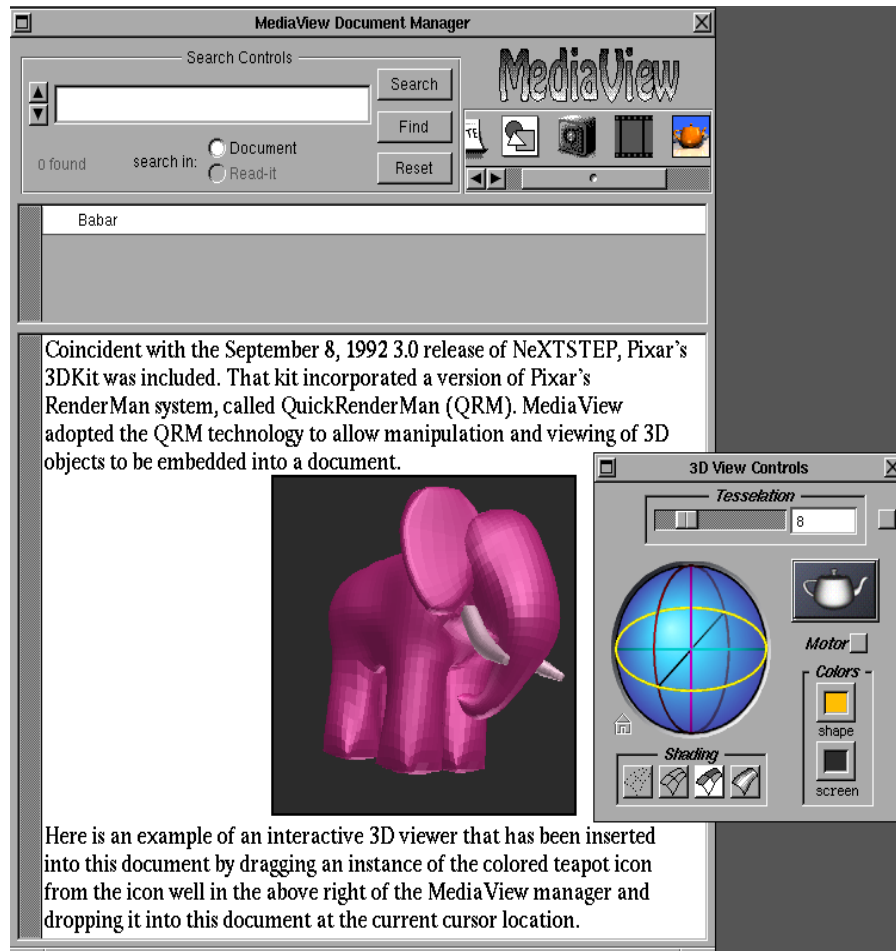
145. The magnifying glass icon on the upper left corner of the matrix equation signifies a live equation. Clicking it brings up the inspector window shown on the right. The graphical entity that can be manipulated by the equation is the small house shown below-right. The user can enter a value of  $\theta$  in the text field and press *Evaluate* to send the house data and the equation to the Mathematica server via UNIX pipes.

146. Mathematica performs the requested operation and records its output in the scrolling results window. Graphical output appears below the textual output. One can reevaluate the equation any number of times, and when the desired result is achieved, pressing the *Copy PS* button will transfer the graphical output to the clipboard. In this example  $\theta = 0$  was used and the output was pasted into the MediaView document to the left of the original graphic.

147. UNIX pipes were used in this example. Remote procedure calls (RPC) could just as easily have been used to access a Mathematica server anywhere on the network. For example, Mathematica could be and was used as part of a distributed system as construed by the Court. See, e.g., McRae Tr. 7:14-8:25.

148. The foregoing MediaView document creation process is thoroughly described in the video demonstrations I prepared and in [LANL93].

149. Another data display and manipulation feature is afforded by the presence of a version of Pixar's RenderMan image viewing technology that became available in NeXTSTEP in 1992. MediaView uses this capability to view and manipulate 3D solid object datasets that are specified by the RenderMan Interface Bytestream (RIB) protocol. Examples of this functionality are shown in the July 1994 cover page and on page 35 of R&D Magazine from July 1994 (showing an aviator and a 3D manipulation panel), and below in Figure 7.

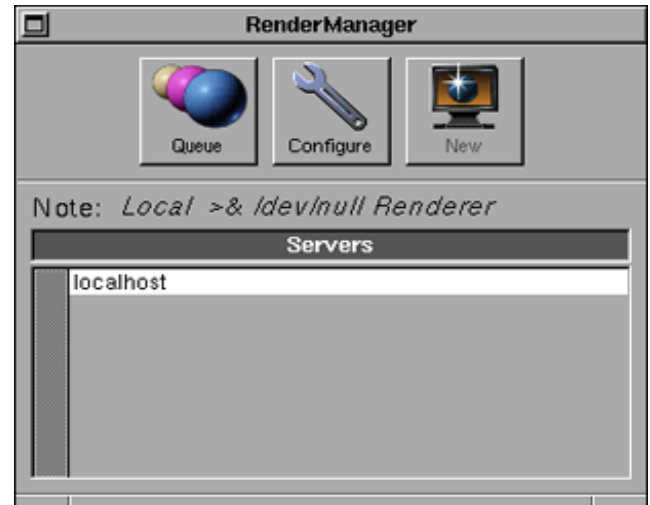


**Figure 7. RenderMan 3D Viewer**

150. The RenderMan viewer has been inserted into this MediaView document by dropping an instance of the colored teapot icon dragged from the icon well in the above right of the MediaView browser. The elephant image can be reoriented simply by clicking in its display view and dragging the mouse until the desired orientation is attained. If a richer range of control is desired the panel to the right can be summoned by double-clicking the image. That panel permits color changes and control over the fineness of image representation, i.e. tessellation. Also, different shading strategies can be applied by clicking the appropriate button. Lastly, while in this example the document has no text, it could, and the 3D pink elephant

(like any other custom component) could be sandwiched between text and other markup in the order in which it will be displayed.

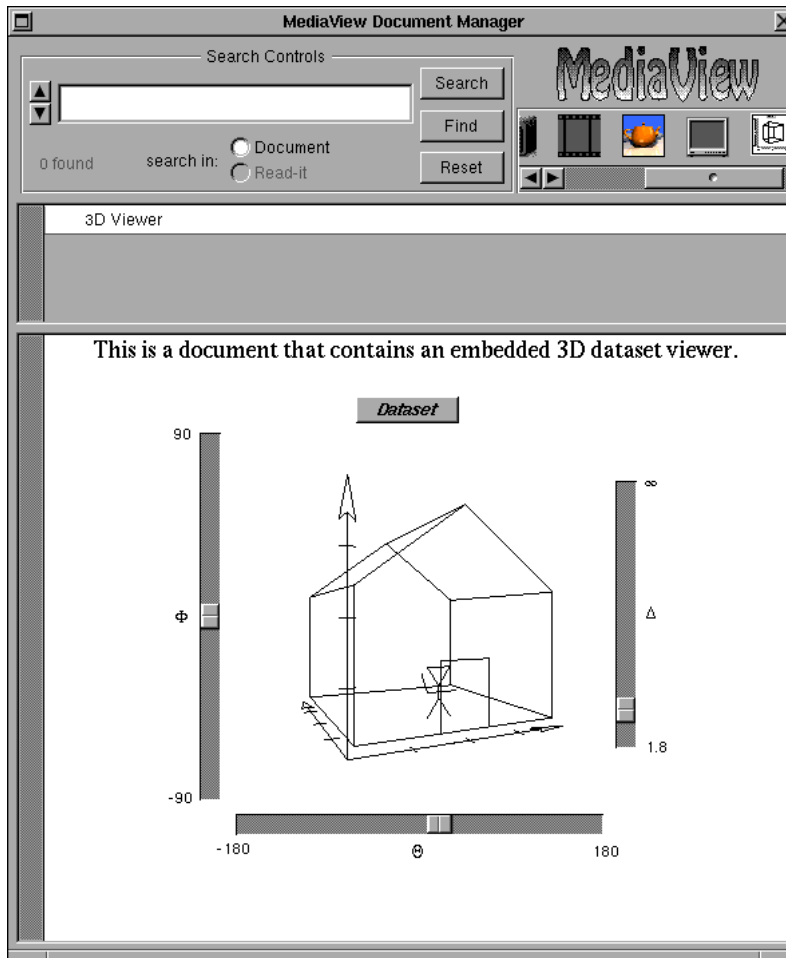
151. Because rendering complex images can be compute-intensive, the NeXT RenderMan system allows for the specification of multiple rendering engines, i.e., parallel processing. Each engine will work on part of the rendering task and submit its results to the control view. Engines need not be on the same machine or, indeed, on the same local network. A tool called the RenderManager can distribute the rendering tasks to machines anywhere on the Internet. This



**Figure 8. RenderManager Control Panel**

control panel is seen in Figure 8 and includes buttons for rendering queue examination, configuring current rendering tasks and adding new ones.

152. Another tool available in MediaView is a 3D point and line dataset viewer and controller. In this tool the viewing environment and its controls are completely embedded in a MediaView document; no additional panel is needed. An example of this tool is seen in Figure 9.



**Figure 9. 3D Line Dataset Viewer**

153. When this document was opened and parsed by the MediaView browser, the presence of this hypermedia component was discovered and automatically activated. A pointer associated with the component referenced a separate dataset, which was then read in and displayed. Moving the sliders reorients the object in real time.

154. All MediaView multimedia objects, including the live equation, RenderMan and 3D line viewer are automatically in a running and active state as soon as a MediaView document has been parsed. This occurs because all objects in MediaView, as in most NeXT applications, implement three archiving methods:

write, read, and awake. At the time the MediaView document is first created the characteristics of all objects contained therein are written to the document file. Upon being transmitted to and opened by a MediaView browser all constituent objects and their data are read into virtual memory and awakened to the state they were in when first created by the author. Thus, there is never a step of explicitly activating any multimedia component in MediaView.

**b. HyperCard**

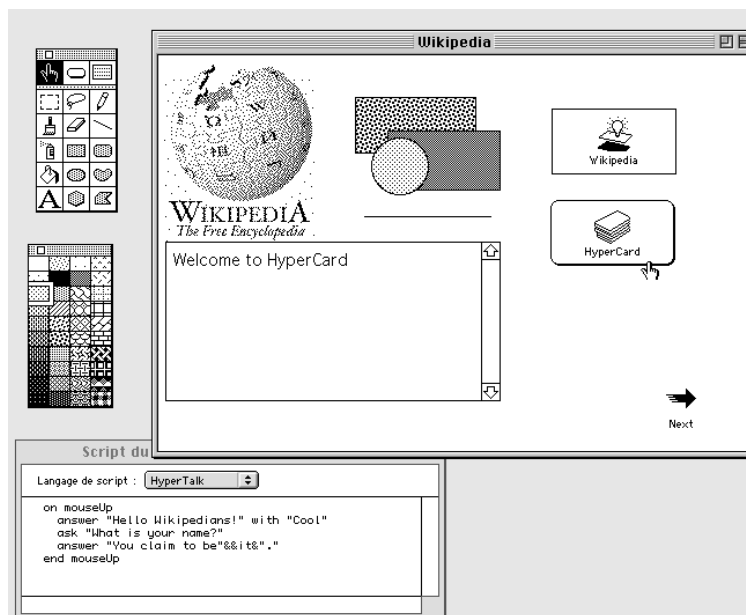
**i. HyperCard Overview and its Operating Environment**

155. HyperCard was an application program (*see* [HyperCard]) and a simple programming environment released by Apple Computer in 1987 for the Apple Macintosh computer. It most closely resembled a database application in concept, in that it stores information, but unlike traditional database systems HyperCard was very flexible and trivially easy to modify. In addition HyperCard included a powerful and easy to use programming language to manipulate that data. The HyperCard application program and its features, including those that I discuss below, are described in [Goodman90]. (See also [HyperCardBasics90]).

156. HyperCard (the generation beginning with version 2.0) required a Macintosh running a minimum of version 6.0.5 of System and associated Finder files. A HyperCard screen shot is shown in Figure 10.<sup>9</sup>

---

<sup>9</sup> See also, e.g., [Goodman90] at Figure 4.



**Figure 10. HyperCard Stack in Development**

157. As part of my preparation for writing this expert report, I inspected and used a Macintosh IIsx computer running HyperCard version 2.1, as shown in the videos I am submitting along with this report.

#### ii. HyperCard's Architecture for Hypermedia Browsing

158. HyperCard was based on the concept of "stack" of virtual "cards". Each card included fields that stored data, and the pattern for each card was known as the "background". Backgrounds could include pictures, picture fields, buttons, text, text fields (editors) and other common GUI elements, which would then be copied onto new cards.

159. HyperCard provided a hypertext environment, in which a user could click on links to browse to other cards or stacks. The hyperlinks could be provided mechanisms such as buttons or fields. The functionality of links could be augmented using HyperTalk scripting, described below.

160. HyperCard could be operated in networked environments too, in which stacks were located on networked computers (such as servers) and users browsed among cards and stacks located throughout those networked computers. (See [Sadowski11] at 58.) I have inspected HyperCard stacks running in networked environments as part of the materials I inspected.

**iii. HyperCard's Architecture for Hypermedia Authoring**

161. Authors could create webs of interlinked HyperCard cards and stacks using HyperCard's authoring environment. The authoring process generally involved combining cards with backgrounds, fields, buttons, and other objects into meaningful stacks of information, and linking those cards and stacks in meaningful ways.

**iv. HyperTalk Scripting Language**

162. Scripting in the HyperTalk language allowed the system to be easily modified and extended. Unlike most programming languages, HyperTalk really was easy to use. Allowable syntax included all sorts of versions of the same statement, all in readable English, to avoid forcing the user to write their programs in a particular format.

163. Adding scripts was also easy. The user simply "option-clicked" on any element in the stack, and an editor would pop-up. The script could then be edited, saved, and used immediately. HyperTalk was used as a programming tool that empowered ordinary computer users. HyperCard 2.0 added a fairly powerful and easy-to-use debugger as well.

164. Scripts could be associated with buttons, cards, or stacks. They could specify actions that HyperCard or other application programs would take in response to various events. By way of example, HyperTalk scripts could include the *on OpenStack* and *on OpenCard* event handlers. These specified actions to be taken



automatically upon opening a stack or a card. By way of contrast, the *on mouseUp* event handler specified actions to be taken after a user clicked on the mouse.

165. In 1988, MacroMind released a product called HyperCard Driver. This product allowed the playback of movies created in VideoWorks within a HyperCard stack, such as that available for Apple computers at the time. The term "driver" refers to a software bridge (for example an application programming interface or API) that permitted two programs to work together seamlessly. [Sadowski11]

**v. XCMDs, XFCNs and Mac-specific Issues**

**1) Data Fork vs. Resource Fork**

166. The *resource fork* was a construct of the early Mac OS operating system used to store structured data in a file, alongside unstructured data stored within the *data fork*. A resource fork stored information in a specific form, such as icons, the shapes of windows, definitions of menus and their contents, and application code (machine code). For example, a word processing file might store its text in the data fork, while storing any embedded images in the same file's resource fork. The resource fork was used mostly by executables, but every file was able to have a resource fork.

**2) XCMDs and XFCNs**

167. The power of HyperCard could be increased significantly through the use of different types of external command and external function modules, more commonly known as XCMDs and XFCNs. Together they were called *externals*. These were compiled code libraries packaged in a resource fork that integrated into either the system generally or the HyperTalk language specifically; this was an early example of the plug-in concept. Externals could be included in a stack, where they were directly available to scripts in that stack. Many vendors offered specialized XCMDs and XFCNs. [Sadowski11] at 12, 13, 17, and 46. At one point, Oracle offered an XCMD that allowed HyperCard to directly query Oracle databases on any

platform. (See [Goodman90] at chapter 49.) XCMDs and XFCNS allow access to the Macintosh Toolbox, which contained many lower level commands and functions not native to HyperTalk, such as control of the serial port. (See [Goodman90] at chapter 49.) During its peak popularity in the late 1980s, many vendors offered thousands of externals for everything from HyperTalk compilers, to graphing systems, database access and internet connectivity among many others.

168. Programmers who developed externals often used products such as Think C or Think Pascal, systems that were Mac programming environments for the C and Pascal languages. (See [Goodman90] at Chapter 51.)

### 3) ResCopy / ResEdit

169. *ResEdit* was a tool that Apple made available to programmers for editing the contents of the resource fork of a file. *ResCopy* was an application that was written to support HyperCard stack development. It allowed a developer to copy selected resources from one stack to another, saving the effort of recreating them. Such tools enabled stacks to retrieve and interoperate with XCMDs located apart from a stack by moving those XCMDs into the stack's resource fork. (See [Goodman90] at Chapter 51.)

### vi. VideoWorks and Director

170. Director was a multimedia development application. Director was originally called VideoWorks, released in 1985, and written to run on an Apple Macintosh computer. Director was used to create multimedia documents that contain animations, sounds, images, text and custom user interactivity. Commonly these files were referred to as Director "movies", and Director users were often referred to as "developers". Director, or in some cases a separate companion program also provided the ability to create a standalone application that would play, though not edit, a set of Director movies. These applications were often referred to as Projectors. Macromedia also provided a multimedia library runtime module. The idea behind

this was that the library could be used by other host applications to incorporate playback of Director files. Depending on the host application, this module could be accessed with a plugin, used as a dynamic loaded library, or compiled in. (See [Sadowski11] generally.)

171. The basic functionality provided by Director included:

- Display multimedia content (images, sound, text, animation)
- Create animation sequences of images with control of location, size, registration, colorization, etc.
- Scripting language to create interactive documents
  - respond to mouse clicks, mouse movements, key presses
  - provide support for user interface items such as menus
  - programmatic control of animation elements
  - control over sequencing of animations
  - extensible language with XObjects
    - allow control of other devices
    - provide a way to access additional operating system features
    - Define a simple sequence of files that display in order
- Ability to create independent executables that play one or more Director files
- Media editing tools (bitmap paint module, basic text editing, simple shapes)
- Import of media elements such as pictures, text and sounds created by other applications
- Store a library of content (images, text, sounds) within a single Director file
- Ability to create documents that are then used in other multimedia host environments.
  - HyperCard
  - Authorware
  - HookUp!

- MazeWars+
- MediaMaker (see [Sadowski11] at 5-23, 29).

172. When HyperCard was introduced by Apple, MacroMind was inspired to provide a way to play animations within that environment. An XCMD was written to access a version of the VideoWorks playback library and then display an animated sequence. The XCMD provided ways for a HyperCard application to use scripting options to control the location and playback of an animation file. The animation file could be located anywhere within the accessible file system of the computer, such as an external disk, floppy drive or on a mounted server. (See [Sadowski11] at 46, 58.)

173. In 1990 a scripting language called Lingo was added to Director 2. From the very first release, Lingo provided the ability for the language to be extended by the use of independent code modules called XObjects. Director also provided an XObject that supported the XCMDs that had originally been written for use in HyperCard. Macromedia provided a developer's tool kit so that programmers could create their own XObjects using the C or Pascal language. (See [Sadowski11] at 47-50.)

#### vii. QuickTime

174. QuickTime was an extensible proprietary multimedia framework developed by Apple Inc., capable of handling various formats of digital video, picture, sound, panoramic images, and interactivity. Apple released the first version of QuickTime on December 2, 1991 as a multimedia add-on for System Software 6 and later. QuickTime was a stand-alone application in its own right and also incorporated a toolkit that allowed other applications to invoke it. QuickTime functions were often invoked from HyperCard stacks through the use of the QuickTime Toolkit stack. There were three often-used QuickTime utilities: Movie XCMD, QTMovie XCMD, and Wild Magic. The Movie XCMD and QTMovie XCMD each invoke functionality from the QuickTime software extension. (See [Drucker92]

at 4); ([Borrell93] at 1-4.) This software extension includes various software components that enable playback of and interaction with QuickTime movies. ([Borrell93] at 1-4.) "The QuickTime extension provides three major pieces of new system software: the Movie Toolbox, the Image Compression Manager, and the Component Manager. Application programs use these software modules to let you play back or record QuickTime movies." (See [Borrell93] at 2.)

#### I. Movie XCMD

175. One played movies in HyperCard by invoking the *movie* XCMD in a script. According to [Drucker92], one displayed a movie in a window and then sent commands to that window to further control the movie. It was possible to open a movie invisibly, set several different properties of the window that contain the movie, and then display the movie when it is ready to go using the *show* command. There are several parameters that could be set by the *movie* command. Among them was the *borderless* parameter which shows the movie in a simple rectangle without a border so the movie appears to be graphics blending with the card. One could also set additional properties of a movie. Among these was *controllerVisible*, which determines whether the movie control widget can be seen. If the controller was not visible, one could set the parameter *badge* to *true* so that the movie title is displayed.

176. The basic syntax of the Movie XCMD was as follows, where "Movie" is the XCMD name and each of the parameters following that name are configurable parameters:

177. Movie <file name>, <window style>, <location>, <visible>, <layering>

178. In addition to the values that these parameters could take on, the Movie XCMD offered a variety of properties. The Movie XCMD's parameters and properties are set forth in Chapter 15 of [Drucker92].

## II. QTMovie XCMD

179. One could also play movies in HyperCard using the QTMovie XCMD. According to [Borrell93], the QTMovie XCMD allows one to play a QuickTime movie in a HyperCard XWindow (external window, a window to which HyperCard sends events such as mouse clicks, update events, and keyboard events) or directly in the card window. When one played a movie in a HyperCard stack, Apple's MoviePlayer appeared, complete with playbar, volume control, and pause and step buttons. There were four variations of the QTMovie command. If it was called with the *Direct* parameter the movie was displayed in the card window. If invoked with the *XwindowType* parameter the movie was displayed in an external window. The general syntax for a *Direct* movie is:

180. QTMovie OpenMovie, Direct, *filename*, location [, options...]

181. Once QTMovie was been invoked there were several HyperCard commands that could be issued to control movie behavior. Among these were *showController* and *hideController*, which acted to either show or hide the movie playbar.

182. The parameters and properties of the QTMovie XCMD were discussed in Chapter 10 of [Borrell93].

183. The QTMovie and its parameters and capabilities were also discussed in [KenDoyle94].

## III. Wild Magic

184. Also disclosed in [Borrell93], Wild Magic was a utility that allowed a person to add QuickTime movies to applications that didn't support QuickTime. With Wild Magic installed on a Macintosh one could paste QuickTime movies into any application that would allow the pasting of a PICT file, the Macintosh standard graphics format. The first step was to play a movie in the MoviePlayer application, select a portion of the movie and then click Copy. Then, when a word processor

document, for example, was open one could invoke the Paste command from the Edit menu. When a user opened the document, a user could click a Play button to cause the movie sequence to appear, complete with the playbar.

**viii. HyperCard in Networked Environments**

185. There were several examples of HyperCard being used in distributed network environments. [Goodman90], at pages 737-739 discussed accessing a stack on a remote file server. It seemed to be quite a transparent operation. He states, "If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack's pathname card for stacks, just as it would be if it were on your own hard disk." He goes on to describe methods for scripting for a shared stack.

186. Again, [Goodman90] at pages 725-729, describes the details of connecting a Macintosh by its serial port to different types of networks and controlling traffic from within HyperCard. He describes the availability of XCMDs and XFCNs to facilitate serial port connection. Apparently, there were few if, any, limitations to connectivity. He states, "In the *HyperCard Developer's Guide*, an example stack shows how to use serial port control to access CompuServe via a modem, all under HyperTalk script control. You can use the same techniques described there to automate access to any other computer (like a bulletin board service, MCI Mail, or Dow Jones News Retrieval) that offers asynchronous modem access." He even goes on to discuss how to connect directly to IBM mainframe computers.

187. An article titled "Mac to Mainframe Connection With HyperCard" appears in [Powers90]. The author was asked to develop software to enable a Macintosh to communicate with an IBM mainframe computer from within HyperCard. The article contains design details for the project and a complete C program listing. He described the task as, "Tri-Data Systems had completed a Mac-based programmatic interface to IBM mainframes and wanted to extend that

capability to HyperCard. Using a HyperCard stack a user should be able to use or write handlers through which a stack could communicate with an IBM mainframe. Traditional communication methods require a dedicated terminal emulator or a custom application program, but Tri-Data Systems wanted more: a HyperCard interface that would greatly simplify the development of custom interfaces by giving the user a powerful set of HyperCard tools combined with extensions for IBM mainframe communication."

188. An Internet-based approach to network communications with HyperCard is reported in [Morgan92]. He describes connectivity through TCP/IP (Transmission Control Protocol/Internet Protocol). His approach was to use several already-available software packages to accomplish his goal. These included any version of HyperCard, MacTCP, and the XCMDs from the HyperCardTCPToolkit. He describes in detail the steps necessary to achieve HyperCard-TCP/IP access and communication protocols. He describes two stacks he created that use the protocol: Mini-Atlas and ListManager.

189. In "The QuickTime Handbook", [Drucker92], there is a discussion about a network-based electronic mail system that delivered news video clips, still graphics and text to attendees at the 1990 EDUCOM conference in Atlanta. This application was actually a HyperCard stack with QuickTime movies using an XCMD external. The news content was provided by a direct microwave video link from CNN and was distributed throughout the conference venue by Ethernet for viewing on 65 Macintosh computers. The technical details of this project are discussed in [Hoffert91] and shown in the video at [PA-NAT-00000036].

## **V. World Wide Web Browsers That Existed By 1993 – 1994**

### **1. The Internet**

#### **a. Background – WWW**



190. The Internet is a global system of interconnected computer networks that use the standard Internet Protocol Suite (TCP/IP) to serve billions of users worldwide. The origins of the Internet reach back to research of the 1960s, commissioned by the United States government in collaboration with private commercial interests to build robust, fault-tolerant, and distributed computer networks.

191. Even during its infancy, researchers understood the appeal of using the Internet to distribute multimedia documents. For example, the Diamond system from BBN Laboratories was a system for creating, editing, transmitting, and managing multimedia documents across the Internet, which at the time was run by the Department of Defense. The Diamond prior art disclosed network transmittal of documents such that multimedia elements were combined into integrated, compound documents. Aside from graphics and voice, Diamond also supported spreadsheet elements within its compound documents. A key element of the distributed architecture disclosed in the Diamond prior art was the interprocess communications facility, which supported interactions among Diamond's components. [Thomas85]

192. In 1991 developers at the European Laboratory for Particle Physics (CERN) in Geneva, Switzerland, created the World Wide Web (WWW) as a way to organize and link related information. Since then, accessing the Web has arguably become the most important usage of the Internet.

193. Background on the origins of the Internet and World Wide Web are set forth in [Cailliau00].

**b. Protocols: HTTP, HTML and W3C**

194. HTML (Hypertext Markup Language) was the dominant markup language used to produce documents that were read by web browsers. HTML was

an SGML-family language which, when used with a particular DTD (Document Type Definition) specified a document type.

195. HTTP (Hypertext Transfer Protocol) was a stateless, lightweight, and extremely fast alternative to FTP (File Transfer Protocol) for network file transfers in the World Wide Web environment. In the videos I am submitting along with this report, I show demonstrations involving HTTP servers as it interoperated with various prior art systems.

196. The convergence around the protocols that now define the World Wide Web was due in large part to the establishment of the World Wide Web Consortium (W3C) in 1994. W3C was formed at the Massachusetts Institute of Technology by Tim Berners-Lee and Michael Dertouzos as a consortium of member organizations which maintain full-time staff for the purpose of working together in the development of standards for the World Wide Web. There are now over 300 such members. It is a tribute to the perseverance of the early web pioneers that W3C was formed, and continues to shepherd changes and enhancements to the World Wide Web. The formation and work of the W3C is discussed in [Cailliau00].

## **2. Mosaic**

### **a. Ordinary Operation**

#### **i. Overview**

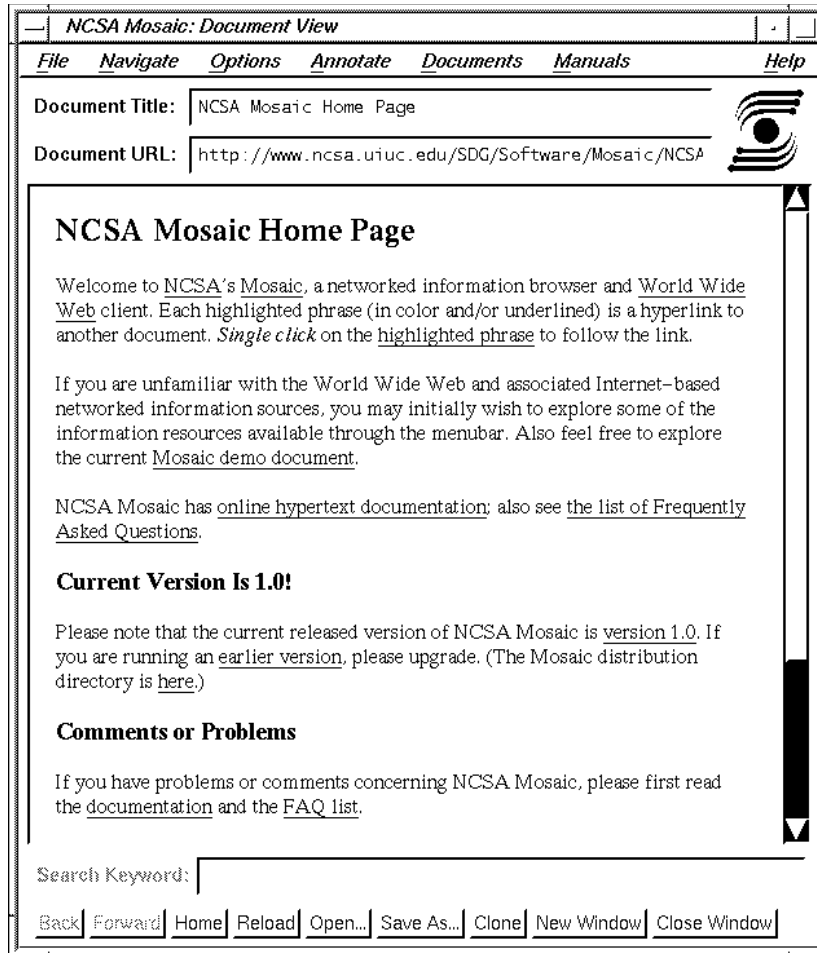
197. NCSA Mosaic was a distributed hypermedia system designed for information discovery and retrieval over the global Internet. Mosaic provided a unified interface to the various protocols, data formats, and information archives used on the Internet and enabled powerful new methods for discovering, using, and sharing information. Development of Mosaic began in December 1992. Versions 0.1-0.9 were the first developed and released. Version 1.0 was released on April 22, 1993, followed by two maintenance releases during summer 1993. Version 2.0 of NCSA

Mosaic was released in December 1993. Mosaic and its abilities were generally described in [Andreessen93b].

198. NCSA Mosaic used a client/server model for information distribution – a server sits on a machine at an Internet site fulfilling queries sent by Mosaic clients, which may be located anywhere on the Internet.

199. Units of information sent from servers to clients were simply termed documents. Documents could contain plain text, formatted text, inlined graphics, sound, and other multimedia data, and hyperlinks to other documents that could have been in turn located anywhere on the Internet.

200. The look of a Mosaic client for the X Window system is seen in Figure 11. Each underlined word or phrase is a hyperlink; clicking on it with a mouse button causes the client to connect to the appropriate server anywhere on the Internet and to retrieve and display the referenced document.



**Figure 11. Mosaic for X – Home Page**

201. Again, from [Andreessen93b], "The initial versions of the NCSA Mosaic clients have the following functionality:

202. • Graphical display of plain text, rich (formatted) text, and hypertext, as well as inlined access to graphs, images, audio clips, video sequences, and scientific data in multimedia and hypermedia documents.

203. • An intelligent graphical user interface featuring on-the-fly font and style selection, cut-and-paste support for formatted text, and extensive customization and user support options."

204. Other early features of Mosaic included HTML+ features, such as inline interactive Motif widgets, as disclosed in [Andreessen93a] and explained in further detail below.

205. I've prepared a video that demonstrates how Mosaic operated in the prior art. (See Appendix C, at Mosaic video 1.wmv.)

**ii. Helper Applications and MIME**

206. To allow interaction with non-native data formats for images, audio, video, and typeset documents that were popular on the Internet, Mosaic relied on a number of external viewers: programs separate from Mosaic that were invoked when necessary to display certain types of data. Such programs were called helper applications. MIME (or Multi-purpose Internet Mail Extensions) was a standard originally developed for attaching media messages to conventional Internet email. Mosaic had adopted this standard and used MIME file types for specifying its external Helper Applications. Some MIME types that were recognized by Mosaic were jpeg and gif images, mpeg and quicktime video, HTML and richtext text, and applications like postscript. To activate these helper applications, Mosaic passed them a file name and data type and they then opened in a separate or "pop-up" window, i.e. not within the browser window. Buttons on the external window allowed the user to control the application behavior and to dismiss the window. Mosaic's use of MIME files types and helper applications was described in the prior art reference [Branwyn94]. This was also described in the patents-in-suit in the specification (see, e.g., '906 patent col. 3:09-3:18) as well as in their file history, as I describe below and as can be seen at, e.g., PH\_001\_0000783895, PH\_001\_0000783958, PH\_001\_0000784041-043, PH\_001\_0000784144. I also prepared a video showing how Mosaic operated in the prior art. (See Appendix C at Mosaic Video 1.wmv.)

**iii. Widgets**

207. With the release of Mosaic 2.0 for X Windows, [Andreessen93a], it became possible to embed Motif widgets inside Mosaic documents. This capability arose due to partial implementation of the HTML+ specification, discussed above. In particular it was pointed out that documents could specify interactive fill-out forms – with input elements including text entry areas, toggle buttons, selection lists, popup menus, etc. – and Mosaic would instantiate such fill-out forms as sets of Motif widgets embedded inside the document. Further, it stated "This provides a way to provide arbitrarily sophisticated front-end interfaces to databases and search engines, as well as other network services – e.g., ordering pizza." The announcement also emphasized the importance of the sophisticated new support for inlining Motif widgets into documents, which enabled the fill-out forms support described above.

#### iv. Distributed Applications

208. NCSA Collage, [Collage92], was a scientific data analysis tool from the Macintosh group of the Software Development Group at NCSA. In a networked environment, this tool provided the capability to distribute most of these data analysis and visualization functions synchronously among a number of users. This was the foundation for the collaborative aspects of this tool's functionality. Ultimately Collage was made available not only on the Macintosh but on X-Windows and PC-compatible platforms as well.

209. Mosaic clients communicated directly with Collage, as well as with Polyview, NCSA's collaborative tool for three-dimensional geometric and polygonal data analysis. Scientific data contained within a hypermedia document could be sent across the network to one of these programs for graphical and statistical inspection, and analysis. This interoperability was disclosed in [Andreessen93b].

210. Among Collage's many features was the ability to establish communication with remote processes, e.g. a simulation running on a supercomputer. These remote processes could be controlled remotely, and images

and data could be transported to and from the remote process. Moreover, one could perform most of these operations not only on one machine, but on any machine that was participating in a collaborative session with NCSA Collage.

211. Consequently, collaborators using Mosaic clients and involved a Collage session could, for example, open and view an HDF (Hierarchical Data Format) file that was produced by a supercomputer computation. Members of the session could also annotate the displayed image to point out significant features.

**b. www-talk**

**i. Suggestions on Inline Viewing**

212. As part of my review of prior art, I read postings on the www-talk forum, which I understand was indexed, archived, and publicly accessible in 1993. Matthey Grey's post, dated August 19, 1993, and available at <http://ksi.cpsc.ucalgary.ca/archives/WWW-TALK/www-talk-1993q3.messages/736.html>.)

213. While Mosaic supported a wide range of helper applications, the www-talk community was pushing for a more convenient and generalized solution. These discussions took place on the publically available and published www-talk message board.

214. The idea of moving the display of an object from a helper application to a display embedded in the hypermedia document was well-known. As Mr. Bina testified, Mosaic was one of the first web browsers to do so. Bina Tr. at 56:11-57:10; 100:20-101:23, and Exhibits 4 (1/31/93 "Addition of outlet to Ghostview for PostScript documents") and 7 (3/4/93 "Some multimedia support. Yew asked for it ... audio...MPEG, PostScript automatically recognized...Inlined MIME/multimedia support will be coming down the road, so this is all just a temporary hack"); McRae Tr. at 108:1-11 (discussing the EMBED tag); 124:16-125:20, 132:9-135:19, 136:3-137:10, 138:18-143:1, 144:6-145:8; 146:20-149:12 (discussing embedding applications,

including xv and video functionality) and Exhibits 37, 38, and 39. Mr. Bina testified that he embedded video into a webpage around the time of the alleged invention too. See, e.g., Bina Tr. at 33:8-35:22; 106:22-107:8; 191:14-20. Mr. Bina also saw Viola demonstrated. See, e.g., Bina Tr. 108:17-111:4; 153:9-154:10; 183:10-19.

215. As another example, over just a two day period in January, 1994 there were several forum messages in a thread titled "Inlined image format" that were urging browser developers to facilitate the display of media content directly in the browser window. From one of the first messages, from Chris McRae [www-talk-00028138], states "There used to be a patch for Xmosaic which allowed for inlining TIFF images. The work was done by Cheong Ang at UCSF. I think you should be able to get it via ftp to ftp.library.ucsf.edu." The next day that was followed by a post from Ellson, [www-talk-00028157], stating, "Beyond TIFFs, what I'd really like to see is an inline window that external viewers can use for their display so that we can have inline movies, inline shells, inline audio control panels... all without hard-coded extensions to the browser." Next, also from Ellson [www-talk-00028167], came the sentiment, "As an author, I should be able to assume that the browser can handle any format as an inline component. (text, image, sound, movie, executable tSipp for local 3D rendering .... ). The browser should either handle it directly, or hand it off to an external viewer. Isn't that an issue that each browser and/or each user can deal with independently under the control of .mailcap or similar mechanism?" Finally, from named inventor David Martin, [www-talk-00028162], seconding Ellson's first post, comes, "I agree with John Ellson that the browsers should be capable of acting as MIME compliant applications and be able to make use of external viewers; in addition, I agree that Mosaic should generalize the ability of an external application displaying w/in the boundaries of Mosaic." All these messages clearly point to a well understood desire and motivation for inline media display.



## ii. IMG Tag

216. In the www-talk forum frequented by WWW developers, there was a lengthy thread about the need for a new HTML tag for inlining or embedding an arbitrary image into a browser document. This was referred to as the IMG tag. After much back-and-forth, Andreessen states that in an upcoming release of Mosaic, "So, we're probably going to go with <IMG SRC="url"> (not ICON, since not all inlined images can be meaningfully called icons). For the time being, inlined images won't be explicitly content-type'd; down the road, we plan to support that (along with the general adaptation of MIME). Actually, the image reading routines we're currently using, figure out the image format on the fly, so the filename extension won't even be significant." [www-talk-00293002]. Thus, the IMG tag points to the url of any kind of image located anywhere in the Internet.

217. The IMG tag is one example of an "embed" tag. For example, Tim Berners Lee proposed (as an alternative to IMG) a tag that would represent figures as:

218. <a name=fig1 href ="fghjkdfghj" REL="EMBED, PRESENT">Figure </a>. [www-talk-00292991].

219. Other alternatives that were proposed in place of the IMG tag included tags with MIME-compliant typing mechanisms to enable embedding media clips. [www-talk-00292995].

220. Furthermore, Eric Bina testified that, during the time of claimed invention of the patents-in-suit, it was known in the art that, in order to embed either images or maps into a webpage, one could use coded information, such as a tag, to make a compound document. Such a compound document is a hypermedia document that is a mix of textual information and multimedia. See, e.g., Bina Tr. 56:11-57:10, 98:24-101:23; 120:14-124:9 (describing images, maps, and forms including how a user could interact with content embedded inline in the webpage).

iii. **Inline xv**

221. In early-to-mid 1993 there was a great deal of interest expressed by members of the WWW-Talk forum about the desirability of incorporating inline image manipulation into an HTML document. See, e.g., Bina Tr. Exhibits 4 and 7. That is, instead of using a helper application that opens a separate window, the sentiment was to embed a window within the document window where the results of image manipulation would be directly displayed. See, e.g., McRae Tr. 115:20-118:17 and Exhibit 37 at 25-26.

222. Some comments from that forum were, "Building viewers so that they can also operate on some pre-existing window, instead of just creating their own, is an excellent idea. I'm always surprised when otherwise-wonderful viewers, such as Ghostscript or xv, don't do this from the start." [www-talk-00293014]. And, another observation was, "The justification for using xv to display inlined images is listed right in your reply: xv displays \*many\* different image types. We are using mosaic to display a database of medical journals we have; however, since the page images are in TIFF format, I have to convert to GIF them all to GIF." [www-talk-00293020]. Further, "There is no way of knowing, *a priori*, what document formats we might encounter and we would like a client with general image display capabilities. Xv fills this role nicely". [www-talk-00293020].

223. One commentator described the methodology for achieving inline XV: passing a window ID to XV so that it appears as a sub-window within the X window of the browser. Bill Janssen replied that he in fact had a modified version of XV that supported the functionality of receiving a window ID, and indicated he would send around that modified version. [www-talk-00293011].

224. In the videos that I am submitting along with my expert report, I demonstrate the concept that Bill Janssen was describing: specifically, I pass a

window ID for a Viola browser to a modified version of XV so that the XV output is painted at an embedded location within the Viola browser window.

225. Another forum poster, Chris McRae [www-talk-00293020, June 26, 1993], disclosed another scenario involving inlining xv. He asks Andreessen, "Suppose someone (such as ourselves) were able to integrate tooltalk functionality into mosaic, \*and\* could provide an HDF- and tooltalk-capable version of xv (or some such image manipulation package). Would the technology fit into your development strategy for mosaic?" Such a proposition clearly presaged a scenario where, within the purview of a Mosaic browser, one had an embedded, interactive object managed by xv, ToolTalk interprocess communication, and access to HDF formatted data in a distributed processing environment.

226. The XV program being discussed is described in [Bradley92]. The description of the program states, "XV is an interactive image manipulation program for the X Window System. It can operate on images in the GIF ', JPEG, PBM, PGM, PPM, X11 bitmap, Sun Rasterfile, and PM formats on 1 -, 4- ,6-, 8-, 16-, 24-, and 32-bit X displays". Thus, after an inline invocation of XV, the control window shown Figure 12 would appear outside the browser window but graphical output would appear within.

227. I also have prepared a video showing prior art XV in use. (See Appendix C at XV.wmv.)

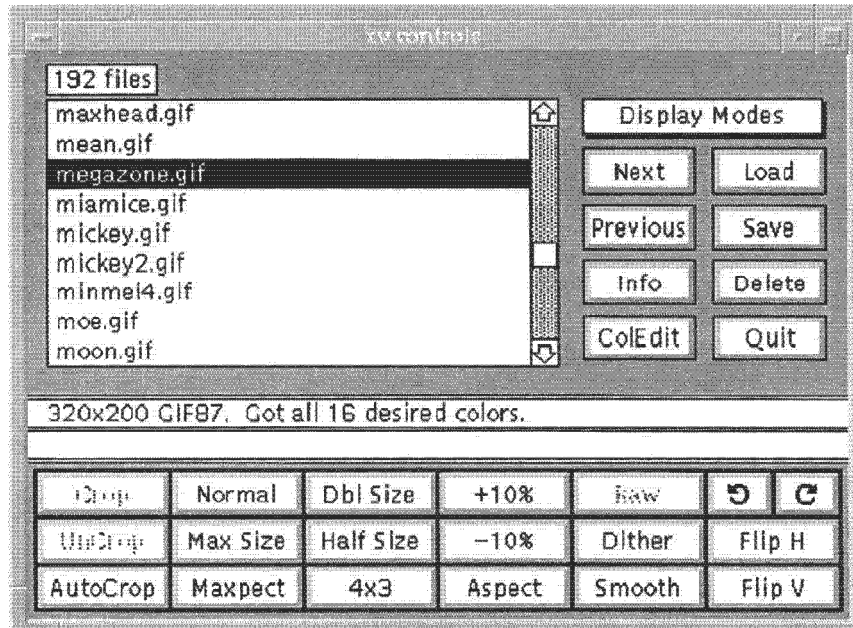


Figure 12. XV Control Window

iv. Interactive Widgets

228. The availability of an embedded, interactive Motif fill-out form widget in Mosaic 2.0 was announced in 1993, as described above. In [www-talk-00318887], Marc Andreessen requests the contribution of additional widgets that were to be developed by WWW-Talk members. The request was for non-standard, flexible, interactive widgets. The actual call was, "If all goes as planned, X Mosaic will be expanded in the not-too-distant future to be able to handle heterogeneous MIME documents. This capability will probably take the form of a MIME "master widget" that will create child widgets to display various data elements - text, GIF images, JPEG images, MPEG movies, and so on. The child widgets will in turn be able to have active elements of their own... a widget representing audio/basic will have a 'play' button, a readout giving the duration of the audio clip, and so on. Widgets representing unknown types will have controls to allow saving the data to a local file or possibly passing it to an external program".

### 3. Viola and ViolaWWW

#### a. Viola Toolkit/Language System

##### i. Overview

229. In the words of its developer, Pei Wei, [Wei94],<sup>10</sup> "Viola is a tool for the development and support of visual interactive media applications. Possible Viola applications range from a simple clock to a World Wide Web hypermedia browser (ViolaWWW)." Further, "Viola provides a quick and easy way to build GUI applications. The viola environment contains a GUI toolkit and a language interpreter. One can build and test GUIs without the usual edit/compilation/link cycle. The built in scripting language can be used to extend the default behavior of the GUIs, and to some extent general programming. The extension scripting language makes it possible to build entire applications, not just the GUIs, in viola. See, e.g., McRae Tr. 92:3-16.

230. At the basic level, the Viola system is a combination of the following major subsystems:

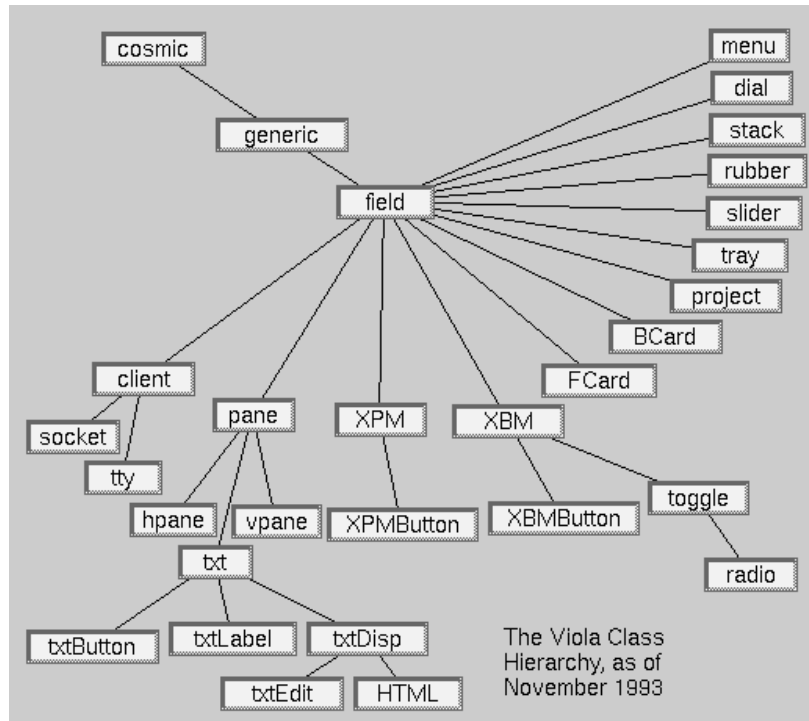
- An object oriented system.
- A scripting language for programming the behavior of objects.
- A graphical user interface toolkit.
- Specialized applications supporting libraries: XPM, GIF, WWW, style sheet, etc." See, e.g., Wei Tr. 146:22-148:19.

##### ii. Object-oriented Architecture

In developing Viola, Wei elected to use a single-inheritance object oriented language. This approach facilitated application development in Viola because data and methods could be encapsulated in reusable and manageable units. Wei defined a

---

<sup>10</sup> My citations to [Wei94] are exemplary citations. My assertions, characterizations, and opinions regarding the Viola prior art are based on my review of other prior art materials as well, including [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], and [Viola-March beta].



**Figure 13. Viola Class Hierarchy Tree**

hierarchy of classes, many of which were graphical user interface (GUI) oriented because emphasis was on development of hypermedia applications. The class hierarchy tree is shown in Figure 13. A description of each class and its application is in [Wei94]. The object-oriented architecture and the class hierarchy are also evidenced by a review of Wei's early codebases, such as the [Viola-5/12/93] codebase. The directory `viola\src` includes files that define and implement Wei's classes. Wei also included documentation about the class hierarchy in his early codebases. For example, [Viola-5/12/93] includes a document `viola\docs\violaChier.hmml` which references the object `viola\apps\chier.v`, and displays a visualization of the class hierarchy similar to what is shown Figure 13.

### iii. Scripting Language

231. Viola included a simple but powerful scripting language that resembles the C language in syntax. Furthermore, its simplicity is reinforced by its support of

only a few language constructs such as *if*, *do*, *while*, *for*, *switch*. All Viola objects could be individually programmed using the scripting language. See, e.g., Wei Tr. 146:22-148:19. Applications, which could incorporate many objects, were saved as files with a suffix \*.v. It should also be noted that \*.v applications and other Viola objects could interoperate and communicate with one another. Here is a simple example of a Viola application, which was defined in a file named hello.v.

```

\class {vpane}
\name {hello}
\children {hello.greet hello.bye}
\width {200}
\height {60}
\class {txtLabel}
\name {hello.greet}
\parent {hello}
\label {Hello there!!}
class {txtButton}
\name {hello.bye}
\parent {hello}
\label {Bye!}
\gapH {4}
\gapV {2}
\script {
    switch (arg[0]) {
    case "buttonRelease":
        exit(0);
    break;
    }
    usual();
}

```

The output  
file is this graphic.



resulting from running this

232. Wei's early codebases are replete with examples of \*.v applications. For example, [Viola-5/12/93] at `viola\apps` includes a variant of the hello.v example above, and dozens of other examples.

#### iv. Application Support Libraries

The Viola Toolkit/Language System included several libraries that supported hypermedia application development and WWW access. Among them were XPM (X Pixmap), an ASCII text image format used by the X Window System, and GIF (Graphics Interchange Format), a bitmap image format. Viola library support also

included access to various style sheets, i.e., documents that control the look of a web page. Viola codebases included the libWWW library (e.g., [Viola-5/12/93] at `viola/libwww`), which provided basic software components for connecting programs to the WWW.

**b. Viola in a Hypermedia Networked Environment**

233. Viola, though designed as a WWW browser, could operate locally (i.e., retrieve hypermedia documents and associated `.v` files from the local computer). Many software developers during that time tested and developed their software locally, because it was easier than testing over networks.

234. Viola could be used in local area networked environments. For example, one could use the Sun Microsystems Network File System (NFS) to mount a server computer for access by a client computer running ViolaWWW, so that ViolaWWW could retrieve hypermedia documents and `.v` files residing on the NFS server. See, e.g., Wei Tr. 360:23-25 (“Q: Did Viola use NFS to transport documentation over the web? A: It could use NFS”). NSF functionality was provided directly as part of SunOS [SunOS 4.1.3].

235. Viola could also be used on the World Wide Web through the HTTP protocol. See, e.g., Wei Tr. 352:20-352:10. As described above, HTTP was a standard protocol for interoperability with the World Wide Web, and this was the very purpose of Viola. (See, e.g., [Viola-5/12/93] at `\viola\apps\www.v`.)

**i. HTTP versioning**

236. Certain codebases of Pei Wei's Viola (such as [Viola-5/12/93] and [Viola-5/27/93]) were developed when HTTP was in transition from version 0.9 to version 1.0. In certain versions of Viola, Pei Wei updated certain portions of his browser to comply with version 1.0 but did not update other portions. ViolaWWW would therefore make requests for hypermedia documents according to HTTP 1.0, but was configured to receive data according to HTTP 0.9. See, e.g., Wei Tr. 80:19-



81:5 (“Q: Can you describe the server bug? A: Well, essentially, in one of the copies that was produced, there was a bug where the browser – the browser essentially thinks the world is in a 0.9 protocol world. But the new code that I had just installed is, in fact, using 1.0. So there’s a little discrepancy there, mismatch between the browser code and the Worldwide Web HTTP code, and it’s a simple matter of changing the if-def to tell the HTML library to use 0.9 so it matches the browser.”); 198:10-201:2. To fix this, one simply needs to comment out certain lines of HTTP version 1.0 so that the HTTP server's response is in accordance with HTTP 0.9. See, e.g., *id.* (“[I]t’s a simple matter of changing the if-def to tell the HTML library to use 0.9 so it matches the browser). Alternatively, one could modify the Viola source code such that it was configured to receive data according to HTTP 1.0. Another option is to use a matching server. See, e.g., *Wei Tr.*: 202:3-10 (describing three ways to fix the problem). In either case, a person of ordinary skill in the art would be able to identify this issue and then implement the modification. See, e.g., *Wei Tr.*: 200:10-15 (“Q: You understand that this is a problem that Microsoft tried to fix in the last case, did they not? A: Yes. I believe it’s quite simple.”). In the videos that I am submitting along with this report, I show how this can be done. (See Appendix C.)

**ii. [Viola-5/27/93] download location**

237. Because of a bug in one of Pei Wei's codebases ([Viola-5/27/93]), Viola looked in the wrong location for the files that it retrieved from an HTTP server. See *Wei Tr.*: 87:2-8 (“It’s the bug where one of the copies of Viola would download an app into one location, but when it’s actually rendering, it renders another copy in a different directory. But because they are the same app, they look the same, but it’s still a bug, nevertheless.”). This was because [Viola-5/27/93] stored files from servers in a folder called /usr/tmp, but looked in a different folder when trying to render those files. See, e.g., *Wei Tr.*: 278:6-279:5. This is a bug that one of ordinary skill could have identified and fixed, and there are at least two ways to overcome the bug: 1)

modify Viola code to point it to usr/tmp when rendering files; or 2) move the relevant files from usr/tmp to the location where Viola looks. In the videos accompanying this report, I show how one can identify this issue and make the necessary modifications. (See Appendix C.)

**iii. Alpha Release's interoperability with HTTP**

238. The issues described above in connection with [Viola-5/12/93] and [Viola-5/27/93] were no longer issues by the time of Pei Wei's October 16, 1993 "alpha" release [Viola-alpha]. As I demonstrate in the videos that I am submitting along with this expert report, [Viola-alpha] operated correctly with HTTP servers without any need for modification.

**c. ViolaWWW: Web Browser Based on Viola Toolkit/Language System**

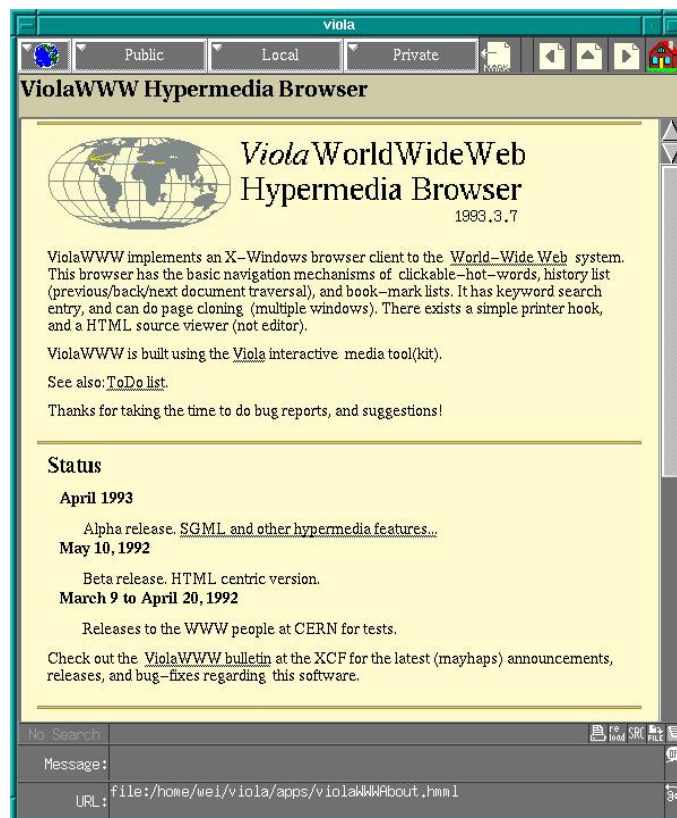
**i. Overview**

239. The most complex application written in the Viola scripting language and using the GUI toolkit was a World Wide Web browser, called ViolaWWW. Various versions of ViolaWWW were released during the 1991-1994 timeframe, with versions building on and adding more functionality to the ones previous. See, e.g., Wei Tr. 150:12-151:17; 49:22-49:5; 50:14-19. The versions I have analyzed for this report include [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], and [Viola-March beta]. While some of my discussion below focuses on one of these codebases (such as [Viola-5/12/93]) for easier explanation, the same architecture and operation of ViolaWWW existed for all these versions of ViolaWWW.

240. In the May 12, 1993, [Viola-5/12/93] Viola codebase, the ViolaWWW application file, [www.v](#), contained almost 2400 lines of code. Upon launching the ViolaWWW browser with the UNIX command, `viola -o www`, the browser shown in Figure 14 would appear. The videos accompanying my expert report show the result of this operation: Viola's default homepage ([Viola-5/12/93] at `\viola\docs\README_alpha.html`) appeared.

241. While the launch of ViolaWWW was accomplished with the UNIX command above, certain viola codebases included a script called RUN.WWW which was designed to automate the process of launching viola – one example is [Viola-5/12/93] at RUN.WWW, located in its root "viola" directory. This script allowed one to specify environment variables, including the WWW\_HOME variable that specified which document would load when the browser launched. Certain other codebases do not include such a RUN.WWW file, but it would be a trivial task for one of ordinary skill to create one. In the videos I am submitting along with this expert report, I show examples of such scripts for [Viola-alpha] and [Viola-Feb Beta].

242. The arrival of ViolaWWW on the World Wide Web stage was hailed by the early WWW pioneers. From the WWW history reminiscences in [Cailliau00] comes the statement, "Viola was to become the first X-browser to make any impact,



**Figure 14. ViolaWWW Browser Application**

but even his early versions went down well at CERN. "ViolaWWW works great," exclaimed Tim [Berners-Lee] on 24 January 1992.

243. Also, "Over the next few months, Pei Wei rewrote Viola and refined the browser that had so impressed Tim. As this ViolaWWW developed, it was to set the standard for everything to follow." Finally, also from [Cailliau00], "His idea was to add interactivity to the Web so that it actually became the graphical user interface. You wouldn't need to have different programs to do spreadsheets or word processing or surfing the Web, you would do it all in ViolaWWW."

**ii. Embedded, Interactive Objects**

244. From the earliest days of ViolaWWW development Wei intended to make it possible to embed interactive objects directly in the browser window. He had discussions about this goal with several colleagues from the Web community. For example, in a 1991 email [www-talk-00293029] to Tim Berners-Lee, Wei stated, "One thing I'd like to do soon, if I have time, is to teach the parser about viola object descriptions, and basically embed viola objects (GUIs & programmability) into html files." In a 1993 email [www-talk- 00293041] to Marc Andreesson, Wei states, "Well, the new violaWWW will support SGML documents. It does one particular DTD now (mine:-), but it's designed to be extensible to other DTDs. So, when the WWW DTD is stabilized, I'll add it. Unlike the old HTML widget, the new "widget" can embed bitmaps and viola objects into the document page."

245. Then, in an email [www-talk- 00293059] to Dave Raggett and Tim Berners-Lee, Wei describes his HyperMedia Markup Language, (HMML), and the tag he uses for embedding applications, <VOBJF>. According to Scott Silvey, HMML was designed to be a "superset" of HTML. It therefore was designed to encompass the functionality of HTML, in addition to adding more functionality to it. Viola, or any browser, which supports HMML would also be capable of displaying HTML type pages. Silvey Tr. 131:8-15. Further, according to Wei's description, "HMML is a

very simple DTD that exists for the sake of working out the browser technology. So, it has no pretense of having the characteristics to make it a candidate as a universal DTD." He further says, "Viola nowadays can handle at least HTML and HMML DTDs. So we're going in the direction of being able to handle multiple DTDs." Finally, "<VOBJF> is used to insert any viola application. This provides \*lots\* of possibilities. But it's also poses a security issue. So we use it for demos, but will probably not release it, unless we can figure out some kind of security mechanism." In general, Wei's Web community correspondence is replete with similar examples.

### iii. Further Development

246. Wei began development of Viola while he was a student at the University of California at Berkeley. Wei Tr. 13:1-16:7. While there, Wei made Viola freely available, and during the period around 1992 hundreds of copies were downloaded, as seen from the corresponding log of FTP activity, [FTP92]. See, e.g., Wei Tr. 378:8-19; 379:2-14. Wei subsequently joined O'Reilly and Associates in Berkeley, where Viola was further developed. Wei Tr. 32:16-23; 36:4-19; 37:16-38:1. Ultimately, embedded, interactive objects were reduced to practice at O'Reilly. In fact, based on my analyses of the various codebases, I have determined that they were reduced to practice no later than May, 1993. I have also determined that this functionality existed in all Viola codebases I analyzed, including [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], and [Viola-March beta].

247. Wei gave multiple demonstrations of Viola to others from May to October of 1993. Wei shared the Viola code and did not place any restrictions on these demonstrations. See, e.g., Wei Tr. 379:2-382:18.

248. On May 7, 1993, Wei gave a demonstration of Viola to a number of people, including Scott Silvey, Dale Dougherty, and two engineers from Sun. Wei Tr. 152:10-155:20; 165:3-20. Wei believes that the Viola code from May 12, 1993 is essentially the same as what Wei showed at May 7<sup>th</sup> demonstration. Wei Tr. 174:9-13;

175:14-176:3. For example, the code contained "VOBJF", Plot.v, testPlot.HMML. Wei Tr. 192:6-11. On May 31, 1993, Wei e-mailed the two engineers from Sun, James Kempf and Karl Jacob, a link to a "well-known" FTP site to download Viola. See Wei Tr. 230:23-238:13 and Wei Exhibit 18.

249. In July of 1993, Wei gave another demonstration of Viola at a workshop at the World Wide Web Wizards conference. See Wei Tr. 157:18-158:6; 282:5-286:7. McRae also recalled seeing the Viola demonstration at the conference. See, e.g., McRae Tr. 100:24-101-25; Bina Tr. 108:23-109:24 and exhibit 3; Silvey Tr. 144:3-16. Wei believes features related to embedded technology were present in all versions of Viola beginning from May 1993 through 1994-1995. Wei Tr. 292:15-20.

250. In September, 1994, Wei presented a talk at the Stanford Computer Forum WWW Workshop where he talked about ViolaWWW progress at O'Reilly. [Wei94b]. His title for the talk was "An Architecture of Extensibility and Plug-in Components in Browsers." One of the applications discussed was an embedded "chess board application that is published on the web server and instantiated locally by the interpreter. A nice [feature] is that you could have high interactivity without incurring a lot of bandwidth- you can move pieces around, and the board can do simple checks for illegal moves, and transmit only the essential movement information."

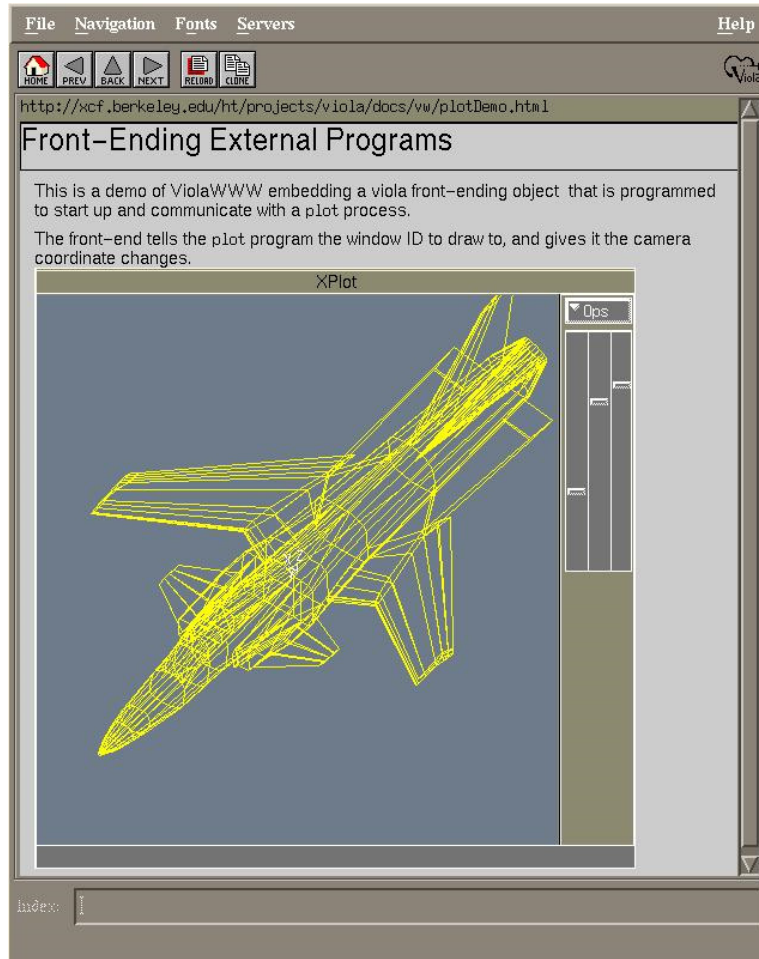
251. Another example he discussed is what he described as a "front-end application to a backend. And, the back-end is what actually does the computation and the drawing." In this particular case, the applet (plot.v) starts up another program (called vplot) on the same machine that viola is running on, and establishes a communication channel between the viola applet and the vplot program. The viola object tells vplot the X-Window ID of the window in which vplot could paint into (the area on the webpage that will display the graphic), and any x/y/z camera angle changes. In this way, when the user operates the sliders (changing the view angles),

this information is transmitted to the vplot program, which then draws in the given view window. This example is depicted in Figure 15. This same example is present in early codebases. For example, [Viola-5/12/93] includes plot.v and a document testplot.hmm1 that embeds plot.v in a ViolaWWW browser window.

252. For either example Wei notes, "It's important to point out that no special modification was made to the browser to make these applications. This is possible because all the basic primitives are already in the interpreter engine, and the rest is put together using the scripting language."

253. All versions of Viola that I analyzed provided the ability to show a hypermedia document that references plot.v so as to invoke an executable application, such as vplot, including [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], and [Viola-March beta]

254. While the "front-end/back-end" example shown in Figure 15 uses the vplot binary executable, it's easy to demonstrate how, with minor modifications, the same plot.v front end can work with other back-end binary executables. Videos submitted with this report demonstrate the use of MMPEG, XV and VIS binary executables, all in conjunction with the plot.v applet. (See Appendix C.)



**Figure 15. Front-End External Program Application**

**d. Additional ViolaWWW Architectural Issues**

**i. HMML and HTML Documents**

255. As stated above, early versions of ViolaWWW supported both HTML and HMML document formats. Wei developed HMML because he desired more capabilities than were then available in HTML. In particular, Wei wanted to implement embedding or inlining of hypermedia material directly in the browser window. While HTML+, which included embedding, had been proposed, it was not yet a WWW standard. HMML is, of course, just one type of DTD. Its definition can be



found in the early Viola codebases at the UNIX directory tree [Viola-5/12/93] at /viola/src/hmml.dtd. In the same codebase Viola included an HTML DTD at [Viola-5/12/93] at /viola/src/html.dtd.

256. As early as 1993 however, Wei was working to develop the HTML functionality of his browser. In [ora-00318281], a Viola status report, he states, "\* HTML stuff. \* Continue to add and migrate as many tags from HMML (my version) and HTML+ (as Dave has it) into HTML. \* Find out from the Design group what tags \*are needed\*." The "Dave" referred to is Raggett, the proposer of HTML+.

257. The versions of Viola that I analyzed supported features of both HMML and the base functionality of HTML including [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], and [Viola-March beta]. By design Viola, or any other browser which supports HMML, would also support the functionality of HTML, which was a moving target at the time since it had not been adopted as a standard.

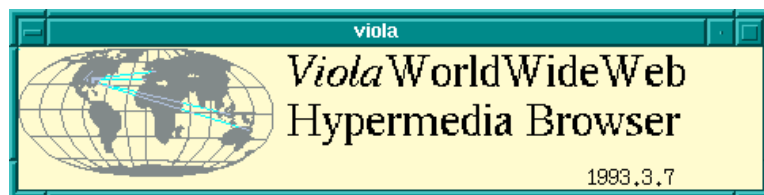
#### ii. HMML and HTML Document Parsers

258. Just as with other browsers of the day, ViolaWWW used a type of parser to convert a document's markup language text to a graphical screen rendering. The early Viola codebases Wei distributed either had a built-in HMML parser or assumed it was installed on the destination machine. In a 1993 email from Wei to prospective Viola downloaders he cautions, "The tar is /home/wei/violaTOGO.tar.z which you can drop in any directory. Beware that sgmls needs to be installed on your system." [ora-00293088]. One parser from that time period which was available to be used is still available at <ftp://ftp.jclark.com/pub/sgmls/>. Moreover, materials to which Pei Wei had access reveal how readily available parsers were. For example, [Viola-Alpha] includes code for an SGML parser at violaTOGO/src/spider. Further, Scott Silvey (creator of vplot) had backup tapes from the time during which he

worked with Pei Wei that included code for a prior art SGML parser, at [PA-NAT-00000024] in the folder 047263-000004\ net\ marble\ usr\ users\ source\ sgmls\.

### iii. Tags for Embedded Objects

259. As stated in above, Wei, in a 1993 email to Raggett and Berners-Lee, described the <VOBJF> tag as the method he used for embedding objects in a ViolaWWW document. See Wei Exhibit 25, at 2. At that time the tag was used in



**Figure 16: ViolaWWW Title Banner App**

HMML documents, but later Wei used it in HTML documents as well. The use of that tag can be found in the 1993 codebase in the file /viola/docs/testplot.hmml and /viola/docs/violaChier.hmml.

260. In the February, 1994 codebase release one finds evidence of support for the <VOBJF> tag in the file /viola/src/libWWW/Library/Implementation/HTMLDTD.h. In either DTD, the syntax for using <VOBJF> is the same: <VOBJF>[filepath]/[filename.v]</VOBJF>, where filepath is the UNIX directory path to the \*.v file, filename.v. In [Viola-5/12/93], one finds a detailed example of object embedding. In Figure 14, above, the banner that appears at the top of the browser window is actually an animated object; the lines on the globe graphic flicker and change position. That object viewed by itself is seen in Figure 16. These Figures depict what can be seen by loading the violaWWWAbout.hmml hypermedia document using [Viola-5/12/93].

261. The embedding of this applet by use of the <VOBJF> tag can be seen in the following fragment from the corresponding HMML file [Viola-5/12/93] at \viola\apps\ violaWWWAbout.hmml:

```
<!DOCTYPE hmml SYSTEM>
<HMML>
<TITLE>ViolaWWW Hypermedia Browser</TITLE>
<SECTION>
<HSEP>gold</HSEP>
<VOBJF>/home/wei/viola/apps/violaWWWtitle.v</VOBJF>
[....]
```

262. All versions of Viola that I analyzed provided the ability to embed objects using the HMML VOBJF tag, including [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], and [Viola-March beta]. HTML support for VOBJF was provided in at least [Viola-Feb Beta], and [Viola-March beta].

#### iv. LINK Tag in HTML

263. In a 1994 email, [Wei-00318312], announcing new developments in Viola, Wei describes a new embedding tag as, "<LINK REL='vobj'> - very flexible way to embed mini applications (ie: customized forms, updating fields, buttons-for-pop-ups, course-ware/ Hypercard-like widgys, etc) into a document, inlined." The full syntax is <LINK REL='vobj' HREF='[filename].v' ARG='[ ]'>, which Wei published in January 28, 1994 at [www-talk-00293096]. The codebases generally supported the use of LINK REL. Specifically, the February, 1994 [Viola-Feb Beta] codebase release shows the HTML incorporation of the LINK tag in the file /viola/src/libWWW/Library/Implementation/HTMLDTD.h.

264. The LINK tag was supported in at least [Viola-Feb Beta], and [Viola-March beta]. These versions also continued to provide functionality for the HMML VOBJF tag.

### v. Examples of Embedded, Interactive Objects

265. There are several examples of embedded applets in the various codebases. One simple applet is called Doodle. It can be found in the May 27, 1993 release [Viola-5/27/93]. Doodle, like vplot, is an example of an application controlling embedded interactive content that can be displayed within the ViolaWWW browser. The HMML file incorporating that applet is at /violaTOGO/docs/violaApps.hmml. The actual \*.v object is located at /violaTOGO/apps/doodle.v. An example of Doodle's graphical output is in Figure 17. McRae recalled that Viola allowed a user to “scribble” by picking a pen, a color, and dragging the mouse around on the screen. McRae further testified that the object

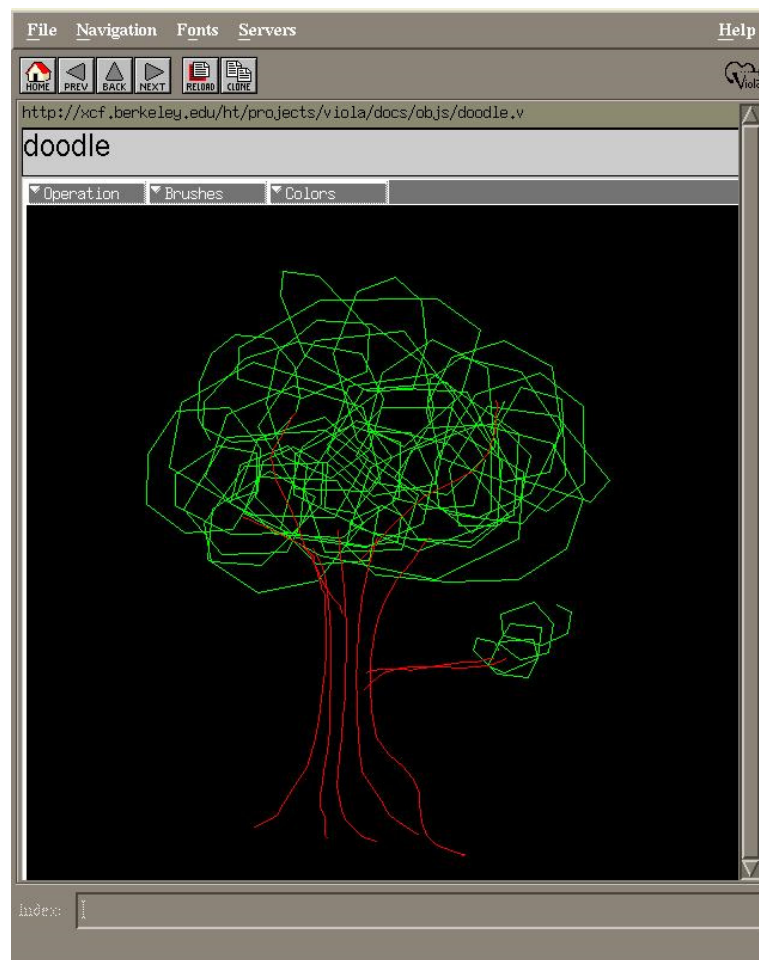


Figure 17. Doodle App

was displayed in the same browser window. McRae Tr. 93:1-22. McRae worked in the same group as the named inventors, was sent to the conference where Viola was shown, and Doyle himself approved the expense reimbursement for McRae's trip to Cambridge. McRae Tr. at 100:13-101:11 and Exhibit 33.

266. Another example of an embedded applet has already been shown in Figure 15 above. There, the Viola application described in the file plot.v, when interpreted invokes a compiled, non-Viola application called vplot to produce graphical output in an embedded view. In the May 12, 1993 codebase [Viola-5/12/93] the HMML file invoking plot.v is located at /viola/docs/testplot.hmml. See Wei Tr. 189:1-191.9. Plot.v itself is located at /viola/apps/plot.v. Note that the vplot code invoked by plot.v is a native binary module. Wei used UNIX interprocess communication techniques between the browser and vplot because they were in separate processes.

267. Vplot was also sometimes known as "xplot," before it was demonstrated to interoperate with ViolaWWW. An example of xplot can be found at [PA-NAT-00000024], in the folder 047263-000006\usr\work\xplot\xplot. I demonstrate this version of xplot in one of my videos. (See Appendix C at xplot.wmv.)

268. Viola's interoperation with doodle and vplot was present in all versions of Viola that I analyzed, including [Viola-5/12/93], [Viola-5/27/93], [Viola-alpha], [Viola-Feb Beta], and [Viola-March beta].

#### vi. Security flag for embedding in beta versions

269. During the examination of prior art witnesses, such as Pei Wei, Eolas focused on security issues with Viola. This issue, which was also raised by Dr. Felton during the reexamination, is largely irrelevant to the patentability of the claims. Reading the patent it is clear that it is not concerned with security issues, nor is it

apparent that it was on the mind of the inventors. In fact, Martin testified that it was not a concern. See Martin Tr. 374:18-376:21 (“...At the time, culturally, security for that type of application in terms of performing some nefarious task was not deemed a high risk...”). Nevertheless, during the prosecution of the patents, Eolas’s prior expert, Felton, submitted a declaration arguing that the patent solved security issues. See 9/27/2007 Felton Declaration at ¶ 66.

270. As broad as [Martin11] has applied the claims, they do not solve any security issues, and it is simply a design choice whether the browser application or the operating system identify and locate the executable application. In this regard, I note that if it is the browser application that identifies and locates the executable application, this does not solve the issue of the browser manufacturer being unaware of the potential scope of damage or security risk that the application running on the operating system might pose. Moreover, if the browser application were to permit any arbitrary script to be executed, which [Martin11] identifies as an executable application, this creates an even bigger security risk because the patent does not claim nor even disclose how it is going to contain some arbitrary script. To respond briefly to the security issues raised by Eolas, I note that in the beta versions of ViolaWWW (e.g., [Viola-Feb beta] and [Viola-March beta]), Pei Wei implemented a security setting that determined whether or not ViolaWWW would allow an interactive object to be embedded within the browser-window, or whether it would be displayed in a separate window. See Wei Tr. 52:19-22; 338:24-339:5. The reason for such a security flag is that when a browser is capable of interoperating with executable applications that it is not familiar with, it may not trust that executable application. To ensure that code in that executable application (about which ViolaWWW may not be aware) does not interfere with ViolaWWW's operation, the security flag prevents the embedding of objects managed by executable applications. If, on the other hand, ViolaWWW is familiar with the executable application, the

security flag would be changed as to allow embedding. See Wei Tr. 83:10-19 ("So I had introduced a flag, a security flag, into the Viola system so that when you download an application, that application will be marked as 'unsecure.' And if the system, if you compile the code to – basically, you can configure in a way that if you download these codes, they may get entered externally. Not inside the browser window; externally.").

271. Both beta codebases include a file called `sgml.c` (*see, e.g.* [Viola-Feb beta] at `\viola\src\viola`; [Viola-March beta] at `\viola940323\src\viola`.) Pei Wei introduced a security flag into these files as can be seen by source code in `sgml.c`: `"SET_security(obj, 1); /* non secure */."` The `SET_security(obj, x)` was the security flag, where the "x" can take the value of either a 1 or a 0. The default setting, as indicated by the code above, was "1," which corresponds to a non-secure mode. In this mode, HMML embedding was not enabled. If one attempted to embed an interactive object run by the `vplot` executable application the application's output would actually appear in a separate window.

272. However, if one changed the security flag to a 0 (i.e., `SET_security(obj, 0)`), this would correspond to "secure" mode, and embedding would be enabled.

273. This security flag suggests that embedding interactive objects was a design choice, and whether or not to implement embedding could depend on how a programmer felt about tradeoffs associated with embedding. One drawback to embedding is security concerns about allowing the browser to interoperate with executable applications of unknown origin that could have unknown and unstable properties.

#### **vii. Distributed Applications**

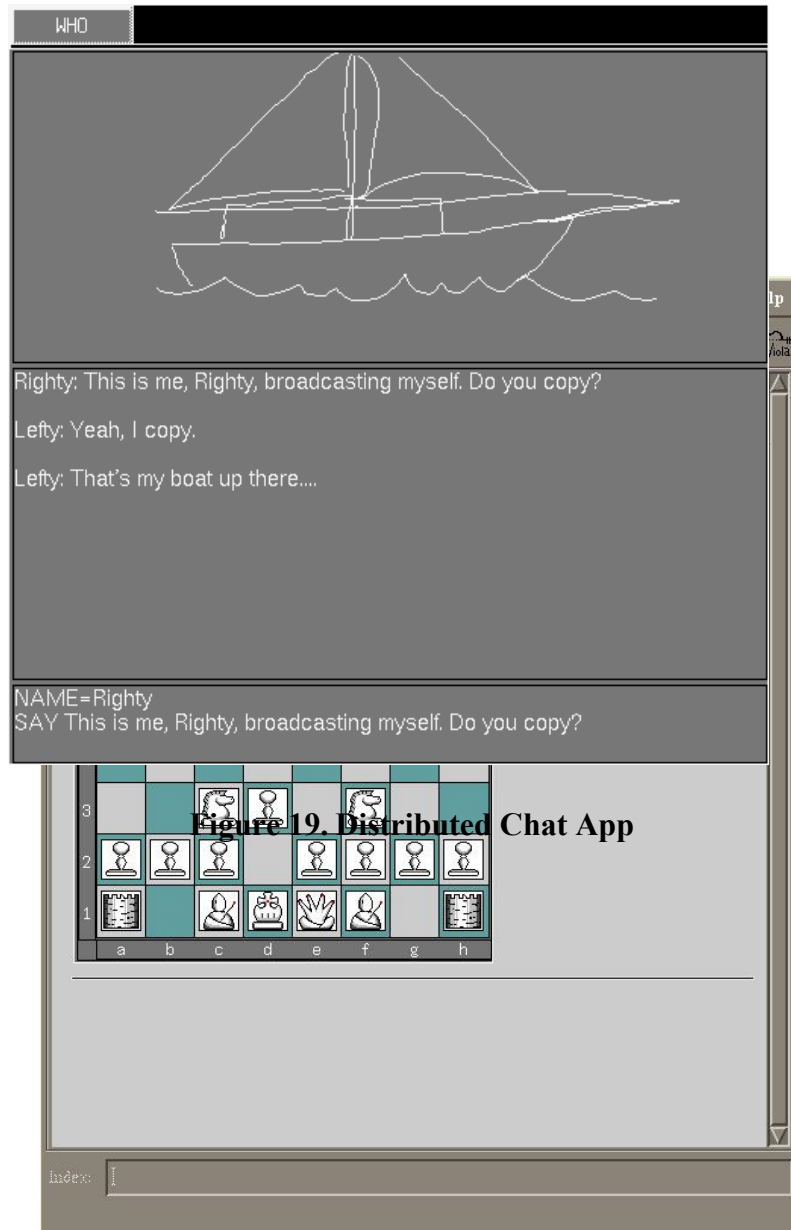
274. In a post to `www-talk`, [www-talk-00293096], Wei discussed the possibility of a client/server based Viola application, specifically an interactive chess

board. He states, "To implement the chess board interface, you'd \*basically\* write a mini viola application that consists of: objects for representing board-cells, chess pieces; an object doing the talking with the chess server; a clock object." Further, he states, "The ViolaWWW browser could fetch this chess-board-application (embedded in HTML or not) from a WWW server, and renders it on the fly." Later in 1994 he disclosed his chess application at a Stanford symposium, [Wei94b]. "This is a little mini chess board application that is published on the web server and instantiated locally by the interpreter. A nice [thing] is that you could have high interactivity without incurring a lot of bandwidth- you can move pieces around, and the board can do simple checks for illegal moves, and transmit only the essential movement information." The chess application is depicted in Figure 18. While 1993 versions of Viola did not include a complete implementation of the chess application, the mechanisms by which it could be implemented were well known and are described in this report.

Finally, in [www-talk-00318314], Wei talks about a distributed chat application and states, "What follows is a screendump of a demo of an embedded viola application that lets readers of this HTML page communicate by typing or drawing. Like the chess board application above, this chat application can stand alone (and have nothing to do with the World Wide Web), or be embedded into a HTML document. By the way, to make this possible, a multi-threaded/persistent server was written to act as a message relay (and to handle HTTP as well)." The chat example is seen in Figure 19.

Scott Silvey also discussed Viola's distributed computing functionality during his deposition in this case. There, he said, "I know that Pei [Wei] implemented some networking capabilities into Viola. He implemented the ability for programs to use sockets to communicate with other programs on the network, interprocess communication. Vplot - as part of the Vplot demonstration that we put together for





**Figure 19. Distributed Chat App**

**Figure 18. Distributed Chess App**

Sun and in developing that capability Pei developed or modified a class called the TTY object and that used interprocess communication.” Silvey Tr. 98 6-24.

## VI. Software development during early 1990s

275. In my review of the prior art references and systems described above, I have observed that the prior art systems used common tools, shared common aspects, and in some cases operated together. Persons of ordinary skill shared, used, and exchanged ideas about a common set of tools that were routinely used in the ordinary course of their development.

276. For example, developers of browser applications collaborated and shared ideas with one another. Pei Wei (creator of Viola), Marc Andreessen (creator of Mosaic), Dave Raggett (creator of HTML+) all communicated with one another, including on the www-talk forum, to exchange ideas and to collaborate. Even the named inventors (e.g., David Martin) and colleagues of the named inventors (e.g., Chris McRae, also later a colleague of Pei Wei) participated in these discussions. I cite many of these correspondences throughout my report. The same can be said for MediaView. (See, e.g., [McRae93].) In 1991-1992, Tim Berners-Lee at CERN developed a browser-editor on a NeXT workstation by developing a Hypertext subclass of the Text class (described in connection with MediaView below) provided by the NeXT development system. In fact, Berners-Lee expressed interest in obtaining MediaView source code to produce an enhanced browser. Moreover, the World Wide Web consortium archived a CERN web site that was maintained by Tim Berners-Lee. That site address is: <http://www.w3.org/History/19921103-hypertext/hypertext>. A link on that site called "Products" points to a page with the title "Systems and Applications." An introductory paragraph states: "Here is a list of existing systems, past present and future, which we have come across. These include both academic research systems and commercially available systems." One entry on that list is MediaView, with pointers to *Note from Dick Phillips* and *README*. The note in turn pointed to a Purdue University Internet repository where a MediaView application and document package could be obtained. The motive for Berners-Lee to

maintain such a list was to bring as much information together as possible, all to make the web grow. His last line on the list was "Thus spreads the web..."

277. Also, persons of ordinary skill in the art were routinely and in the ordinary course of development using their applications in networked, client-server environments, in some cases using local networks (e.g., using the Network File System I describe in this report), and in other cases connecting to the Internet. It was readily known how to do either. For example, Viola and Mosaic did both, and HyperCard XCMDs allowed HyperCard to do both as well.

278. The use of text formats, such as markup languages, to structure documents was routine and part of the ordinary course of software development. Mosaic used HTML and HTML+; Viola used HMML (which drew on HTML and also aspects of HTML+); MediaView used RTF; and HyperCard used (as one example) HyperTalk scripts. All were readily known to and could have been used by persons of ordinary skill in their software development.

279. Embedding objects was likewise routine and known to persons of ordinary skill in the art. One well-known method to achieve this was "passing a window id", a concept I discuss extensively in this report, and which persons of ordinary skill in the art discussed in connection with Mosaic and Viola. See, e.g., McRae exhibit 37 at 25-26 (June 1993) and 44-51 (April 1993). HyperCard and MediaView also used known programming constructs to embed objects. In fact, prior art developers sometimes chose to embed and sometimes chose not to embed; this was just a design choice. In HyperCard, for example, movie objects can either be embedded or displayed in external windows at an author's choice. In Mosaic, some objects (such as image objects) were shown inline, and I discuss www-talk emails below where persons of ordinary skill in the art weigh design tradeoffs in discussing whether other objects (e.g. XV-compatible objects) should be displayed inline.

Similarly, multimedia objects in MediaView were either embedded or shown in external windows, and it was known how to achieve both configurations.

280. Likewise, there were many tools readily available for interaction with objects. Some applications used separate control panels (e.g. XV, VIS) and some used controller bars (e.g., QuickTime, vplot). Handling user interactions could be achieved with standard programming tools, such as those that I describe above in connection with X Windows.

281. Above, I described well known tools for interprocess communication with external applications that were widely known and used by application developers, including creators of Viola, Mosaic, HyperCard, and MediaView. I also discussed that the use of distributed applications as that external application was well known for all these applications.

282. As discussed, these prior art systems used common tools, shared common aspects, operated together, and were part of the routine development that was occurring in the ordinary course during the early 1990s. Persons of ordinary skill shared, used, and exchanged these ideas and used a similar set of tools to achieve them.

## **VII. The patents-in-suit**

### **1. The patent specification and asserted claims**

283. Eolas asserts two patents in this action: U.S. Patent Nos. 5,838,906 (the "'906 patent") and 7,599,985 (the "'985 patent"). The '906 patent, titled "Distributed hypermedia method for automatically invoking external application providing interaction and display of embedded objects within a hypermedia document," was filed on October 17, 1994 and issued on November 17, 1998. The '985 patent, issued on October 6, 2009, is a continuation of an application, now abandoned, which was in turn a continuation of the '906 patent. Aside from a few minor wording changes, the

two patents share the same specification. The claims currently asserted are: claims 1-3, 6-8, 11, and 13 of the '906 patent and 1-11, 16-19, 20-28, 36-43 of the '985 patent.

284. The specifications of the patents-in-suit state that the "invention relates generally to manipulating data in a computer network, and specifically to retrieving, presenting and manipulating embedded program objects in distributed hypermedia systems." *See* '906 patent, col. 1:19-23. The inventors acknowledged in the specifications that the prior art already included numerous concepts found in the claimed invention, including:

- Retrieving and displaying "hypertext documents" or "hypermedia documents" that contain links to "data objects" located locally or "over a network," such as Apple's HyperCard. *See, e.g.,* '906 patent, col. 1:60-2:6; 2:55-65. The specification, however, incorrectly states that "HyperCard is a standalone system where the data objects are local to the user's system." As I show in this report, it was well-known by the time the named inventors filed their patent that HyperCard operated over networks.
- Displaying images "embedded" within the text of a "hypermedia document." *See, e.g.,* '906 patent, col. 2:66-3:3.
- "[E]mbedding interactive program objects in documents," such as through the Microsoft OLE technology and Apple OpenDoc technology. *See, e.g.,* '906 patent, col. 3:52-54.
- "[S]pecifying and locating" a "linked object" using HTML. *See, e.g.,* '906 patent, col. 2:37-47.
- Using a "browser program," such as the Mosaic browser, to automatically invoke a "viewer program" to display "objects" such as images. *See, e.g.,* '906 patent, col. 2:66-3:26.
- Using an external application, such as "VIS" and "Panel," to display and enable a user to interact with data objects. *See, e.g.,* '906 patent, 10:17-27; 15:60-64.
- Enabling a user of the "browser" to "launch" a "data object to automatically execute" computer code. *See, e.g.,* '906 patent, col. 3:33-51.
- Using "client" computers to "request" information from a server, and using "servers" to provide information to "client" computers. *See, e.g.,* '906 patent, col. 4:32-59.

285. The claims of the patents-in-suit consist primarily of these elements that were well-known in the prior art, including using a client workstation to request a hypermedia document from a server, executing a browser application on the client workstation to display the document, and invoking a separate executable application to display embedded interactive objects within the document. For example, independent claim 1 of the '906 patent recites:

286.

1. A method for running an application program in a computer network environment, comprising:

providing at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment;

executing, at said client workstation, a browser application, that parses a first distributed hypermedia document to identify text formats included in said distributed hypermedia document and for responding to predetermined text formats to initiate processing specified by said text formats;

utilizing said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server, wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation, wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies the location of at least a portion of an object external to the first distributed hypermedia document, wherein said object has type information associated with it utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document, and wherein said embed text format is parsed by said browser to automatically invoke said executable application to execute on said client workstation in order to display said object and enable interactive processing of said object within a display area created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.

287. The other asserted independent claims of the patents-in-suit are variations of claim 1 that generally share similar elements. Independent claim 6 of the '906 patent and independent claims 1, 16 and 36 of the '985 patent also claim methods and systems for running an application program, independent claims 20 and 40 of

the '985 patent discuss methods of serving information, and independent claim 24 of the '985 patent is directed to a method for running an executable application.

288. The asserted dependent claims of the patents-in-suit are minor variations on the asserted independent claims and incorporate basic principles already well-recognized and well-known in the prior art or enumerate obvious implementation choices already known to one of ordinary skill. For example, dependent claims 2, 3, 7 and 8 of the '906 patent requires that the browser and the executable application communicate with each other, dependent claims 11 and 13 of the '906 patent requires that the client computer and server communicate with each other, and the asserted dependent claims of the '985 patent are generally directed to parsing the file received from the server (claims 5 and 6), the ordering and specificity of the text format and embed text format (claims 7-10), further obviating user interaction when the executable application is invoked (claim 11), and the type of information received from the server, such as requiring that the information comprise text formats (claims 2, 17, 21, 25, 37 and 41), HTML tags (claims 3, 18, 22, 26, 38 and 42), or at least one embed text format (claims 4, 19, 23, 27, 39 and 43).

289. The alleged novelty of the patents-in-suit lies, therefore, in *automatically* invoking an executable application and displaying embedded interactive content on a web page within the *same* web browser window, in contrast to manually clicking to execute the external application or view the embedded content in a separate window. The inventors of the patents-in-suit implemented this alleged novelty of their invention by visually combining two prior art software programs: Mosaic and VIS.

290. The National Center for Supercomputing Application's (NCSA) Mosaic software is identified in the specifications as an example of a prior art browser program developed by the University of Illinois at Urbana/Champaign, Ill. The inventors further acknowledged that "many viewers exist that handle various file formats such as '.TIF,' '.GIF,' formats" and that when a browser program, such as

Mosaic, "invokes a viewer program, the viewer is launched as a separate process" and the "view displays the full image in a separate 'window.'" See '906 patent, col. 3:9-20. I also prepared a video showing how Mosaic operated in the prior art. (See Appendix C at Mosaic Video 1.wmv.)

291. The VIS software program is identified by the inventors as an example of a viewer program they developed at the Center for Knowledge Management at the University of California, San Francisco. The inventors authored several papers about VIS, dating over one year before the '906 patent was filed. See, e.g., Doyle, et al., *Processing of Cross-Sectional Image Data for Reconstruction of Human Developmental Anatomy from Museum Specimens*, February, 1993. VIS allowed researchers to visualize complex scientific data in three dimensions and was designed to process large volumes of data quickly and in real-time. See, e.g., Doyle, et al., *Medicine Meets Virtual Reality II*, [Doyle94], at pg. 4; '906 patent, col. 5:31-55. The inventors were able to accomplish this because VIS operated on powerful "workstations," existed as a compiled native binary program, as opposed to an interpreted application, that could be directly executed on the "workstations," and could make use of "distributed computing." See Doyle Declaration ([ '858-PH] at PH\_001\_0000787072); ([ '831-PH] at PH\_001\_0000785963); [Doyle93]. I prepared videos showing how VIS operated in the prior art. (See Appendix C at VIS - A.wmv and VIS - B.wmv.)

292. The inventors' preferred embodiment was a visual combination of Mosaic and VIS such that the output from VIS was automatically invoked, obviating the need for a user to click, and displaying the output inline within the same browser window. As described earlier, Mosaic was already capable of identifying and using helper applications to display interactive objects in a separate window after a user clicks on a link to the object. The inventors defined an HTML tag called EMBED that had a "type" field, which used the same MIME type information that Mosaic used to identify helper applications, and a "HREF" field, which was a standard HTML



attribute used to specify the location of the object. *See e.g.*, '906 patent, Table II; 2:66-3:31. The inventors modified the source code to the publicly available NCSA Mosaic browser, version 2.4 such that when the modified browser parsed an HTML file and identified the EMBED tag, it would automatically launch VIS without a user clicking on a link, and the output from VIS would be displayed within the Mosaic webpage at the location of the EMBED tag. *See, e.g.*, '906 patent, 14:9-16:46. I prepared a video demonstrating the named inventors' preferred embodiment. (See Appendix C at Mosaic video 2.wmv.)

293. The inventors used standard well-known programming tools in modifying the source code to Mosaic. The inventors added routines in the HTMLparse.c file of Mosaic that checks to see whether each item parsed (e.g. word, tag or symbol) from a hypermedia document is an EMBED tag. If an EMBED tag is identified, a standard routine in HTMLparse.c is called to assign an enumerated type to the tag, which is an identifier with a unique integer associated with it. Once all of the text in the hypermedia document has been parsed, routines in a separate HTMLformat.c file are called to process the enumerated type using standard X-Window techniques. In particular, the enumerated type is processed as if it is a regular X-Window Motif/XT widget, and involves creating an internal object representation of the EMBED tag with values for the "width and height of a window on the display to contain an image, position of the window, style, URL of the image object, etc." The internal object is then passed to HTMLwidget.c, which uses standard X-Window routines to create a DrawingArea widget to display the data object. A routine checks as to whether the type attribute of the object is an application and if so, the appropriate application is launched according to a predefined list of application type/application pairs defined as a user-configurable XResource. The inventors also noted that an alternative embodiment could use the standard MIME database as the source of the list of application type/application pairs. The external

application, i.e. VIS, is started as a child process of Mosaic and is passed the window ID of the DrawingArea. VIS has its own process, Panel, which "creates the graphical user interface (GUI) thru which the user interacts with the data object."

Communication between Mosaic and Panel are handled through standard inter-client communication techniques provided in the X-Window environment, and communication between VIS and Panel are handled through a communication protocol, such as ToolTalk developed by Sun Microsystems and well-known at the time. *See, e.g., '906 patent, 14:9-16:46.* Also, see my video of the preferred embodiment, Appendix C at Mosaic video 2.wmv.

294. As demonstrated in this report, the alleged novelty of the patents-in-suit is disclosed or rendered obvious by multiple prior art references.

**2. Claim construction**

295. I understand that the parties have agreed to the construction of several claim terms, which is summarized in the table below. I have considered these constructions and believe that the analysis presented within my report is consistent with them. Where terms are not identified in the table below I have applied the plain and ordinary meaning of the claim terms as they would have been understood to a person of ordinary skill in the art in the 1993-1994 time frame.

<u>IDENTIFY AND LOCATE</u>	
<u>CLAIM TERM</u>	<u>AGREED CONSTRUCTION</u>
type information . . . utilized by said browser to identify and locate [an / said] executable application	the identify and locate functions are performed by the browser
with the browser application: ... utilizing the type information to identify and locate an executable application	
utilize the browser to: ... utilize the type information to identify and locate an	

executable application external to the file	
type information is utilized by the browser to identify and locate said executable application	
with the browser application: ... identifying and locating an executable application	
executable application ... is identified and located by the browser	
<b><u>INTERACTION</u></b>	
enable interactive processing of said object	allow the object to be processed based on the user's interaction
[enable / enabling] an end-user to directly interact with [said / the / an] object	allowing a user to directly interact with the object
<b><u>INTERACTIVELY CONTROLLING</u></b>	
interactively control[ing]	controlling through back-and-forth interactions between a user and the controllable application

296.

297. Below is a table reflecting my understanding of the claim constructions as based on the Court’s August 22, 2011 Order (“Order”) and September 23, 2011 Order (“Supp. Order”):

CLAIM TERM	EOLAS’S CONSTRUCTION	DEFENDANTS’ CONSTRUCTION	COURT’S CONSTRUCTION/ DISCUSSION
<b><u>AUTOMATIC INVOCATION</u></b>			
automatically [invoking / invoke] [the / said] executable	automatically calling or activating the executable	the executable application is launched to permit a user to	the executable application is launched without user activation (Order at 13)

CLAIM TERM	EOLAS'S CONSTRUCTION	DEFENDANTS' CONSTRUCTION	COURT'S CONSTRUCTION/ DISCUSSION
application	application	interact with the object without any intervening activation of the object by the user	
executable application is automatically invoked by the browser	executable application is automatically called or activated by the browser		
<b><u>WORKSTATION</u></b>			
workstation	a computer system connected to a network that serves the role of an information requester	a desktop or deskside computer with an operating system and hardware designed for technical or scientific applications that provides higher performance than a personal computer	a computer connected to a network that serves the role of an information requester (Order at 23)
<b><u>NETWORK SERVER</u></b>			
network server	a computer system that serves the role of an information provider	a computer running software that is capable of executing applications responsive to requests from a client workstation, and that processes commands from a client workstation to locate and retrieve documents or files from storage	a computer system that serves the role of an information provider (Order at 23)

CLAIM TERM	EOLAS'S CONSTRUCTION	DEFENDANTS' CONSTRUCTION	COURT'S CONSTRUCTION/ DISCUSSION
<b><u>EXECUTABLE APPLICATION</u></b>			
executable application	any computer program code, that is not the operating system or a utility, that is launched to enable an end-user to directly interact with data.	a native binary program that remains separate from the browser and is not part of an operating system or a utility	any computer program code, that is not the operating system or a utility, that is launched to enable an end-user to directly interact with data (Supp. Order at 4)
<b><u>OBJECT</u></b>			
object	text, images, sound files, video data, documents or other types of information that is presentable to a user of a computer system.	information presentable to a user of a computer system, which is not a program and which does not include source code or byte code	text, images, sound files, video data, documents and/or other types of information that is presentable to a user of a computer system (Order at 20)
<b><u>HYPERMEDIA DOCUMENT</u></b>			
[first] hypermedia document	a document that allows a user to click on images, sound icons, video icons, etc., that link to other objects of various media types, such as additional graphics, sound video, text, or hypermedia or hypertext documents	a document received by the browser that includes links (specified by the hypertext format) to graphics, sound, video or other media	a document that allows a user to click on images, sound icons, video icons, etc., that link to other objects of various media types, such as additional graphics, sound video, text, or hypermedia or hypertext documents (Order at 21)
[first] distributed hypermedia document	[first] hypermedia document that allows a user to		a document that allows a user to click on images, sound icons,

CLAIM TERM	EOLAS'S CONSTRUCTION	DEFENDANTS' CONSTRUCTION	COURT'S CONSTRUCTION/ DISCUSSION
	access a remote data object over a network.		video icons, etc., that link to other objects of various media types, such as additional graphics, sound video, text, or hypermedia or hypertext documents (Order at 21)
file containing information to enable a browser application to display [, on] [said/the] [client workstation,] at least [a / said] portion of [a / said] distributed hypermedia document	the file contains information to allow the browser application to display at least part of a distributed hypermedia document.	a file containing information received by the browser that includes hyperlinks to graphics, sound, video or other media	"does not need further construction. When read in view of the Court's construction of '[first] hypermedia document,' the term will be understood by a jury" (Order at 21)
<b><u>TEXT FORMAT</u></b>			
text format	text that initiates processing	tags or symbols that specify document formatting	coded information that describes how the content of a hypermedia document is to be displayed by a browser application (Order at 14)
<b><u>EMBED TEXT FORMAT</u></b>			
embed text format	text format for embedding an object	a tag that specifies the object to be embedded at the location of the tag	coded information that specifies to a browser application that an object is to be embedded in a displayed hypermedia document (Order at 15)
<b><u>EMBED TEXT FORMAT / FIRST LOCATION</u></b>			
embed text format, located at a first location in	embed text format located at a first location in the	tag located at the place in the received	"In light of the Court's construction of 'text format,' this phrase

CLAIM TERM	EOLAS'S CONSTRUCTION	DEFENDANTS' CONSTRUCTION	COURT'S CONSTRUCTION/ DISCUSSION
said first distributed hypermedia document	first distributed hypermedia document.	document where the embedded object will appear within the displayed document	needs no construction...Regarding the 'specifies the location' language, the parties' proposed constructions are merely attempts to restate the claim language or an effort to include the plain and ordinary meaning." (Order at 16)
embed text format [which] correspond[s/ing] to [a / said] first location in the document	embed text format which relates to a first location in the document	tag located at the place in the received file where the embedded object will appear within the displayed document	"This phrase does not require additional construction beyond the Court's construction of 'embed text format.' The 'location' terms are clear and understandable. Defendants' proposed construction improperly includes a requirement that the term designate where the embedded object will appear within the displayed document. This proposed construction accrued based on a limitation that is later in the claim language. The claim goes on to recite '...in order to display said object...within a display area created at <i>said first location...</i> ' ('906 Patent, Claim 1 at 17:24-26). This later

CLAIM TERM	EOLAS'S CONSTRUCTION	DEFENDANTS' CONSTRUCTION	COURT'S CONSTRUCTION/ DISCUSSION
			limitation logically demonstrates that the embed text format location in the document is where the displayed object will appear." (Order at 17)
<b><u>DISTRIBUTED APPLICATION</u></b>			
distributed application	an application that may be broken up and performed among two or more computers	an application in which tasks are broken up and performed in parallel on two or more computers	an application that is capable of being broken up and performed among two or more computers (Order at 22)
<b><u>IDENTIFY AN EMBED TEXT FORMAT</u></b>			
identify[ing] an embed text format	detecting an embed text format	detecting an embed text format during parsing of a hypermedia document	no construction is necessary (Order at 18)
an embed text format . . . is identified	an embed text format is detected		

298. One term from the Court’s construction that warrants mentioning is the term “executable application.” For this term the Court initially adopted a construction that expressly excludes scripts (e.g. “native binary”) consistent with the plain meaning, and subsequently adopted the construction from the prior litigation, which was made before the reexamination of the ‘906 patent.

299. It is my opinion that a person of ordinary skill in the art’s would understand the Illinois Court’s construction of “executable application” to likewise exclude scripts because a script is not “any computer program code, that is not the operating system or a utility, that is launched to enable an end-user to directly interact with data.” A script is not launched in the fashion described in the Court’s construction because it must be parsed and compiled into an intermediate form (e.g.



byte code) that is then executed. I note for example that one of the Examiner's reasons for distinguishing Viola from the '906 patent was that the Viola clock script was not an "executable application" because it was script that required the browser to interpret and translate the script into binary code. See Director's Examination (831) pg. 54 "The Examiner finds that the Viola code publication does not fairly teach nor suggest that the browser automatically invokes an executable application, external to the hypermedia document, to display the object and enable interactive processing of the object, when the instant '906 patent claims 1 and 6 are properly accorded the broadest reasonable interpretation." See also, 1:99-cv-00626, Dkt # 815, at 20-21.

300. In this regard, in my initial report, I applied what I thought to be a broad construction in my initial report consistent with how I believe a person of ordinary skill in the art would understand the term. [Martin11] identifies scripts as being executable applications, and in Dr. Martin's supplemental report he does not retract this contention based on the Court's construction. In so doing, he purports to apply the Court's construction of executable application in a way that is much broader than, in my opinion and as I applied in my initial report, a person having ordinary skill would have.

301. While I do not agree that [Martin11] is correct in its application of the Court's construction, to the extent it is, then the scripts in Viola would anticipate and/or render obvious the claims of the patents because they are "executable applications" that enable interactivity in the browser-controlled window. For the same reason, a HyperTalk script or any other script in HyperCard would also be an "executable application." In my opinion, should "executable application" in the claims cover scripts, then such broad claims are not supported by the relatively narrow disclosure in the specification, as discussed below with regard to the 112 defenses.

302. I note that [Martin11] at ¶118 concludes that the term “browser” means “web browser.” [Martin11] provides no analysis to support this conclusion, and it is not the plain and ordinary meaning of the term to a person of ordinary skill in the art at the time of the alleged invention.

303. [Martin11] cites ‘906 patent, 1:61-64, but this does not discuss web browsers, rather it refers to hypertext documents (and not HTML documents). The paragraph continues to the top of column 2, which [Martin11] omits from the citations, where the patent’s example provides HyperCard as an example of a browser. I note that the patent’s description of HyperCard is incorrect since the system does not have to be a standalone system. For example, HyperCard stacks and data objects may be obtained from a remote system in a client-server environment or network environment such as NFS or AppleTalk. See, e.g., [Sadowski11].

304. [Martin11] also cites ‘906 patent, 2:7-10, but this is continuing the example (which again [Martin11] omits) that describes HyperCard as a browser. Likewise, the citation in [Martin11] to ‘906 patent, 2:19-3:15 does not limit the claim language. Specifically, lines 3:9-15 states, in non-limiting fashion, that Mosaic and Cello are two “example[s]” of browsers.

305. [Martin11] also cites ‘906 patent, 5:17-30, but this again does not say anything about “web browsers,” rather it describes client-servers generally in a network environment and is not limited to any browsers or any types of servers. Likewise, of Figures 1, 5, 6, 9, and 10, only Figures 9 and 10 state “Mosaic.” The other figures are general in nature.

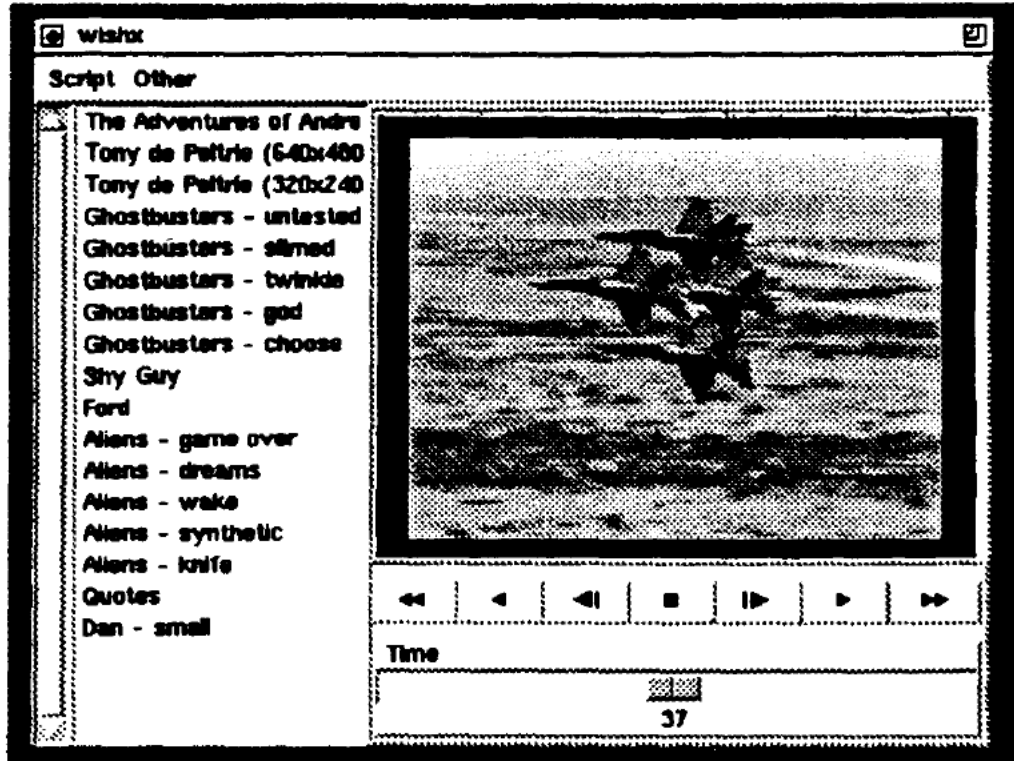
306. Under the logic of [Martin11] at ¶118, if the term “browser” means “web browser” then it should not appear in the literature discussing hypertext and hypermedia documents since “web browsers” were relatively new at the time.

307. However, a January 1992 paper published in the Communications of the ACM (Vol. 35, No. 1), by Bernard J. Haan et al., entitled “IRIS Hypermedia

Services,” discloses all the concepts cited in the portions of the specification cited by [Martin11] in a hypermedia browser that is not a web browser or a system that processes HTML files. The title of [Haan92] shares the term “hypermedia” with the title of the patent. [Haan92] discusses the Intermedia system, which was a hypertext system in the late 1980s according the article (see also [Meyrowitz86] cited in the patent and discussing the same technology). [Haan92] states that the term hypermedia was used “to emphasize the linking of graphics, animation, sound, and video with textual information.” [Haan92] at 38. [Haan92] describes the use of a client server database management system with a graphical user interface called a “browser.” [Haan92] at 38, 40 (“browser” left column, top), 41 (“browser” left column, top; client-server over the Internet, middle column, top), 46 (Fig. 5, discussing client-server), 49 (discussing Wide Area Networks or WANs mounting NFS file systems), etc.

308. Another reference, from 1982, and long before the World Wide Web, [Meyrowitz82] at 399 states: “While most are beyond the scope of this paper, we mention **browsers** as an interesting method for traversing hierarchies.” [Meyrowitz82] continues: “[t]he browser is important not only for its convenient techniques for tree traversal, but for its notion of letting a user browse through an entire collection of information, examining an editing at will. A **browser-like interface would be attractive in other environments as well...**”

309. Other references also show that the term “browser” is generic. See, e.g., [vanDam87] at 892 (keynote from Hypertext ’87, “despite whatever other experiences there are out there with programming environment **browsers**”); [Rowe92] at § 1 (“The initial application we are implementing to test our abstractions is a video **browser**...Figure 1 shows a screen dump of the **browser** interface.”), Fig. 1:



[Ohtsu93] at § 4.3 (“a more elaborate design of a file **browser** for Ensemble”) and References (citing another University of California paper entitled “The next best thing in file **browsers**”); [Hindus93] at 380 (“The Sound**Browser** project at Apple is the most closely related recent work.”), 384 (“The postcall **Browser** application displays the entire conversation and provides additional editing functions”), 387 (“The **Browser**’s status display was considerable less understandable”), and Fig. 3:

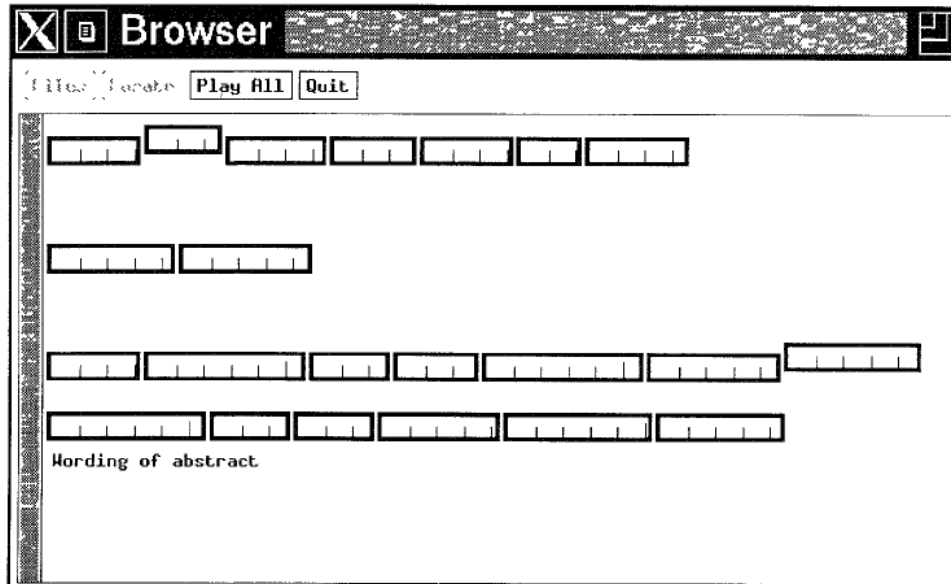


Fig. 3. Browsing a stored conversation with three groups of saved segments.

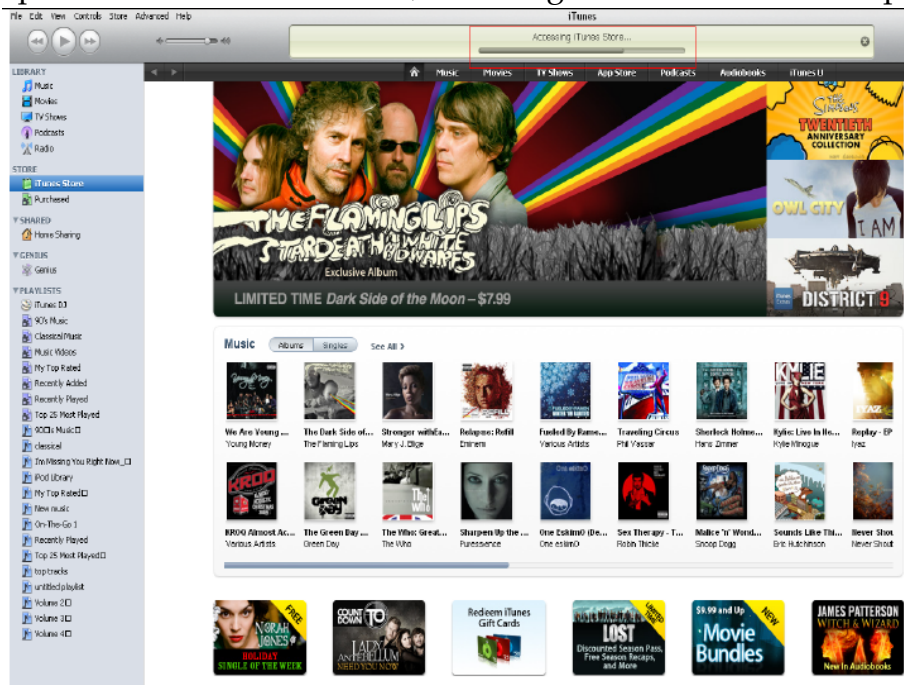
310. During prosecution of the patents, the Examiner rejected the claims in view of the BookManger READ program in the Cohen patent, which he found was a “browser application.” [858 PH Ex. 3 at PH\_001\_0000786973]:

Thus, at the time of the invention, one of ordinary skill in the art would consider the BookManager READ program of Cohen as a “browser application”. The NCSA Mosaic browser application, which presents hypermedia documents “that can include full color images and sounds” to users in “pages like an interactive, scrollable, on-line book”, include the same basic functions as the BookManager READ product, which also presents full color images and sounds within hypermedia documents, and manages, searches, and shows on-line books to users. Therefore, the BookManager READ product, taught by Cohen, can be considered as an equivalent to a browser application, as shown by the NCSA Mosaic reference, thus obtaining the invention as specified in claim 1.

311. Additionally, Mr. Ang, a named co-inventor, testified under oath in his deposition that “And we certainly described in the description itself distributed hypermedia environment. It’s not restricted to the web and HTML documents.” 7/21/11 Ang Tr. at 293:5-20; see also 8/11/11 Doyle Tr. at 443:18-446:5.

312. I also note that Mr. Martin, a named inventor, explained the prior art HyperCard browsing system as being like a web browser too. Martin Tr. at 268:5-10 (“Q. What is your understanding of the functionality of the HyperCard? A. I didn’t use it much. It displayed content on a screen, and you could click on highlighted words, I think, which would take you to another screen. Kind of like what a web browser does today.”)

313. Furthermore, Eolas’s Local Patent Rule infringement contentions against the Defendants in this case did not limit the application of its claims to web browsers either, as I noted in my opening report on invalidity with respect to Acrobat. For example, Eolas accused iTunes of infringing the patents too. See 985 – Apple – iTunes (Final).pdf (there is a chart for the ‘906 patent too). The chart states “Apple iTunes for Windows, Apple operating systems, the iPad and the iPhone runs in a computer network environment, including the internet. For example:



source: iTunes 9 (Windows)

314. Above, Eolas accused browsers that were not “web browsers” of infringement and did not limit its infringement analysis to the Web or the Internet for that matter---access to a network would be sufficient under those contentions.

315. At the time the patent was filed, and earlier, the ordinary meaning of the term “browser” was not “web browser.” It had a broader meaning at the time and was not limited simply to “web browsers.” It simply meant a computer application that was used to view or “browse” electronic documents. While I appreciate that today, in 2011, and as early as 2000, the term “browser” alone may be more commonly associated with a web browser, that was not the case in the 1993-1994 timeframe to a person of ordinary skill in the art. It was also not the case with respect to the prior art applied by the Patent Office during prosecution (e.g. the Cohen patent discussion at 4/18/2008 Office action at ¶ 22).

316. The Court also found that the terms “file” and “hypermedia document” need no construction. This relates to the ’985 patent claims. In my opinion, one of ordinary skill in the art would understand the plain and ordinary meaning of the language that “file” and “hypermedia document” in the claims are synonymous, or at least that the “file” is a portion of the “hypermedia document.” This is how I applied the term in my earlier report. However, even if they are not found to be synonymous or one is not part of the other, then there are two different files referred to in the claims. In this latter case, the file could be any hypermedia content or a hypermedia document that is used or assists in the presentation of the hypermedia document, such as a text file, an object (e.g. an image sequence or other multimedia), or a data file used as indicated in the claims.

### **3. The prosecution histories of the patents in suit**

317. I have reviewed the prosecution histories of the patents-in-suit, including that of the originally filed applications, and reexamination and interference proceedings. My observations are described below.

#### **a. The '906 Patent**

##### **i. The Original Patent Prosecution Of The '906 Patent**

318. I have reviewed the original prosecution of the application that would become the '906 patent. The application, No. 08/324,443 ("the '443 application"), was filed on October 17, 1994. The '443 application listed named inventors Michael D. Doyle, David C. Martin, and Cheong S. Ang.

#### **First Office Action**

319. The first Office Action in the original prosecution was issued on May 6, 1996. (PH\_001\_0000783863-878). In this office action, the Examiner rejected claims 1-4, and 15-26 in view of the "Mercury Project". Further, claims 1-43 were rejected in view of prior art provided by the applicant in combination with Hansen, "Enhancing documents with embedded programs: How Ness extends insets in the Andrew ToolKit." The Applicants' response to this office action was filed on August 6, 1996, amending claims 1-9, 14-15, 23-24, 28, and 34. The applicants cancelled claim 16, and added new claims 44-56.

320. With respect to the Examiner's rejection over The University of Southern California Mercury Project, the applicants stated that the rejection was traversed because the Mercury Project did not disclose "utilizing a browser to display a first hypermedia document in a first window with the hypermedia document including a tag format specifying the location of an external object and an external executable application." (PH\_001\_0000783893). The applicants stated that the Mercury Project instead "utilized CGI where a <FORM> tag identifies a program on the server but not an external object." (PH\_001\_0000783893). The applicants further state that, "unlike CGI, the claimed executable application does not generate a static HTML document to be displayed in place of the first document but displays and processes the object in a portion of the window." (PH\_001\_0000783894).

321. With respect to the Examiner's rejection over Hansen, the applications argued that the claims of the '443 were not obvious over the disclosed prior art in



view of Hansen because, "[t]here is no disclosure in the references, singly or in combination, of displaying a hypermedia document in a first window including a text format specifying the location of an external object and identifying an external executable application or of invoking the external application to display and process the external object within the first window." (PH\_001\_0000783895).

### Second Office Action

322. The second Office Action in the original prosecution of the '443 application was filed on December 13, 1996, rejecting claims 1 and 44 in view of Vetter, "Mosaic and the World-Wide Web" ("Mosaic"), further in view of Hansen, "Andrew as a Multiparadigm Environment for Visual Language." Additionally, claims 2-5, 10-14, 24-27, 45-48, and 55 were rejected in view of Mosaic, Hansen, and U.S. Patent No. 5,347,632 to Filepp. This Action was made final. The applicants' response to this Office Action was filed on January 8, 1997. (PH\_001\_0000783957-996). In addition to its remarks, attached to this response was a declaration by inventor Doyle wherein he swore behind the Mosaic reference. Essentially, in this declaration, inventor Doyle stated that the Mosaic reference was not prior art because the subject matter of the '443 application had been reduced to practice prior to October 1994, the asserted priority date for the Mosaic reference.

323. With respect to the Examiner's rejection in view of Mosaic, the Examiner had indicated that "Mosaic's functionality can be extended by having custom servers and letting other applications control its display remotely. Hence it would have been obvious to extend Mosaic's functionality to enable external application to display and process the object within the browser-controlled window because it would have improved the system by reducing cluttering of the display and aiding the reader comprehension of the hypermedia document." (PH\_001\_0000783958). In response to this rejection, applicants stated that, in view of

the Doyle Declaration swearing behind the Mosaic reference, "the feature of controlling Mosaic's display remotely is not disclosed in the prior art." (PH\_001\_0000783958). Therefore, the rejected claims were said to be allowable over Mosaic.

#### **Third Office Action**

324. The third Office Action in the original prosecution of the '443 application was filed on January 24, 1997. (PH\_001\_0000783997-4008). In this Office Action, the Examiner again rejected multiple claims of the '443 application. He rejected claims 1-5, 10-14, and 44-48 in view of Wynne, "Lean Management, Group Support Systems, and Hypermedia: A combination whose time has come") in combination with Hansen (as above). Following this Office Action, inventor Doyle and prosecuting attorney Charles Krueger attended an Interview with the Examiner on February 24, 1997. A summary of this Interview was filed by the Examiner on February 26, 1997, in which the Examiner indicated that the parties discussed the Wynne reference, and that he viewed the applicants' argument to overcome his rejection as "persuasive." (PH\_001\_0000784011).

#### **Fourth Office Action**

325. The fourth Office Action in the original prosecution of the '443 application was filed on March 26, 1997. (PH\_001\_0000784013-025). In this Office Action, the Examiner rejected claims 1 and 44 in view of U.S. Patent N. 5,206,951 to Khoyi. The applicants responded to this Office Action on June 2, 1997, amending claims 1-5, and 44-48. Additionally, the applicants cancelled 6-15, 17, 43, and 49-56. In addition to the applicants' remarks, this response also included another declaration from inventor Doyle. Unlike his prior declaration, here inventor Doyle's statements were intended to support the applicants' in favor of patentability.

326. The applicants' response argued that (1) the features recited in claim 1 of the application were not disclosed in the references cited by the Examiner, and (2) there is no suggestion in the prior art that would make the claimed invention obvious; the exercise of the invention would be required by one skilled in the art to make the claimed combination; and the commercial success of products incorporating claimed features establishes that the invention was not obvious. (PH\_001\_0000784040).

327. With respect to the rejection in view of Khoyi, the applicants stated that, unlike Khoyi, "in the claimed invention, the external object is displayed in a window in the document and interactively processed using the external executable application." (PH\_001\_0000784040).

328. With respect to the Examiner's rejections based on obviousness, the applicants states that the claimed invention "solves a different problem than each of the references," and "allows the hypermedia document to act as a coordinator and deployment mechanism as well as a container, for any arbitrary number of external *interactive* data/application objects, while hiding the details of such coordination and deployment from the document's reader as the reader uses the various data/application objects." (PH\_001\_0000784041) (emphasis in original).

329. Further, with respect to Mosaic, the applicants state, in attempting to differentiate Mosaic from the claimed application, that "Mosaic displays links, embedded in a first hypermedia document, and retrieves information identified by a link when a user activates the link. The retrieved information either replaces the first hypermedia document, *or is displayed in a separate window than the window displaying the first hypermedia document.*" (PH\_001\_0000784041-042) (emphasis added). In response to the Examiner's suggestion of combining Khoyi with Mosaic, the applicants further argue that "there is no suggestion in Khoyi of modifying Mosaic so that an external application, by analogy to Khoyi the source document

manager, is invoked to display and interactively process the object *within the document window while the document is displayed by Mosaic in the same window.*" (PH\_001\_0000784043) (emphasis added).

330. When the applicants stated that prior art Mosaic displayed content "in a separate window" after a user clicks on a link, one example of this functionality is that of "helper applications," which I discuss in this report. Examples of helper applications included MMPEG or XV. The manner in which Mosaic interoperated with the MMPEG helper application in the prior art is explained above in my report and is demonstrated in my accompanying video demonstration exhibits. (See Appendix C at Mosaic video 1.wmv.)

331. Additionally, beyond arguing that neither modification of Mosaic or Khoyi would have been obvious, and that both would have required novel, inventive, and non-obvious steps, the applicants further argued that the combination of Mosaic and Khoyi itself would "require a multiplicity of separate, novel, inventive, and awkward combinative steps that are too complicated to be considered obvious." (PH\_001\_0000784043).

#### **Fifth Office Action**

332. The fifth Office Action in the original prosecution of the '443 application was filed on August 25, 1997. (PH\_001\_0000784091-097). In this Office Action, the Examiner rejected 1, 2, 5, 44, 45, and 48 were rejected in view of prior art disclosed to the PTO by the applicants in combination with U.S. Patent No. 5,581,686 to Koppolu. Further, the Examiner rejected claims 3, 4, 46, and 47 in view of the combination of Koppolu and Moran, "Tele-Nicer-Dicer: A new tool for the visualization of large volumetric data." The applicants filed a response to this fifth Office Action on October 31, 1997. (PH\_001\_0000784099-124). This response included, in addition to the remarks of the applicants, yet another declaration by inventor Doyle. In this declaration, like in his first declaration, inventor Doyle swore behind the Koppolu

patent, and alleging that the invention of the '443 was reduced to practice prior to April 15, 1994, which I understand it the date of reduction to practice of the Koppolu reference.

333. On this basis of this Doyle Declaration, the applicants argued that the rejected claims were allowable over the Koppolu reference cited by the Examiner.

334. The Examiner's summary of a November 6, 1997 Interview indicates that all the claims and all the prior art were discussed at the Interview, and that the applicants argued that the prior art does not teach or disclose the feature of "automatic invoke of external application to provide interactive control." (PH\_001\_0000784099-125).

335. On December 23, 1997, applicants filed yet another 37 C.F.R. § 1.131 declaration signed by inventor Doyle, in which he reiterated his claim that the invention of the '443 application was reduced to practice prior to April 15, 1994. Additionally, the applicants offered arguments to overcome the Examiner's rejections in view of Mosaic, Koppolu, and OLE. In particular, with respect to Mosaic, the applicants note that "the Examiner has stated that Mosaic does not have an embed text format specifying an *external object which automatically invokes an external application to execute and enable interactive processing with a portion of the browser controlled window.*" (PH\_001\_0000784144) (emphasis added).

336. I have added the above emphasis because it is my opinion that both the Examiner and the applicants believe that phrase sums up the only differences between the invention and Mosaic.

#### Notice of Allowance

337. The PTO issued a Notice of Allowance for the '442 application on March 30, 1998. (PH\_001\_0000784167-172). The Examiner cited inventor Doyle's declarations, swearing behind the various prior art cited by the Examiner as among

his reasons for allowance. The Examiner further cited the *automatically invoking* step of the claims in the application as being a critical limitation that was lacking in the cited prior art. (PH\_001\_00007841698-169).

**ViolaWWW Was Not Mentioned Or Considered  
During the Original Prosecution Of The '906 Patent**

338. Based on my review of the prosecution history of the Original Prosecution of the '443 application which became the '906 patent, at no time during the original prosecution of the '906 patent did inventor Doyle or his representative, Mr. Krueger, disclose any information regarding the ViolaWWW browser to the PTO. They did not disclose either the Viola article published by Pei Wei on August 16, 1994, or the fact that Pei Wei, the ViolaWWW developer, informed Dr. Doyle of public demonstrations of ViolaWWW that occurred more than one year prior to the filing of the '906 patent application.

**ii. The Director-Ordered Reexamination Of The '906 Patent**

339. I reviewed the file history related to the first reexamination of the '906 patent. This reexamination was assigned No. 90/006,831 ("the '831 reexamination proceeding").

340. The PTO received a first prior art submission related to this reexam on October 14, 2003. (PH\_001\_0000784762-778). This submission disclosed the following prior art references: (1) Raggett, HTML+ (Hypertext Markup Language) (July 23, 1993) ("Raggett I"); (2) Posting of Dave Raggett, dsr@hplb.hpl.hp.com, to www-talk@nxoc01.cern.ch (June 14, 1993) ("Raggett II").

341. The PTO received a second prior art submission related to this reexam on October 24, 2003. (PH\_001\_0000784780-792). This submission also disclosed Raggett I and Raggett II as prior art references.

342. Following these submissions, the Director of the PTO ordered that the '906 patent be reexamined in view of the references that had been submitted.(PH\_001\_0000784818-826). I understand that reexaminations are ordered when submitted prior art presents a substantial new question of patentability. In this case the prior art that created this new question was Raggett I, Raggett II, and Berners-Lee, (Hypertext Markup Language (*HTML*), Internet Draft, IEFT, pp. 1-40 (June 1993)),

343. Subsequently, the applicants submitted and Information Disclosure Statement to the PTO on December 30, 2003, which included source code for the Viola browser. The source code was identified by what I understand were trial exhibit numbers from a prior litigation between Eolas and Microsoft. In addition to providing the Viola sourcecode, the applicants suggested to the PTO that it was not prior art. I understand that this suggestion could have led the Examiner to believe he did not have to consider the Viola sourcecode in connection with the reexamination proceeding.

#### **First Office Action**

344. The PTO filed the first Office Action of this first reexamination on February 26, 2004. (PH\_001\_0000785292-303). In this first Office Action, the Examiner rejected claims 1-3 and 6-8 in view of the Berners-Lee, Raggett I, and Raggett II references mentioned above. The Examiner also stated that the Viola sourcecode ("DX34" and "DX37") that had been provided by the applicants had not been considered in preparing this first Office Action. (PH\_001\_0000785294).

345. An interview between the Examiner and the patentee on April 27, 2004, (PH\_001\_0000785316-339). At the Interview, a PowerPoint summary of the invention was presented to the Examiner, and prior art was discussed. The Viola browser was not discussed during the Interview, and was not included in the PowerPoint presentation. (PH\_001\_0000785316-339); (Krueger Tr. 70:13 - 76:10).

346. Subsequently, the patentee filed a response to the first office action on May 10, 2004. (PH\_001\_0000785359-379). Included with this office action were declarations by Dr. Doyle, as well as Prof. Edward Felten, and Charles Krueger. The Doyle declaration stated "facts relating to reactions by experts in the field at the time the technology recited in claims 1 and 6 of the '906 patent" was introduced. The Felten Declaration "travers[ed] the rejections of claims 1 and 6 of [the '906 patent]." The Krueger Declaration "set[] forth testimony from the *Eolas v. Microsoft* trial and other exhibits." The response also contained numerous arguments in favor of patentability.

347. With respect to Raggett I and II, the arguments against the Examiner's rejections included the following: (a) Raggett I and II do not disclose interactivity with the image displayed in the browser window because the external rendering application of Raggett I and II would cease execution when it returned a static image to the browser; (b) Raggett's proposed EMBED tag within the FIG tag requires that Raggett's proposed EMBED tag return a static, and non-interactive image; (c) Raggett I teaches away from the claimed element of automatically invoking an executable application on the client workstation in order to display the object and to enable in-place interaction.

348. The Viola reference was not discussed by the patent owners in the response to the first Office Action.

### **Second Office Action**

349. The second Office Action related to the first reexamination proceeding was filed by the Examiner on August 16, 2004. In this Office Action, the Examiner rejected claims 1-10 in view of Berners-Lee, Raggett I, and Raggett II, as well as Toye (SHARE : A Methodology and Environment for Collaborative Product Development, Proceedings, Second Workshop on Enabling Technologies: Infrastructure for



Collaborative Enterprises, 1993, IEEE, pp. 33-47, April 22, 1993). (PH\_001\_0000785553-571). In response to that Office Action, the patentee submitted Remarks on October 12, 2004. (PH\_001\_0000785803-832). The Application included Declarations by Prof. Edward Felten (as well as Prof. Felten's previously submitted May 10, 2004 Declaration), and Robert Dolan, the Dean of the University of Michigan Business School.

350. With respect to the Examiner's rejections, the patent owners argued that "[t]here is no suggestion or teaching in either Toye, the admitted prior art (Mosaic), Berners-Lee, Raggett I or Raggett II of *automatically invoking an external application to execute on a client computer*, when an embed text format is parsed, to display and interactively control an object in a display window in a hypermedia document, received over a network from a network server, being displayed in a browser-controlled window on the client computer." (PH\_001\_0000785804) (emphasis added).

351. Further, the patent owners stated that "[t]here is no suggestion or teaching in either Toye, the admitted prior art (Mosaic), Berners-Lee, Raggett I or Raggett II of parsing an embed text format at a first location in the hypermedia document and displaying the object and enabling *interactive processing of the object within a display area created at the first location within the portion of the hypermedia document being displayed*." (PH\_001\_0000785804) (emphasis added).

352. The Viola reference was not discussed by the patent owners in the response to the second Office Action.

353. On August 18, 2005, another interview occurred between the patentee and the Examiner. (PH\_001\_0000785902-904). In this interview, it was communicated for the first time by the Examiner that the Viola reference was being considered. However, I understand that the reference was not, at this time, considered by the Examiner, but was rather considered by the Office of Patent Legal Administration as

to whether the Examiner should consider it in relation to the reexamination proceeding. (PH\_001\_0000785902).

354. No additional mention was made of the Viola reference in this communication to the PTO.

#### Notice Of Intent To Issue A Reexam Certificate

355. The PTO issued a Notice of Intent to Issue a Reexamination Certificate on September 27, 2005. (PH\_001\_0000785905-981). In this Notice, the Examiner indicated that the "DX37" code had been considered a publication by the PTO that constitutes prior art (PH\_001\_0000785953) and gave rationale for the allowance of the claims over the "DX37" code (PH\_001\_0000785953-969).

356. It is evident from the description given by the Examiner in his Notice that he "text searched" the files contained in "DX37", and reviewed them in accordance with the "hits" produced from that search. (PH\_001\_0000785954). For this search the Examiner stated he used a commercial product called *dtSearch*, which is described on the web site [www.dtsearch.com](http://www.dtsearch.com). I have visited that site and explored various searching demos described there. In general, the search strategies can be: any or all of a group of words, a structured Boolean group of words or phrases, and exact phrases. While the Examiner describes getting "hits" to his searches, he fails to disclose what terms he used in his search. Since many of the asserted limitations of the '906 and '985 patents focus on data being plotted in embedded windows, any of the preceding underlined words would seem to be likely search terms. In fact, a quick search shows that there are 90 occurrences of variations of the word "embed" in the '906 patent.

357. Thus, using a Windows application called *Search and Replace*, I performed a regular expression search on all 1030 files in the "DX37" codebase using the term \*embed\* and found 15 occurrences in 12 files, which is not very much data

to track down. One occurrence is in the file testPlot.html, which leads to this very revealing information:

```
<TITLE>XPlot</TITLE>
```

```
<H1>An equation and vector-objects plotting program</H1>
```

```
<P>
```

```
The <CMD>xplot</CMD> program was written by Scott Silvey at the  
Experimental Computing Facility of UCB. In an experiment,
```

```
<CMD>xplot</CMD> was made accessible by other program, such as viola.
```

```
<P>
```

```
Here's a viola front-end to <CMD>xplot</CMD>, which is embedded  
into <ITALIC>this</ITALIC> document.
```

```
<VOBJF>/usr/work/viola/apps/plot.v</VOBJF>
```

```
<EXAMPLE>
```

```
file /usr/work/vplot/off/x29.geom
```

```
equation .4 * sin(.4 * x + y)
```

```
equation .4 * sin(1 - x*x - y*y)
```

```
equation .4 * exp(1 - x*x - y*y) * sin(1 - x*x - y*y)
```

```
equation .4 * exp(1 - x*x - y*y) * sin(1 - x^4 - y^4)
```

```
</EXAMPLE>
```

358. From this one gleans that there is a program called "xplot," which is "An equation and vector-objects plotting program and was made accessible by other program, such as viola." Furthermore, it shows "Here's a viola front-end to <CMD>xplot</CMD>, which is embedded into <ITALIC>this</ITALIC> document" and points out the role of the Viola application "plot.v" and refers to a file path "file /usr/work/vplot/off/x29.geom." A further examination of plot.v shows additional code that pertains to plotted output.

359. As discussed further herein, it is evident from the Examiner's description of his review of "DX37" that he reviewed non-material information ("clock.v") and did not review the material information ("testPlot.hmml" and "plot.v") contained on the CD. From the Examiner's description of his review, it appears the review was focused solely on a Viola script entitled "clock.v". According to the Examiner, "clock.v" is "interpreted" to embed an interactive application object within the same window of the Viola browser. (PH\_001\_0000785955). As discussed further herein, "clock.v" is a script, and is not a compiled executable application as can be found elsewhere in the reference. It is therefore my opinion (as also discussed below) that the Examiner's search and review of this reference was not "full and comprehensive." Indeed, it is unlikely that the examiner would have used the term "clock" in his search, as it does not appear in or otherwise relate to the patent.

360. The Examiner provided the following summary of his review of "DX37":

While Viola DX37 supports hypermedia and a type of interpreted script-based interactive processing, the Examiner can find no indication from a comprehensive text search of the DX37 files that such interactivity results from the use of a parsed *embed text format* that specifies the location of an *object* external to the hypermedia document, where the browser application *uses type information associated with the object to identify and locate an external executable application*, and where the parsing step results in the browser automatically invoking the *executable application* to display the *object* and enable interactive processing of the *object* within the same browser-controlled window, when the instant '906 patent claims 1 and 6 are properly accorded the broadest reasonable interpretation consistent with the specification. (PH\_001\_0000785955) (emphasis in original).

361. It is further evident from the Examiner's Notice, that the files "testPlot.hmml" and "plot.v", which was also contained within "DX37", was not reviewed or considered by the Examiner in conjunction with his review of the Viola prior art.

**"testPlot.hmml" and "plot.v" Were Not Considered  
By The Examiner, Or Identified To The Examiner  
By The Patent Owners During The First  
Reexamination Proceeding Of The '906 Patent**

362. In my review of the '831 Reexamination Proceeding, I found nothing that would suggest that "testPlot.hmml" and "plot.v" were specifically disclosed to the Examiner, or that it was otherwise considered by the Examiner. The document "testPlot.hmml" invokes "plot.v" which in turn invokes vplot, an executable application. The file "clock.v", however, is merely a script. Therefore, in my opinion, the reasoning offered by the Examiner for allowance of the '906 patent over the ViolaWWW reference would not have applied had he followed the obvious search and analysis path I described above.

363. It is my opinion that the "DX37" codebase was never sufficiently examined, partly because the patent owners failed to provide the Examiner with tools for analyzing it, compiling it or identifying the relevant portions.

**iii. The Second Reexamination Proceeding**

364. I also reviewed the prosecution history for the second reexamination proceeding. This proceeding was assigned No. 90/007.858 ("the '858 Reexamination Proceeding").

365. The PTO received a request for additional reexamination of the '906 patent on December 22, 2005. (PH\_001\_0000786091-133). This request cited Janssen (a public posting to the www-talk email list), which I again understand to have been thought to create a substantial new question of patentability. Among the reasons for this claim were that the Janssen reference filled in the gaps in the Raggett II reference that had been previously identified by the examiner, and refutes positions taken by the patent owner with respect to the Raggett II reference. The requester also argued that "the Janssen posting does not merely teach the combination with Raggett II that

the Examiner found lacking; it effects and constitutes that combination." (PH\_001\_0000786095). The requester further identified another piece of withheld prior art. This was a reference authored by Doyle and Ang, two of the named inventors in the '906 patent, entitled "Visible Embryo Project." The publication "describes the project's application software as being distributed between a client workstation and another computer connected over a network, with ongoing communication between the client workstation and the network computer as recited in at least claims 4, 5, 9, and 10 of the '906 patent." (PH\_001\_0000786095).

366. The request for reexamination was granted by the PTO on February 9, 2006. (PH\_001\_0000786255-266). The assigned Examiner agreed that a substantial new question of patentability was raised by the Janssen reference in combination with the other prior art that had previously been reviewed.

367. Between February 9, 2006, and July 30, 2007, the patent owners submitted numerous Information Disclosure Statements to the PTO. Among these was an IDS, filed on November 1, 2006 that included information regarding the ViolaWWW reference. [858-PH] at PH\_001\_0000786301 - 307]. The patent owners stated that "DX34" and "DX37" had not previously been submitted in their entirety, and were, finally, submitted in their entirety along with the November 1, 2006 IDS. The information submitted with the IDS did not encourage the Examiner to review the "testPlot.hmm1" document or the "plot.v" application as part of the '858 Reexamination Proceeding.

368. Once again, I point out that the patent owners failed to provide the Examiner with tools for analyzing, compiling or identifying the relevant portions of either "DX34" or "DX37".

#### **First Office Action**

369. The first Office Action related to this second reexamination was filed by the Examiner on July 30, 2007. (PH\_001\_0000786943-982). The Examiner disagreed

that the combination of Janssen with Raggett II and the remaining prior art anticipated the claims of the '906 patent. Specifically, the Examiner notes that he "can find no evidence within the references of Janssen and Raggett that specifically teach of 'automatically invoking the executable application to execute on said client workstation in order to display said object.'" (PH\_001\_0000786952).

370. I note that, at the time of the '858 Reexamination Proceeding, the Examiner did not have the benefit of reviewing testimony of Mr. Janssen that has been given in connection with this litigation. I have reviewed this testimony, and have considered it in forming my opinions offered in this report.

371. In this Office Action, Claims 1-10 were rejected in view of Pei Wei's "Viola Article" ("A Brief Overview of the Viola Engine, and its applications"). Additionally, Claims 1-3 and 6-8 were rejected in view of U.S. Patent No. 5,367,621 to Cohen. The Examiner stated that "the secondary reference of NCSA Mosaic is being utilized to show that the BookManager READ product of the primary reference of Cohen can be considered as a 'browser application', therein proving that the Cohen reference has an 'enabled disclosure.'" (PH\_001\_0000786969).

372. An interview between the patentee and the Examiner took place on September 6, 2007. (PH\_001\_0000787018-021). In that interview the Viola and Cohen prior art were discussed, along with the NCSA Mosaic reference. With respect to the Viola reference, the Examiner's notes indicate that the patent owners argued that '906 patent was reduced to practice before the publication date of the Viola article, and that the Viola article was therefore not prior art to the '906 patent. With respect to the Cohen reference, the Examiner's notes indicate that the patent owners stated that "Cohen does not teach of the specific claimed features of: interactive processing, an embed text format, a display within the browser window, and type information used by the browser to identify and locate." (PH\_001\_0000787021). The Examiner agreed to review and reconsider the rejections in view of these references.

373. The patentee filed a response to the July 30, 2007 Office Action on September 27, 2007. (PH\_001\_0000787028-051). The response included another declaration by inventor Doyle in which he again swore behind the Viola reference. Again, I understand that this means that inventor Doyle claimed that the invention of the '906 patent had been reduced to practice before the Viola reference. (PH\_001\_0000787070-191). The response also included an additional Declaration of Prof. Edward Felten, rebutting the arguments made by the Examiner in the first Office Action in connection with the '858 Reexamination. (PH\_001\_0000787052-069).

374. Beyond the Doyle Declaration, the response provided no substantive response to the Examiner's rejections in view of the Viola reference. There is no suggestion in the response that the Examiner undertake and further review of the Viola reference, including "DX34" or "DX37".

375. With respect to the Cohen reference, the patent owners argued that the "interactive processing" claimed in the '906 patent was not explicitly found in Cohen, nor inherent in Cohen, nor enabled by Cohen. Further, they argued that the "embed text format" claimed in the '906 patent is not explicitly found in Cohen, nor inherent in Cohen. Finally, they argued that the "display area" claimed in the '906 patent is not explicitly found in Cohen, nor inherent in Cohen.

### **Second Office Action**

376. A second Office Action related to the second reexamination proceeding was filed by the Examiner on April 18, 2008. (PH\_001\_0000787208-244). In the Office Action, the Examiner noted his acceptance of the assertions made in inventor Doyle's declaration. There is no indication that the Examiner undertook any further review of the Viola reference, "DX34" or "DX37" in withdrawing this objection. The Examiner let stand the previous rejections of the '906 patent claims (1-3 and 6-8) in view of Cohen, stating that the arguments of the patent owners had not been persuasive. The



Examiner's argument was, essentially, that the patent owners had interpreted Cohen too narrowly in their response to the previous office action, and in a manner that was not in compliance with the rules of the Patent Office with respect to claim construction during patent examination proceedings.

377. An additional pair of interviews between the Examiner and the patentee took place on May 9, 2008 and June 3, 2008. (PH\_001\_0000787254-256). At the interview, an amendment proposed by the patent owner was discussed. The Examiner appears to have believed that the proposed amendment appeared sufficient to overcome the rejections over the Cohen reference because, "as upon initial review, Cohen is unclear if an end-user directly interacts with the object, rather than the 'interactive processing' as claimed." (PH\_001\_0000787256).

378. The Amendment discussed above was filed with the PTO on June 23, 2008. (PH\_001\_0000787257-273), amending several claims of the '906 patent. Notably, independent claims 1 and 6 were amended to "include the language that said executable application enables 'an end-user to directly interact with said object' in order to obviate the ground of rebuttal relating to interactive processing asserted by the examiner". (PH\_001\_0000787271).

#### **Notice Of Intent To Issue A Reexam Certificate**

379. On September 10, 2008, in reliance on the June 23, 2008 Amendment, the PTO filed a Notice of Intent to Issue a Reexamination Certificate related to this second reexamination on September 10, 2008. (PH\_001\_0000787394-407). In his reasons for allowance of the '906 patent, the Examiner stated that "Cohen does not specifically disclose the feature of allowing an end-user to directly interact with the object within the display area of the browser after the object is automatically invoked."

380. Thus, it is my understanding that the '906 patent claims were only narrowly allowed over Cohen in view of the Amendment which added the limitation specifying that the '906 patent enabled the end-user to interact directly with the object inside the browser window. The original language of claims 1 and 6 allowed "interactive processing of," a phrase accepted by the Examiner in his earlier rejection.

381. Moreover, it is my understanding that the '906 patent claims were allowed over the Viola reference only in view of the Doyle Declaration swearing behind that reference.

**No Further Review Of "DX34" or "DX37" Was Undertaken During The Second Reexamination Proceeding, Nor Was The "testPlot.hmml" Document Nor The "plot.v" File Considered**

382. Based on my review of the file history of the '858 Reexamination, I found nothing that would suggest that any further examination of "DX34" or "DX37" or other Viola code occurred. Therefore, as with the First Reexamination proceeding, neither the "testPlot.hmml" document nor the application "plot.v" was considered by the Examiner, nor was any portion of "DX34". Instead, the patent owners, including Dr. Doyle effectively circumvented any further examination of the substance of the Viola prior art through declarations and assertions suggesting alternatively that (1) "DX34" and "DX37" were not appropriate for consideration in a reexamination proceeding; and (2) that the Viola article was not prior art because the alleged invention of the '906 patent was reduced to practice before the publication date of the Viola article.

383. In my opinion, neither one of these assertions is true, for reasons described elsewhere in this report.

**iv. The "Koppolu" Interference**

384. I understand that the '906 patent was also involved in an interference proceeding with the Koppolu application (09/442,070) ("the Koppolu Application"). I understand that, with respect to this interference, Dr. Doyle, and co-inventors David Martin and Cheong Ang, representing the '906 patent, were the "junior party". (PH\_001\_0000787690). I further understand that Srinivasa Koppolu and co-inventors C. Douglas Hodges, Barry MacKichan, Richard McDaniel, Rao Remala, and Anthony Williams, representing the Koppolu Application were the "senior party." (PH\_001\_0000787690).

385. I understand that, with respect to patent interference proceedings, the patent or application belonging to the "junior party" is presumptively invalid because it claims the same subject matter previously invented by the "senior party."

386. I understand that a decision on the merits was not reached in this Interference proceeding. Instead, I understand that the parties reached a settlement in advance of such judgment on the merits. (PH\_001\_0000787683-685).

**b. The '985 Patent****i. The Original Patent Prosecution Of The '985 Patent**

387. I have reviewed the original prosecution of the application that would become the '985 patent.

388. The application, No. 10/217,955 ("the '443 application"), was filed on August 9, 2002. The '955 application listed named inventors Michael D. Doyle, David C. Martin, and Cheong S. Ang. The '955 application was a continuation of application no. 08/324,443 filed on October 17, 1994, and which had previously issued as the '906 patent (I note that there appears to have been a previous continuation of the '442 application as well).

### First Office Action

389. The first Office Action in the original prosecution of the '955 application was filed by the Examiner on July 20, 2004. (PH\_001\_0000784201-212). Claims 1-3 were rejected in view of the prior art that had previously been considered in Berners-Lee, Raggett I, Raggett II, and Toye. I note that this Office Action appears to have been filed during the pendency of the first reexamination proceeding of the '906 patent, and that the above references were also a basis for provisional rejection of the '906 patent at this time.

390. The applicants submitted their response to this first Office Action on March 11, 2005. (PH\_001\_0000784213-244). The response included additional declarations by Dr. Edward Felten. These were the same declarations submitted in conjunction with the '906 Reexamination. (PH\_001\_0000784283-284).

391. There is no mention of the Viola prior art in the March 11, 2005 response to the first Office Action.

392. The applicants filed an Information Disclosure Statement on October 24, 2005, including a CD containing Viola sourcecode.

393. Thereafter, the PTO notified the applicants that prosecution of the '955 application would be suspended for a period of 6 months. I understand that this suspension was ordered to allow the reexamination of the '906 patent to proceed, as the outcome of that reexamination would likely also be dispositive of the outcome of the '955 application. (PH\_001\_0000784285-286).

394. Several additional suspensions of the prosecution of the '955 application were subsequently filed.

395. On February 5, 2009, the applicants filed an additional amendment to the application. (PH\_001\_0000784613-697).

### Notice of Allowance

396. The PTO filed a Notice of Allowability of the '995 application on March 20, 2009. (PH\_001\_0000784728-734). The Examiner indicated that the reasons for allowance of the '995 application were essentially that the specification and subject matter were identical to that of the '906 patent, and that therefore the '955 application was allowable for the same reasons as the '906 patent. (PH\_001\_0000784733).

Thereafter, the '985 patent issued on October 6, 2009.

397. There is no indication in the Notice of Allowance that the Examiner gave any further consideration to the Viola reference or any of the other prior art cited or reviewed during the prosecution or reexaminations of the '906 patent.

398. Specifically, there is no indication that "DX34" or "DX37" were reviewed in connection with the prosecution of the '985 patent, and no indication that the Examiner reviewed the "testPlot.hmml" document or the "plot.v" application during prosecution of the '985 patent.

399. It is therefore my opinion that the PTO never fully considered "DX34" or "DX37" with reference to the prosecution of the '906 or '985 patents, and has not specifically considered the "testPlot.hmml" document or the "plot.v" example of the Viola prior art at any time with reference to the '906 or '985 patents or the applications from which these patents matured.

## VIII. Legal Standards

400. I am not an attorney, and I will offer no opinions on the law. I am, however, informed on several principles concerning patent validity, which I have used in arriving at my conclusions in this declaration.

401. First, for a patent to be valid, the invention claimed in the patent must be new, useful, and non-obvious in light of what came before it. That which came before is generally referred to as "prior art," which will be explained in detail below. In addition, for a patent to be valid, it must also satisfy various requirements relating

to the form of the patent. For example, the patent text must adequately describe the invention, it must sufficiently teach those of skill in the art to make and use the invention, and it must provide the "best mode" for practicing the invention. These various validity requirements will be explained in more detail below.

402. I understand that the burden is on the accused infringer to prove invalidity by "clear and convincing" evidence, which is a higher standard than a "preponderance" of the evidence, but a lower standard than "beyond a reasonable doubt."

403. I understand that a validity analysis is essentially a two-step process. First, the claims of the patent are construed to resolve any disputes as to their meaning and scope. Claims define the invention. I understand that claim construction is for the Court. Second, after the claims are construed, the content of the prior art is compared to the properly construed claims. To the extent a term has not been construed and I have not provided an explicit construction, I have used the meaning the term would have as usually and customarily understood by a person of ordinary skill in the art.

404. Prior art is generally the state of technology in the relevant field at the time of the invention, including both systems described in publications, such as conference papers, and systems actually in use at some time prior to the patent filings. I understand that 35 U.S.C. § 102 states that a person shall be entitled to a patent unless:

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for patent, or

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of the application for patent in the United States, or

(c) he has abandoned the invention, or

(d) the invention was first patented or caused to be patented, or was the subject of an inventor's certificate, by the applicant or his legal representatives or assigns in a foreign country prior to the date of the application for patent in this country on an application for patent or inventor's certificate filed more than twelve months before the filing of the application in the United States, or

(e) the invention was described in — (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for the purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language; or

(f) he did not himself invent the subject matter sought to be patented, or

(g) . . . (2) before such person's invention thereof, the invention was made in this country by another inventor who had not abandoned, suppressed, or concealed it. In determining priority of invention under this subsection, there shall be considered not only the respective dates of conception and reduction to practice of the invention, but also the reasonable diligence of one who was first to conceive and last to reduce to practice, from a time prior to conception by the other.

405. I understand that there are two ways in which prior art can be used to invalidate a patent. First, the prior art can be shown to "anticipate" the claim. Second, the prior art can be shown to "render obvious" the claim. My understanding of the two legal standards is set forth below.

### **1. Anticipation**

406. I understand that the following standards govern the determination of whether a claim in a patent is invalid as "anticipated." If an invention is not new, it has been "anticipated" by the prior art. I have applied these standards in my evaluation of whether the claims asserted in this case are anticipated.

407. In general, for a patent claim to be invalid as "anticipated" by the prior art, each and every feature of the claim must be found, expressly or inherently, in a

single prior art reference or product arranged as in the claim. Claim limitations that are not expressly found in a prior art reference are inherent if the prior art necessarily functions in accordance with, or includes, the claim limitations. It is acceptable to examine evidence outside the prior art reference (extrinsic evidence) in determining whether a feature, while not expressly discussed in the reference, is necessarily present within that reference.

408. In general, the standard for anticipation is the mirror image of the standard for infringement. Both require a comparison of the limitations of the claim to the elements of a prior art reference (for anticipation) or an accused device (for infringement). If each and every limitation of the claim can be found in the prior art reference, then the claim is anticipated (regardless of whether the prior art reference includes additional features). Similarly, if each and every limitation of the claim can be found in the accused device, the claim is infringed (regardless of whether the accused device includes additional features).

409. I understand that there are a number of different ways that anticipation can occur.

410. First, if the claimed invention was "known or used by others" in this country before the asserted date of invention, then the claim is anticipated. A demonstration or oral presentation could suffice; printed publications are not required.

411. Second, if the claimed invention was "in public use" in this country more than one year prior to the date of the application for the patent in the United States, then the claim is anticipated. Public knowledge of the invention or an enabling disclosure is not required; only public use is required.

412. Third, if the claimed invention was "described in a printed publication" anywhere in the world more than one year prior to the date of the application for the patent in the United States, then the claim is anticipated. To anticipate, however, the



printed publication must also enable one skilled in the art to make and use the claimed invention.

413. Fourth, if the claimed invention was made in this country by another inventor before the asserted date of invention, and was not abandoned, suppressed, or concealed, then the claim is anticipated. It is normally the first inventor to conceive, rather than the first to reduce to practice, who is entitled to priority, assuming that the first to conceive was reasonably diligent in reducing the invention to practice from a time prior to conception by the other.

## 2. Obviousness

414. I understand that an inventor is not entitled to a patent if his or her invention would have been obvious to a person of ordinary skill in the field of the invention at the time the invention was made. I understand that 35 U.S.C. § 103(a) states:

A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

415. The following standards govern the determination of whether a claim in a patent is obvious. I have applied these standards in my evaluation of whether the claims asserted in this case are obvious.

416. A claim in a patent is obvious when the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains.

417. I understand that obviousness may be shown by considering more than one item of prior art. I also understand that the relevant inquiry into obviousness

requires consideration of four factors (although not necessarily in the following order):

- (1) The scope and content of the prior art;
- (2) The differences between the prior art and the claims at issue;
- (3) The level of ordinary skill in the pertinent art; and
- (4) Whatever objective factors indicating obviousness or non-obviousness may be present in any particular case.

418. In addition, I understand that the obviousness inquiry should not be done in hindsight, but should be done through the eyes of a person of ordinary skill in the relevant art at the time the patent was filed.

419. I understand the objective factors indicating obviousness or nonobviousness may include: commercial success of products covered by the patent claims; a long-felt need for the invention; failed attempts by others to make the invention; copying of the invention by others in the field; unexpected results achieved by the invention; praise of the invention by the infringer or others in the field; the taking of licenses under the patent by others; expressions of surprise by experts and those skilled in the art at the making of the invention; and the patentee proceeded contrary to the accepted wisdom of the prior art.

420. I understand that the combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results. I also understand that when a work is available in one field of endeavor, design incentives and other market forces can prompt variations of that work, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, that variation is not patentable. I understand that for the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill.

421. In my understanding, the obviousness analysis need not seek out precise teachings directed to the specific subject matter of the challenged claim, for a court can take account of the "ordinary innovation" that does no more than yield predictable results, which are inferences and creative steps that a person of ordinary skill in the art would employ.

422. I understand that sometimes it will be necessary for a court to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in the art. I understand that all these issues may be considered to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue.

423. I understand that the obviousness analysis cannot be confined by a formalistic conception of the words "teaching, suggestion, and motivation." I understand that in 2007, the Supreme Court issued its decision in *KSR Int'l Co. v. Teleflex, Inc.* where the Court rejected the previous requirement of a "teaching, suggestion, or motivation to combine" known elements of prior art for purposes of an obviousness analysis as a precondition for finding obviousness. It is my understanding that *KSR* confirms that any motivation that would have been known to a person of skill in the art, including common sense, or derived from the nature of the problem to be solved, is sufficient to explain why references would have been combined.

424. A person of ordinary skill attempting to solve a problem will not be led only to those elements of prior art designed to solve the same problem. I understand that under the *KSR* standard, common sense is important and should be considered. Common sense teaches that familiar items may have obvious uses beyond their primary purposes, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle. As such, the prior art

considered can be directed to any need or problem known in the field of endeavor at the time of the invention and can provide a reason for combining the elements of the prior art in the manner claimed. In other words, the prior art does not need to be directed towards solving the same specific problem that is addressed in the patent. Further, the individual prior art references themselves need not all be directed towards solving the same problem.

425. It is my understanding that an invention that might be considered an obvious variation on, or modification of, the prior art may be considered non-obvious if one or more prior art references discourage or lead away from the line of inquiry disclosed in the reference(s). A reference does not "teach away" from an invention simply because the reference suggests that another embodiment of the invention is optimal or preferred. My understanding of the doctrine of teaching away requires some clear discouragement of that combination in the prior art — such as expressly stated reasons why one should not make the claimed combination or invention.

426. I understand that a person of ordinary skill is also a person of ordinary creativity.

427. I further understand that in many fields, it may be that there is little discussion of obvious techniques or combination, and it often may be the case that market demand, rather than scientific literature or knowledge, will drive design trends. When there is such a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense. In that instance the fact that a combination was obvious to try might show that it was obvious. The fact that a particular combination of prior art elements was "obvious to try" may indicate that the combination was

obvious even if no one attempted the combination. If the combination was obvious to try (regardless of whether it was actually tried) or leads to anticipated success, then it is likely the result of ordinary skill and common sense rather than innovation.

### 3. Section 112 Requirements

428. I understand that 35 U.S.C. § 112 states:

[¶ 1] The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same, and shall set forth the best mode contemplated by the inventor of carrying out his invention.

[¶ 2] The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

[¶ 3] A claim may be written in independent or, if the nature of the case admits, in dependent or multiple dependent form.

[¶ 4] Subject to the following paragraph, a claim in dependent form shall contain a reference to a claim previously set forth and then specify a further limitation of the subject matter claimed. A claim in dependent form shall be construed to incorporate by reference all the limitations of the claim to which it refers.

[¶ 5] A claim in multiple dependent form shall contain a reference, in the alternative only, to more than one claim previously set forth and then specify a further limitation of the subject matter claimed. A multiple dependent claim shall not serve as a basis for any other multiple dependent claim. A multiple dependent claim shall be construed to incorporate by reference all the limitations of the particular claim in relation to which it is being considered.

[¶ 6] An element in a claim for a combination may be expressed as a means or step for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.

429. I understand that the invention claimed in the patent must be adequately described. In exchange for the right to exclude others from making, using, selling or offering for sale the claimed invention, the patent owner must

provide the public with a complete description of the invention in the patent and how to make and use the claimed invention. This is called the "written description" requirement. One of the key purposes of this requirement is to make sure that the inventor actually invented the invention that is claimed in the patent. For a claim to be valid, a person of ordinary skill in the art must be able to determine from the patent application as originally filed that the inventor actually had "possession" of the invention.

430. I understand that the patent specification must enable those skilled in the art to make and use the full scope of the claimed invention without undue experimentation. This is to ensure that the public knowledge is enriched by the patent specification to a degree at least commensurate with the scope of the claims. I understand that the relevant time frame for measuring enablement is the date of filing the application.

#### **4. Level of Ordinary Skill in the Art**

431. I understand and have been instructed that the factors that may be considered in determining the ordinary level of skill in the art include: (1) the levels of education and experience of persons working in the field; (2) the types of problems encountered in the field; and (3) the sophistication of the technology. I understand and have been instructed that a person of ordinary skill in the art is not a specific real individual, but rather a hypothetical individual having the qualities reflected by the factors discussed above.

432. I have considered the levels of education and experience of persons working in the field, based on my review of prior art materials and based on my personal experience having been a member of this field when the patents-in-suit were filed.

433. I have also considered the types of problems encountered in the field, and in particular have considered which types of problems were thought to be

routine problems that were solved as an ordinary part of any development project, as well as types of problems that would have been considered more substantial and whose solutions would have been considered inventive.

434. I have also considered the sophistication of the technology, including the level of sophistication of the work of the named inventors in creating their preferred embodiment.

435. Based on my experience and my consideration of the factors above, the level of skill in the art in 1994, which is represented by the '906 and '985 patents, is that of a Bachelor's degree in computer science or computer engineering, or a Bachelor's degree in any engineering field with additional education or relevant experience in, for example, web browsers and Internet technology. After teaching computer science and computer graphics to thousands of students and directing and participating in dozens of relevant research projects, I feel very qualified to judge the required level of skill in the art relative to the '906 and '985 patents.

## 5. Materiality

436. I understand that an applicant has a duty of candor and good faith in dealing with the PTO. This duty includes a duty of disclosure, which requires applicants and their representatives to provide information known to be material to patentability of the claims pending in the application.

437. I understand that material information can include any type of information that would be considered important in deciding whether to allow the claims of an application. Such information includes but is not limited to published prior art, prior public uses, sales or offers for sale, invention by another, or any other information that would impact the PTO's decision to issue the patent.

438. It is also my understanding that for purposes of reexamination, a *cumulative* reference is one that substantially reiterates verbatim the teachings of a

reference that was either previously relied upon or discussed in a prior PTO proceeding even though the title or the citation of the reference may be different.

439. I understand that the materiality of withheld prior art is determined using a "but-for" standard: when an applicant fails to disclose prior art to the PTO, that prior art is but-for material if the PTO would not have allowed a claim had it been aware of the undisclosed prior art. That determination is made while giving claims their broadest reasonable construction.

440. I understand that one exception to the "but-for" standard of materiality is cases of affirmative acts of egregious misconduct, such as the filing of an unmistakably false affidavit. Affirmative acts of egregious misconduct before the PTO (such as dishonesty) are always material, regardless of whether the PTO would have allowed the claim in the absence of such conduct.

**IX. My Opinions Relating To This Case**

**1. Viola**

**a. Analyzed source code**

441. In forming my opinions about Viola, I reviewed several Viola codebases from the disc PA-NAT-78. These codebases are cited by filepath within PA-NAT-78 in this report. In addition, I confirmed that codebases within PA-NAT-78 were equivalent to codebases from other materials. Specifically, I used the commercially available comparison tool "Beyond Compare" to confirm that the following codebases are identically equivalent to one another:

<u>Location within PA-NAT 78</u>	<u>Equivalent to:</u>	<u>Short form:</u>
\PA-NAT-00000078\viola\1993-05-12 DX34 - Ex A to Inv Contentions	Contents of [PA-NAT-00000020]	[Viola-5/12/93
\PA-NAT-00000078\viola\1993-05-27 DX37 - Ex AE to Inv Contentions	Contents of [PA-NAT-00000021]	Viola-5/27/93



\PA-NAT-00000078\viola\1993-10-16 Alpha release- Ex AF to Inv Contentions	violaTOGO.tar.Z in the disc [PA-NAT-00000024] ("1 - Tar Secondaries") in the folder 047263-000006\usr2\users\pei\.	Viola-Alpha
\PA-NAT-00000078\viola\1994-02-23 February beta release - Ex AG to Inv Contentions	viola940223.tar.gz in the folder 051209_1055(E)\ in [PA-NAT-00000023]	Viola-Feb Beta
\PA-NAT-00000078\viola\1994-03-23 March beta release - Ex AH to Inv Contentions\viola940323	Contents of 047263-000004\usr\work\viola940323 obtained from the disc [PA-NAT-00000024]	Viola-March Beta
\PA-NAT-00000078\vplot\Tar Secondaries\000007_usr_users\scott\vplot (binary)	Contents of 047263-000007\usr\users\scott\vplot, in [PA-NAT-00000024]	Vplot-Sun
\PA-NAT-00000078\vplot\MS_SUPP_1205_001\petra\vplot	The folder petra\vplot in PA-NAT-00000022 in the folder 051208_1551(E)\	Vplot-petra

#### b. Vplot

442. It is my opinion that the vplot executable application is prior art to the patents-in-suit.

443. Although there are many instances of vplot source code and executables located throughout the materials I reviewed, for simplicity I cite two vplot implementations.

444. The first is [vplot-sun], which I believe to be the vplot binary that was used in a May 7, 1993 demonstration of Viola to Sun. The vplot binary is not part of [Viola-5/12/93], however, as discussed above, [Viola-5/12/93] is functionally the same as the version of Viola that was used to demonstrate Viola and [vplot-sun] to Sun on May 7, 1993. [vplot-sun] is also contained in the complete DX34 trial exhibit, admitted during the *Eolas v. Microsoft* litigation. My review of the vplot binary

shows that it has a last-modified date of May 7, 1993. This binary is discussed in more detail in connection with [Viola-5/12/93] below.

445. This vplot binary is prior art at least because it was known or used by others in this country before the invention by the applicants of the patents-in-suit, because it was in public use in this country more than one year prior to the date of the application of the patents-in-suit, and because it was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed.

446. The second is [vplot-petra], which has a vplot executable with last-modified date of August 12, 1993. Accordingly, [vplot-petra] is prior art at least because it was known or used by others in this country before the invention by the applicants of the patents-in-suit, because it was in public use in this country more than one year prior to the date of the application of the patents-in-suit, and because it was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed.

447. Vplot was one example of an executable application which ViolaWWW could interoperate with. The vplot versions that I have reviewed could interoperate with any of the ViolaWWW codebases that I reviewed.

448. To the extent that there is any dispute that the vplot instances identified above are prior art, I reserve the right to rely on other prior art vplot instances, including any of those located in the directory *PA-NAT-78\vplot*.

449. To the extent Eolas asserts that any of the claims (either based on their ordinary meaning or in light of a claim construction) do not require a browser application, and can be infringed solely based on an external executable application that can interoperate with a browser application, I believe that the vplot executable application would anticipate such a claim.

**c. Doodle**

450. In view of the breadth of Eolas's application of the term "executable application" in [Martin11] and in view of the Court's supplemental order regarding claim construction, if the term "executable application" is not limited to executable applications that are in the general form of native binaries, as I had assumed in my initial report for the reasons described above, it is my opinion that the doodle script included with all the Viola code bases is also prior art to the patents in suit.

451. Although there are many instances of the doodle script located through the materials I reviewed, for simplicity I refer to the implementation that was included in the [Viola-5/27/93] code base that Wei posted to his FTP site. This particular doodle script can be found in [Viola-5/27/93] by extracting the Viola code at " PA-NAT-00000078\viola\1993-05-12 DX34 - Ex A to Inv Contentions \violaTOGO.tar.Z" and looking within the extracted folders to "DX37 violaTOGO.tar \violaTOGO\violaTOGO\apps" for the "doodle.v" file.

452. This doodle script is prior art at least because it was published before the invention by the applicants of the patents-in-suit , because it was known or used by others in this country before the invention by the applicants of the patents-in-suit, because it was in public use in this country more than one year prior to the date of the application of the patents-in-suit, and because it was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed.

453. Doodle was a second example of an executable application, under [Martin11]'s application of the claim terms, which could interoperate with ViolaWWW.

454. Viola used the doodle as an embedded interactive application in the same manner as vplot. As described in greater detail in the claim chart attached to this report, vplot is launched when the Viola browser parses the testplot.html file.

(See Claim Chart Exhibit 2). When Viola parses the `violaApps.html` file within the Viola browser it encounters the `<VOBJF>` tag discussed in greater detail as it relates to the `testplot.html` file for `vplot`. In `violaApps.html` the `<VOBJF>` tag references `doodle.v`. The following text from the `violaApps.html` file shows the code directing the Viola browser to the doodle application:

```
<H2>Noodle Doodles</H2>  
<VOBJF>../apps/doodle.v</VOBJF>
```

(See `violaTOGO\docs\violaApps.html`). The Viola browser finds the location of the doodle script in the previously identified `\apps` directory and creates an embedded interactive doodle application within the `violaApps.html` hypermedia document. My analysis with regards to `vplot` as anticipatory prior art applies with regard to doodle to the extent the court determines that the term "executable application" is not limited to executable applications that are in the general form of native binaries.

455. To the extent that there is any dispute that the doodle instance identified above is prior art, I reserve the right to rely on other prior art doodle instances, including any of those located in the directory *PA-NAT-78*.

456. To the extent Eolas asserts that any of the claims (either based on their ordinary meaning or in light of a claim construction) do not require a browser application, and can be infringed solely based on an external executable application that can interoperate with a browser application, I believe that the doodle executable application, or any of the other alleged executable applications alleged to infringe by Eolas, would anticipate and/or render obvious such a claim.

**d. Viola-5/12/93**

457. It is my opinion that [Viola-5/12/93] is prior art to the patents-in-suit.

458. There is ample evidence that [Viola-5/12/93] was known or used by others in this country before the invention by the applicants of the patents-in-suit.

Among other evidence, [ora-00293042], a May 4, 1993 email from Dougherty to Wei, discusses arranging a Viola demonstration for Sun Microsystems engineers to be held on May 7, 1993.

459. [ora-00293056] is a May 8, 1993 follow-up email from Dougherty to Wei and other staff at O'Reilly Associates (ora.org) that describes the success of the 5/7/93 meeting. In particular that reference states, referring to Sun employees, "They heard about what we are doing through the ora.com catazine and are interested in Viola and WWW as sample applications of their technology. They were very impressed with what we are doing, and couldn't wait to show it around at Sun. In particular, they said that their user interface group, especially Bruce Tognazzi (sp?), formerly of Apple, was really interested in getting into the issues that we are developing - that a document becomes a user interface for the network; *that programs are embedded in documents*, etc." Preceding emphasis is mine.

460. It is also my understanding that the Federal Circuit held that a district court erred in finding that the demonstration of Viola was not a prior art public use. The relevant passage from the ruling is, "In sum, with respect to the district court's prior art rulings, this court finds: the district court erred in finding as a matter of law that Viola was abandoned, suppressed or concealed within the meaning of section 102(g); Wei's May 7, 1993 demonstration to two Sun Microsystems employees without confidentiality agreements was a public use under section 102(b)." I reviewed the Federal Circuit's opinion, which I understand is cited as 399 F.3d 1325 (Fed. Cir. 2005).

461. As further evidence of this demonstration to Sun, I identified a *vplot* binary dated May 7, 1993, the same date that the references above identify as the date of the demonstration to Sun. In [Vplot-Sun], upon issuing the *dir* command, one finds the listing:

05/07/1993 05:35 PM 98,304 vplot

462. Viola was also presented to Usenix on May 7, 1993, the same day as it was presented to Sun. According to Scott Silvey, who attended this presentation, the demonstration to Usenix used the same code that had been demonstrated to the Sun engineers earlier in the day. [Silvey Tr.:210:6-9; 238:3-18]. Therefore, I believe that [vplot-sun] was demonstrated to both Sun and Usenix on May 7, 1993. And, like the Sun presentation, there is no indication that the Usenix presentation was protected by confidentiality agreements.

463. Other date samplings of the Viola codebase files provide clear evidence that the Viola codebase was known or used by others in this country before the invention by the applicants of the patents-in-suit. Specifically, in Viola, after unarchiving, one finds that the extracted files have "date modified" fields of 5/12/1993. Thus, I believe that Pei Wei archived the codebase he demonstrated to Sun shortly after the demonstration.

464. It is therefore clear to me that [Viola-5/12/93] was known or used by others in this country before the invention by the applicants of the patents-in-suit. The [ora-00293042] and [ora-00293056] references, the dates of the files in Viola, and the Federal Circuit opinion, and the associated discussion above provides ample evidence of this assertion.

465. It is also clear to me that [Viola-5/12/93] was in public use in this country before October 17, 1993. The references [ora-00293042] and [ora-00293056], the Federal Circuit opinion, the May 12, 1993 date for files in the archive [Viola-5/12/93] , the May 7, 1993 date for [Vplot-Sun], and the associated discussion above, all provide ample evidence of this assertion.

466. It is also my opinion that [Viola-5/12/93] was made in this country before the invention by the applicants for the patents-in-suit, and was not abandoned, suppressed, or concealed. The references [ora-00293042] and [ora-00293056], the Federal Circuit opinion, the May 12, 1993 date for files in the archive

[Viola-5/12/93], the May 7, 1993 date for [Vplot-Sun], and the associated discussion above, all provide ample evidence of this assertion.

467. In addition, trial testimony from Pei Wei, Scott Silvey, and Dale Dougherty corroborates that [Viola-5/12/93] is prior art to the patents-in-suit. In particular, Pei Wei's testimony from July 28-29, 2003 in connection with the trial *Eolas Technologies, Inc. v. Microsoft Corp.*, No. 99-C-626 (N. D. Ill. Jul. 8, 2003 – Aug. 7, 2003) corroborates my conclusions above. Scott Silvey and Dale Dougherty's testimony from July 29, 2003 from that same trial likewise corroborate my conclusions. (See EOLASTX-E-0000000644.) Scott Silvey's deposition testimony from this case, taken on September 15, 2011, further confirms both my conclusions above, and his prior testimony during the Microsoft litigation. See Silvey Tr. 238:13-239:3. In addition, a portion of the deposition testimony of Karl Jacob (one of the Sun engineers who witnessed the Viola demonstration) from October 4, 2001 in connection with *Eolas Technologies, Inc. v. Microsoft Corp.*, No. 99-C-626 corroborates that [Viola-5/12/93] was prior art to the patents-in-suit. (See *Eolas Technologies, Inc. v. Microsoft Corp.*, No. 99-C-626, Document # 816-6.)

468. As still further corroboration, an email exchange between Pei Wei and Michael Doyle corroborates that [Viola-5/12/93] is prior art. The contents of the email exchange can generally be seen at the following documents: [www-talk-00293117]; [www-talk-00293119]; [www-talk-00293120]; [www-talk-00293122]; [www-talk-00293123]; [www-talk-00293124]; [www-talk-00293126]; [www-talk-00293128]; [www-talk-00318682]; [www-talk-00293130]; [www-talk-00293138]; [www-talk-00293140]. I summarize certain key points from this email exchange below, in my discussion of why Pei Wei's August 1994 paper [Wei94] renders obvious all claims of the patents-in-suit.

469. Finally, I prepared videos demonstrating [Viola-5/12/93]. (See Appendix C at Viola Video 1.wmv, Viola Video 2.wmv, Viola Video 3.wmv, Viola

Video 4.wmv, Viola Video 5.wmv, Viola Video 6.wmv, Viola Video 7.wmv, Viola Video 8.wmv, Viola Video 8B.wmv, Viola Video 9.wmv.) I am aware that Scott Silvey has reviewed these videos, and has concluded that the videos demonstrating Vplot “looked exactly as we demonstrated it [to Sun]”. Mr. Silvey “remember[ed] even the colors, the geometry, everything was like that.” Silvey Tr. 238:13-239:3. The videos are thus an accurate representation both of the abilities of Viola on May 7, 1993, and of the content of the presentation that was given to Sun on that date.

470. It is my opinion that [Viola-5/12/93] anticipates claim 1 (and dependent claims 2, 3, and 11), and claim 6 (and dependent claims 7, 8, and 13) of the '906 patent. It is also my opinion that [Viola-5/12/93] anticipates claim 1 (and dependent claims 2-11), claim 16 (and dependent claims 17-19), claim 20 (and dependent claims 21-23), claim 24 (and dependent claims 25-27), claim 28, claim 36 (and dependent claims 37-39), and claim 40 (and dependent claims 41-43) of the '985 patent. In my Invalidity Claim Chart Based on [Viola-5/12/93] , appended hereto as Claim Chart Exhibit 1, I have shown on an element by element basis that [Viola-5/12/93] constitutes anticipatory prior art for all asserted claims of the '906 and '985 patents.

**e. Viola-5/27/93**

471. There is ample evidence that [Viola-5/27/93] was known or used by others in this country before the invention by the applicants of the patents-in-suit. Among other evidence, [ora-00293070], a May 31, 1993 email from Wei to James Kempf, a Sun engineer, informs him of a public ftp posting of viola.tar.Z, the [Viola-5/27/93] codebase.

472. The reference [ora-00293071] documents an exchange of emails between Wei and two Sun engineers, James Kempf and Karl Jacob about the ftp location for the latest version of [Viola-5/27/93]. There he states, "Ok, the tar is on



xcf.berkeley.edu in tmp/violaTOGO.tar." For this and codebases, Wei continued to use the terminology "TOGO" when intending to publish them for ftp access.

473. The reference [ora-00293074] documents an exchange of emails between Dougherty, Wei and two Sun engineers, James Kempf and Karl Jacob. The general thread of the document is the current status of [Viola-5/27/93] and how Sun can use it. One comment from Kempf was, "Also, about Viola. We are collecting apps for some demos at the moment, and would like to include it as a particularly creative use of WWW."

474. In addition, the codebase [Viola-5/27/93], after unarchiving, has files with "date modified" fields of May 27, 1993, suggesting that this was the codebase that Wei described above for his FTP posting.

475. It is also my understanding that the Federal Circuit decision I describe above held that a district court erred in finding that the demonstration of Viola was not a prior art public use. The relevant passage from the ruling is, "In sum, with respect to the district court's prior art rulings, this court finds: the district court erred in finding as a matter of law that Viola was abandoned, suppressed or concealed within the meaning of section 102(g); Wei's May 7, 1993 demonstration to two Sun Microsystems employees without confidentiality agreements was a public use under section 102(b)." Also, with respect to [Viola-5/27/93], that ruling said, "and the district court erred in its JMOL that [Viola-5/27/93] did not as a matter of law anticipate or render the '906 patent obvious. As a result, this court remands for additional proceedings on these issues."

476. It is clear to me that [Viola-5/27/93] was published before the invention by the applicants of the patents-in-suit. The [ora-00293070], [ora-00293071], [ora-00293074], dates of [Viola-5/27/93], the Federal Circuit decision, along with the associated discussion above, provides ample evidence of this assertion.

477. I believe [Viola-5/27/93] was in public use in this country more than one year prior to the date of the application of the patents-in-suit. The [ora-00293070], [ora-00293071], [ora-00293074], dates of [Viola-5/27/93], the Federal Circuit decision, along with the associated discussion above, provides ample evidence of this assertion.

478. In my opinion [Viola-5/27/93] was published more than one year prior to the date of application of the patents-in-suit. The [ora-00293070], [ora-00293071], [ora-00293074], dates of [Viola-5/27/93], the Federal Circuit decision, along with the associated discussion above, provides ample evidence of this assertion.

479. It is my opinion that [Viola-5/27/93] was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed. The [ora-00293070], [ora-00293071], [ora-00293074], dates of [Viola-5/27/93], the Federal Circuit decision, along with the associated discussion above, provides ample evidence of this assertion.

480. In addition, trial testimony from Pei Wei, Scott Silvey, and Dale Dougherty corroborates that [Viola-5/27/93] is prior art to the patents-in-suit. In particular, Pei Wei's testimony from July 28-29, 2003 in connection with the trial *Eolas Technologies, Inc. v. Microsoft Corp.*, No. 99-C-626 (N. D. Ill. Jul. 8, 2003 – Aug. 7, 2003) corroborates the my conclusions above. Scott Silvey and Dale Dougherty's testimony from July 29, 2003 from that same trial likewise corroborate my conclusions. (See EOLASTX-E-0000000644.) In addition, a portion of the deposition testimony of Karl Jacob (one of the Sun engineers who witnessed the Viola demonstration) from October 4, 2001 in connection with *Eolas Technologies, Inc. v. Microsoft Corp.*, No. 99-C-626 corroborates that [Viola-5/27/93] was prior art to the patents-in-suit. (See *Eolas Technologies, Inc. v. Microsoft Corp.*, No. 99-C-626, Document # 816-6.)

481. As still further corroboration, an email exchange between Pei Wei and Michael Doyle corroborates that [Viola-5/27/93] is prior art. The contents of the

email exchange can generally be seen at the following documents: [www-talk-00293117]; [www-talk-00293119]; [www-talk-00293120]; [www-talk-00293122]; [www-talk-00293123]; [www-talk-00293124]; [www-talk-00293126]; [www-talk-00293128]; [www-talk-00318682]; [www-talk-00293130]; [www-talk-00293138]; [www-talk-00293140].

482. Finally, I prepared videos demonstrating [Viola-5/27/93]. (See Appendix C at Viola Video 10 - A.wmv, Viola Video 10 - B.wmv, Viola Video 11.wmv, Viola Video 12.wmv, Viola Video 13.wmv, Viola Video 14.wmv, Viola Video 15.wmv.)

483. It is my opinion that [Viola-5/27/93] anticipates claim 1 (and dependent claims 2, 3, and 11), and claim 6 (and dependent claims 7, 8, and 13) of the '906 patent. It is also my opinion that [Viola-5/27/93] anticipates claim 1 (and dependent claims 2-11), claim 16 (and dependent claims 17-19), claim 20 (and dependent claims 21-23), claim 24 (and dependent claims 25-27), claim 28, claim 36 (and dependent claims 37-39), and claim 40 (and dependent claims 41-43) of the '985 patent.

484. In my Invalidity Claim Chart Based on [Viola-5/27/93], appended hereto as Claim Chart Exhibit 2, I have shown on an element by element basis that [Viola-5/27/93] constitutes anticipatory prior art for all asserted claims of the '906 and '985 patents.

**f. Alpha release**

485. There is ample evidence that the [Viola-alpha] was known or used by others in this country before the invention by the applicants of the patents-in-suit. Among other evidence, [ora-00293084], an October 14, 1993 email from Wei, announces the status of the alpha release. In it he states, "It's so so close. Will do another pass at testings today. All of what I set out to do two weeks ago have been done, except for network protocol testings and complete documentation."

486. The email, [ora-00293086], from Scott Silvey to Wei indicates that the first alpha release be distributed to ORA staff first, before a more general release. An Alpha release status update email from Wei to ORA staff, [ora-00293088], states "The tar is /home/wei/violaTOGO.tar.z which you can drop in any directory."

487. As noted above, Wei continued to use the terminology "TOGO" when intending to publish them for ftp access. This filename (violaTOGO.tar.z) is the same filename as used in [Viola-alpha]. Furthermore, if one extracts the files from [viola-alpha], they have "date modified" fields of October 16, 1993, suggesting that this codebase is the same codebase that Wei was emailing about on or around October 16, 1993.

488. For example, for the directory, (*\PA-NAT-78\viola\1993-10-16 Alpha release- Ex AF to Inv Contentions\viola\ violaTOGO\demo\objs*), upon issuing the *dir* command, one finds the listing:

```
10/16/1993 09:20 AM      10,199 plot.v
```

489. Then, in [ora-00293089], Wei announces the posting of [viola-alpha] with the information, "OK, Jay. an alpha release is available at ftp://xcf.berkeley.edu/src/local/viola/tmp/violaWWW931117.tar.z." Shortly after that Wei received email from Jason Bluming, [ora-00293090], who had already downloaded the Alpha release and stated, "Having heard about the power of your Viola environment, I was wondering how hard it would be to add a relatively complete DTD-interpreting structure to it."

490. It is clear to me that [viola-alpha] release was published before the invention by the applicants of the patents-in-suit. The [ora-00293084], [ora-00293086], [ora-00293088], [ora-00293089], and [ora-00293090] references, the dates on the files found in [viola-alpha], and the associated discussion above provide ample evidence of this assertion.

491. It is also clear that [viola-alpha] was in public use in this country more than one year prior to the date of application of the patents-in-suit. The [ora-00293084], [ora-00293086], [ora-00293088], [ora-00293089], and [ora-00293090] references, the dates on the files found in [viola-alpha], and the associated discussion above provide ample evidence of this assertion.

492. In my opinion [viola-alpha] was a printed publication more than one year prior to the date of application of the patents-in-suit. The [ora-00293084], [ora-00293086], [ora-00293088], [ora-00293089], and [ora-00293090] references, the dates on the files found in [viola-alpha], and the associated discussion above provide ample evidence of this assertion.

493. Also, it is clear that [viola-alpha] was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed. The [ora-00293084], [ora-00293086], [ora-00293088], [ora-00293089], and [ora-00293090] references, the dates on the files found in [viola-alpha], and the associated discussion above provide ample evidence of this assertion

494. In addition, trial testimony from Pei Wei, Scott Silvey, and Dale Dougherty corroborates that [Viola-Alpha] is prior art to the patents-in-suit. In particular, Pei Wei's testimony from July 28-29, 2003 in connection with the trial *Eolas Technologies, Inc. v. Microsoft Corp.*, No. 99-C-626 (N. D. Ill. Jul. 8, 2003 – Aug. 7, 2003) corroborates the my conclusions above. Scott Silvey and Dale Dougherty's testimony from July 29, 2003 from that same trial likewise corroborate my conclusions. (See EOLASTX-E-0000000644; see also Wei and Bina transcripts from this case.)

495. Finally, I prepared videos demonstrating [Viola-alpha]. (See Appendix C, at Viola Video 16 - A.wmv, Viola Video 16B.wmv, Viola Video 17.wmv, Viola Video 19.wmv).

496. It is my opinion that [viola-alpha] anticipates claim 1 (and dependent claims 2, 3, and 11), and claim 6 (and dependent claims 7, 8, and 13) of the '906

patent. It is also my opinion that ViolaWWW Alpha release anticipates claim 1 (and dependent claims 2-11), claim 16 (and dependent claims 17-19), claim 20 (and dependent claims 21-23), claim 24 (and dependent claims 25-27), claim 28, claim 36 (and dependent claims 37-39), and claim 40 (and dependent claims 41-43) of the '985 patent.

497. In my Invalidity Claim Chart Based on [viola-alpha], appended hereto as Claim Chart Exhibit 3, I have shown on an element by element basis that [viola-alpha] constitutes anticipatory prior art for all asserted claims of the '906 and '985 patents.

**g. February beta release**

498. There is ample evidence that [Viola-Feb Beta] was known or used by others in this country before the invention by the applicants of the patents-in-suit. Among other evidence, [ora-00293098], a February 25, 1994 email from Wei entitled "ViolaWWW beta release available", announces the posting of the beta release. In it he states, "The new ViolaWWW is now available for ftp'ing. It's beta and feedback is very welcomed. The README file follows...." He further states, "Based on and drawing from the Viola scripting language and toolkit, ViolaWWW provides a way to build relatively complex hypermedia applications that are beyond the provisions of the current HTML standard." To access the release he states, "Source and binary can be found in <ftp://ora.com/pub/www/viola>. Sparc binary is supplied."

499. The "Viola Stuff" folder [PA-00318733] includes further corroboration that [Viola-Feb Beta] is prior art. This is because the "Viola Stuff" folder includes a printout of the email I describe above, showing that it was not only downloaded, but also printed and read.

500. When one examines the files in [Viola-Feb Beta], one sees that the "date modified" files on these files are February 23, 1994.

501. For example, for the directory, (*\PA-NAT-78\viola\1994-02-23 February beta release - Ex AG to Inv Contentions\viola\violaDocs\objs*), upon issuing the *dir* command, one finds the listing:

02/23/1994 05:11 PM 10,199 plot.v

502. Thus, it is my opinion that [Viola-Feb Beta] is what Wei emailed about in [ora-00293098].

503. It is my opinion that [Viola-Feb Beta] was published before the invention by the applicants of the patents-in-suit. The [ora-00293098] reference, the dates of the files in [Viola-Feb Beta], and the associated discussion above provide ample evidence of this assertion.

504. It is my opinion that [Viola-Feb Beta] was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed. The [ora-00293098] reference, the dates of the files in [Viola-Feb Beta], and the associated discussion above provide ample evidence of this assertion.

505. In addition, trial testimony from Pei Wei, Scott Silvey, and Dale Dougherty corroborates that [Viola-Feb Beta] is prior art to the patents-in-suit. In particular, Pei Wei's testimony from July 28-29, 2003 in connection with the trial *Eolas Technologies, Inc. v. Microsoft Corp.*, No. 99-C-626 (N. D. Ill. Jul. 8, 2003 – Aug. 7, 2003) corroborates the my conclusions above. Scott Silvey and Dale Dougherty's testimony from July 29, 2003 from that same trial likewise corroborate my conclusions. Scott Silvey and Dale Dougherty's testimony from July 29, 2003 from that same trial likewise corroborate my conclusions. (See EOLASTX-E-0000000644.)

506. In addition, I prepared videos demonstrating [Viola-Feb beta]. (See Appendix C at Viola Video 21.wmv, Viola Video 24.wmv.)

507. It is my opinion that [Viola-Feb Beta] anticipates claim 1 (and dependent claims 2, 3, and 11), and claim 6 (and dependent claims 7, 8, and 13) of the

'906 patent. It is also my opinion that ViolaWWW February beta release anticipates claim 1 (and dependent claims 2–11), claim 16 (and dependent claims 17–19), claim 20 (and dependent claims 21–23), claim 24 (and dependent claims 25–27), claim 28, claim 36 (and dependent claims 37–39), and claim 40 (and dependent claims 41–43) of the '985 patent.

508. In my Invalidity Claim Chart Based on [Viola-Feb Beta], appended hereto as Claim Chart Exhibit 4, I have shown on an element by element basis that [Viola-Feb Beta] constitutes anticipatory prior art for all asserted claims of the '906 and '985 patents.

509. I note that in my invalidity claim chart for [Viola-Feb Beta], I focus on ViolaWWW's ability to embed interactive objects using the HTML tags LINK and VOBJF. However, I further note that [Viola-Feb Beta] could embed interactive objects using the HMML VOBJF tag in like manner as could be done for [Viola-Alpha]. One would just need to change the security flag that I described in my report above as to enable embedding. Thus, while my claim chart describes HTML functionality, I also believe that [Viola-Feb Beta] and its HMML functionality using the VOBJF tag anticipate all asserted claims for the same reasons that I explain in my claim chart for [Viola-alpha].

#### **h. March beta release**

510. There is ample evidence that [Viola-March Beta] was known or used by others in this country before the invention by the applicants of the patents-in-suit. Among other evidence, [ora-00293100], a March 24, 1994 email from Wei entitled "ViolaWWW Release", announces the posting of a second beta release. In it he states, "Another beta release of Viola is now available. Source and binary can be found in "ftp://ora.com/pub/www/viola". For more information please see <http://xcf.berkeley.edu/ht/projects/viola/README>.



511. When one examines the files in [Viola-March Beta], one sees that the files have a "date modified" field set to March 23, 1994, indicating that this is the codebase Wei was referring to in [ora-00293100].

512. For example, for the directory, (`\PA-NAT-78\viola\1994-03-23 March beta release - Ex AH to Inv Contentions\viola940323\violaDocs\objs`), upon issuing the `dir` command, one finds the listing:

```
03/23/1994 07:04 PM          9,715 plot.v
```

513. Also, the "Viola Stuff" folder [PA-00318733] provides corroboration because it includes a printout stating the following:

```
514.
ViolaWWW, Version 3.1 Beta Mar 23 1994
=====
ViolaWWW is an extensible World Wide Web hypermedia
browser for XWindows.
. . . .
Notable features in the new ViolaWWW
-----
. . . .
* Embeddable in-document and in-toolbar programmable viola
objects. A document can embed mini viola applications (ie: a
chess board), or can cause mini apps to be placed in the toolbar.
. . . .
Availability
-----
Source and binary can be found in ftp://ora.com/pub/www/viola.
Sparc binary is supplied.
. . . .
Pei Y. Wei (wei@ora.com)
O'Reilly & Associates, Inc.
```

515. It is my opinion [Viola-March Beta] was published before the invention by the applicants of the patents-in-suit. The [ora-00293100] reference, the dates of the files in [Viola-March beta], and the associated discussion above provide ample evidence of this assertion.

516. It is my opinion that [Viola-March Beta] was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed. The [ora-00293100] reference, the dates of the files in [Viola-March beta], and the associated discussion above provide ample evidence of this assertion.

517. In addition, trial testimony from Pei Wei, Scott Silvey, and Dale Dougherty corroborates that [Viola-March Beta] is prior art to the patents-in-suit. In particular, Pei Wei's testimony from July 28-29, 2003 in connection with the trial *Eolas Technologies, Inc. v. Microsoft Corp.*, No. 99-C-626 (N. D. Ill. Jul. 8, 2003 – Aug. 7, 2003) corroborates the my conclusions above. Scott Silvey and Dale Dougherty's testimony from July 29, 2003 from that same trial likewise corroborate my conclusions.

518. It is my opinion [Viola-March Beta] anticipates claim 1 (and dependent claims 2, 3, and 11), and claim 6 (and dependent claims 7, 8, and 13) of the '906 patent. It is also my opinion that ViolaWWW March beta release anticipates claim 1 (and dependent claims 2-11), claim 16 (and dependent claims 17-19), claim 20 (and dependent claims 21-23), claim 24 (and dependent claims 25-27), claim 28, claim 36 (and dependent claims 37-39), and claim 40 (and dependent claims 41-43) of the '985 patent.

519. In my Invalidity Claim Chart Based on [Viola-March Beta], appended hereto as Exhibit 5, I have shown on an element by element basis that [Viola-Feb Beta] constitutes anticipatory prior art for all asserted claims of the '906 and '985 patents.

520. I note that in my invalidity claim chart for [Viola-March Beta], I focus on ViolaWWW's ability to embed interactive objects using the HTML tags LINK and VOBJF. However, I further note that [Viola-March Beta] provided support for embedding interactive objects using the HMML VOBJF tag, which would operate in like manner as [Viola-Alpha]. One would need to change the security flag that I described in my report above as to enable embedding. Thus, while my claim chart describes HTML functionality, I also believe that [Viola-March Beta] teaches the use of HMML functionality using the VOBJF tag, and this teaching anticipates all asserted claims for the same reasons that I explain in my claim chart for [Viola-alpha].

**i. Other codebases and materials**

521. I understand that other versions of the code bases identified above may exist among materials produced by Defendants during the course of this litigation. I reserve the right to rely on those codebases to the extent they support my analysis or help me to rebut assertions from Eolas.

522. I further understand that other Viola, xplot, or vplot code bases dated during the 1992-1994 time period exist, and I reserve the right to rely on those codebases to the extent they support my analysis or help me to rebut assertions from Eolas. As one example, such codebases may help show the trajectory of Viola's development, including when certain features or capabilities were developed or removed.

523. I further note that materials kept by Viola, xplot, or vplot developers from the time period in which these applications were being developed help to show what was known and what was available to a person of ordinary skill in the art. Accordingly, I reserve the right to rely on such materials in support of my analysis or help me to rebut assertions from Eolas.

524. Examples of where such other code bases or materials can be found include:

- the DVDs labeled "1 - Tar Secondaries" [PA-NAT-00000024], "2 - Solaris Secondaries" [PA-NAT-00000025], "3 - TAR Images" [PA-NAT-00000026], and "4 - Solaris images" [PA-NAT-00000027], containing relevant materials that were maintained by Pei Wei, Scott Silvey, and/or O'Reilly and Associates.
- The discs labeled DX34 and DX37. [PA-NAT-00000020 and PA-NAT-00000021] containing relevant materials that were maintained by Pei Wei, Scott Silvey, and/or O'Reilly and Associates
- The CD-ROMs labeled "MS\_SUPP\_1205\_001" [PA-NAT-00000022] and "MS\_SUPP\_1205\_002." [PA-NAT-00000023] containing relevant materials that were maintained by Pei Wei, Scott Silvey, and/or O'Reilly and Associates
- The disc labeled "SunOS4.1.3.cpio.gz" containing the SunOS 4.1.3 operating system. [PA-NAT-00000029.]
- The discs labeled "Viola Email 93-94 + demo from UCB / Subp94." [PA-NAT-00000030], and [PA-NAT-00000062], both containing other Viola-related materials.

**j. Conferences and demonstrations**

525. To the extent that the Viola functionality I identified above in connection with [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], or [Viola-March beta] was demonstrated, it is my opinion that such a demonstration would be an anticipatory prior art event as showing that ViolaWWW and its relevant functionality was known or used by others in this country before the invention by the applicants of the patents-in-suit and that ViolaWWW and its relevant functionality was made in this country before the invention by the applicants for the patents-in-

suit and was not abandoned, suppressed, or concealed. For demonstrations taking place by October 17, 1993, such demonstration show that ViolaWWW and its relevant functionality was in public use more than one year prior to the date of application of the patents-in-suit.

526. There were at least two conferences or demonstrations in which ViolaWWW was demonstrated.

527. One public demonstration was a lecture by Pei Wei at Stanford called "WWW Browsers: Extensibility issues. An architecture of Extensibility and Plug-in Components in Browsers," which took place on September 20-21, 1994. Corroborating evidence includes the schedule for that conference [Stanford-schedule], the abstract and slides from that talk [Wei94b] which I discussed in greater detail above, and prior corroborating testimony of Pei Wei [EOLASTX-E-0000000644]. This lecture shows that the relevant functionality of ViolaWWW that I found anticipatory in connection with the Viola codebases was publically used, made, and not abandoned, suppressed, or concealed as of the lecture's date. For example, one figure from the slides (shown at [PA-00293135]) shows the example by which Viola interoperates with vplot ("[t]his is a demo of ViolaWWW embedding a viola front-ending object that is programmed to start up and communicate with a plot process. The front-end tells the plot program the window ID to draw to, and gives it the camera coordinate changes.") Also shown is a figure demonstrating the Chess application (shown at [PA-00293136]) that I discuss in this report as an example of a distributed application.

528. Another was the World-Wide Web Wizards Workshop ("WWW Wizards Conference"), which took place from Wednesday July 28, 1993 – Friday July 30, 1993, in Cambridge, Massachusetts. Corroborating evidence includes the email message from Dale Dougherty announcing the event [www-talk-00293081], photographs of the attendees of this conference [www-conference-photos], and prior

corroborating testimony of Pei Wei, Scott Silvey, and Dale Dougherty [EOLASTX-E-0000000644].

**k. Pei Wei's August 1994 paper**

529. Pei Wei's August 16, 1994 paper was published and was publically available on the Internet as <http://www.viola.org/viola/violaIntro.html> before the invention by the applicants of the patents-in-suit. In response to an initial posting by Doyle to a publicly-accessible email distribution list on August 30, 1994, a series of responses and counter-responses between Doyle and Wei was set in motion. The essence of these missives is summarized below. Versions of this paper are available at [PA-00318355] (which is labeled DX93A) and at [PA-00318383] (which is labeled DX95). In addition, copies of earlier publications of this paper appear at [Viola-5/27/93] at \violaTOGO\docs; [Viola-alpha] at violaTOGO\docs\viola; [Viola-Feb Beta] at viola\violaDocs\vw; [Viola-March Beta] at \violaDocs\vw. I reviewed relevant portions of [PA-00318383], and also reviewed [PA-00318355] and find that both support invalidation of the patents-in-suit based with equal force and based on corresponding disclosure. By way of example, the text set forth in [PA-00318355] can be found in identical form at [PA-00318383] beginning at [PA-00318385]. Also, I have considered (in the alternative) that the pages spanning the bates range PA-00318383 through PA-00318392 as a separate prior art reference from the remainder of that document, and also believe that this range renders obvious the patents in suit, to the extent that the prior art status of the remainder of that document is called into question.

530. I note that Wei testified that there were two versions of the 1994 paper, an HTML version and a PostScript version. See Wei Tr. 90:1-8. Wei stated that the differences between the versions is that one of the versions contain more text describing features of HTML and document rendering, and that the PostScript version shows the chessboard applet inside the browser, and the other one shows it

cropped out. Wei stated that he cropped out the browser to save space. Wei Tr: 90:10-92-12.

531. On August 30, 1994, at approximately 11:15 p.m. California time, Doyle posted a "Press Release" (which can be seen within the document ([www-talk-00293120])) to the publicly-accessible VRML e-mail distribution list that included the following statements:

Researchers at the U. of California have created software for embedding interactive program objects within hypermedia documents. Previously, object linking and embedding (OLE) has been employed on single machines or local area networks using MS Windows -TM-. This UC software is the first instance where program objects have been embedded in documents over an open and distributed hypermedia environment such as the World Wide Web on the Internet.

532. On August 31, 1994, at approximately 6:52 p.m. California time, Pei Wei posted a response on the publicly-accessible VRML e-mail distribution list that included the following statements: "I don't think this is the first case of program objects embedded in docs and transported over the WWW. ViolaWWW has had this capabilities for months and months now." This can be seen at [www-talk-00293122].

533. Pei Wei's response included a link to an FTP site where anyone "interested in learning more about how violaWWW does this embedded objects thing can get a paper on it." [www-talk-00293122]. The paper cited by Pei Wei was entitled "A Brief Overview of the Viola Engine, and its Applications" and is the [Wei94] paper I discussed above. The paper cited by Pei Wei was dated August 16, 1994 – over two months before the application for the '906 patent was filed.

534. I understand that the Federal Circuit has noted that Doyle downloaded and read the paper.

535. On August 31, 1994, at approximately 9:06 p.m. California time, Doyle responded to Pei Wei's statement at approximately 6:52 p.m. that "I don't think this is the first case of program objects embedded in docs and transported over the

WWW. ViolaWWW has had this capabilities for months and months now." Doyle responded by asking Pei Wei, "How many months and months? We demonstrated our technology in 1993." [www-talk-00293119].

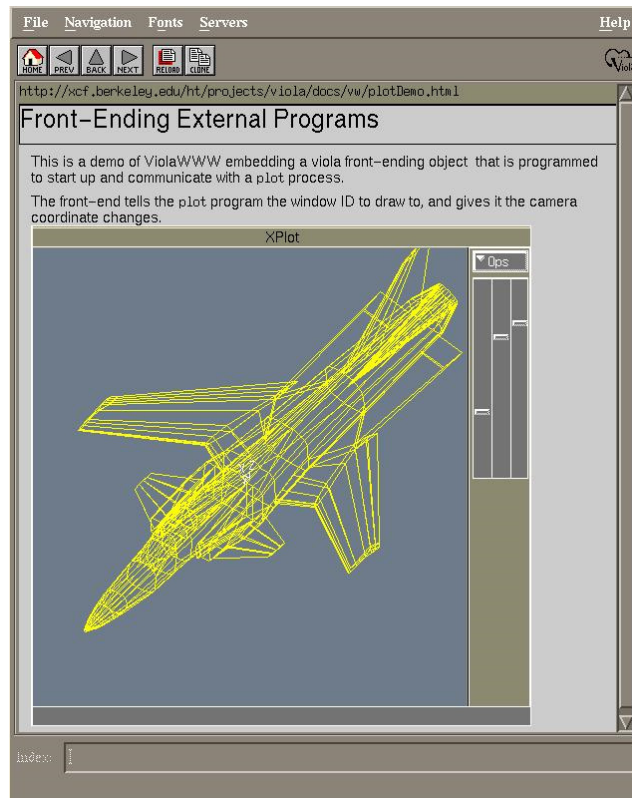
536. On August 31, 1994, at approximately 11:16 p.m. California time, Pei Wei responded to the message that Doyle had sent at approximately 9:06 p.m. Pei Wei's response (which can be seen at [www-talk-00293126]) included the following statements:

Definitely by May 8, 1993 we had demonstrated that plotting demo (the very one shown in the viola paper) to visitors from a certain computer manufacturer... This demo was memorable because someone and I at ORA had lost sleep the night before the meeting, in order to cook up that particular plotting demo :) We had to show something cool.

That demo wasn't very hard to do because by that time the basic capability was already in place for violaWWW to fetch viola objects over HTTP (or whatever) and plug them into documents. Of course, our wire-frame plotting demo isn't anywhere as comprehensive as yours. But, the point was that there was a way to embed programmable & interactive objects into HTML documents.

537. When Pei Wei referred to the "plotting demo (the very one shown in the viola paper)," he was referring to the plot of the fighter jet shown above in the window titled "XPlot." The figure representing that "plotting demo" is shown below as Figure 20.





**Figure 20. Front-end external program application**

538. When Pei Wei referred to a demonstration "by May 8, 1993" to "visitors from a certain computer manufacturer," he was referring to a demonstration of the plotting demo to Karl Jacob and James Kempf from Sun Microsystems on May 7, 1993. This demonstration took place in Northern California. There was no limitation, restriction or obligation of secrecy on Karl Jacob or James Kempf. (Pei Wei Trial Testimony, *Microsoft Corp.*, No. 99-C-626 (N. D. Ill. Jul. 8, 2003 – Aug. 7, 2003) [EOLASTX-E-000000644]).

539. As I discuss in this report, Wei's May 7, 1993 demonstration of Viola to Sun was a prior art event.

540. On August 31, 1994, at approximately 11:13 p.m. California time, Doyle responded again to the message that Pei Wei had sent at approximately 6:52 p.m.

Doyle's response was sent after Doyle had read Pei Wei's paper about the ViolaWWW browser dated August 16, 1994.

541. Doyle's response included the following statements: "Pei is mistaken on two counts, as I describe below . . . . As Pei's paper on Viola states, that package did not support what it calls 'embeddable program objects' until 1994. . . . Furthermore, Viola merely implements an internal scripting language . . . ." (See [www-talk-00293120]).

542. On August 31, 1994, at approximately 11:36 p.m. California time, Doyle responded to the message that Pei Wei had sent at approximately 11:16 p.m. Doyle's response included the following statements: "Out of curiosity, did you publicly demonstrate this or publish any results before 1994?" [www-talk-00293117]

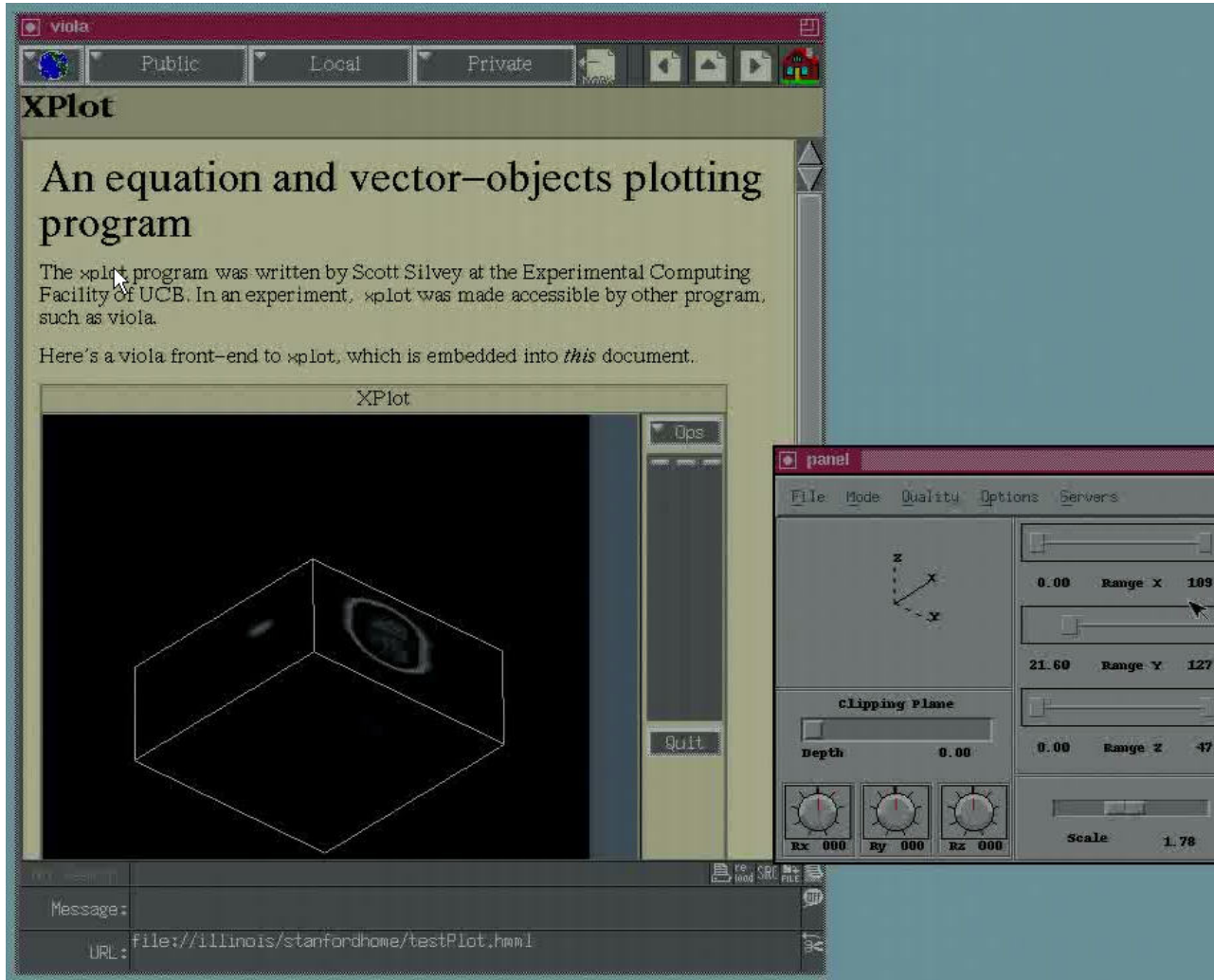
543. Pei Wei's message at approximately 12:08 a.m. was also responsive to the message that Doyle had sent at approximately 11:36 p.m. Pei Wei's message to Doyle (which can be seen at [www-talk-00293128]) at 12:08 a.m. included the following statements:

Well. Viola's model was \*demonstrated\* in 1993, \*released\* freely in 1994. . . . And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it.

544. The preceding description by Wei is the essence of how the "plotting demo" was produced. I have examined the relevant Viola code from [Viola-5/12/93] and it is clear that the window in which the jet fighter lines are drawn is embedded in the body of the ViolaWWW browser. Furthermore, the program that plots those lines, vplot, is a compiled binary executable that, in this case, resides on what Wei

calls the localhost and runs in a different UNIX process than the browser. There is no limitation in Viola, however, that prevents a binary executable from being accessed over a network, as I will demonstrate in what follows.

545. I have produced a video recording, [Viola video 9.wmv], that demonstrates how simple it is to make trivial changes to [Viola-5/12/93] code to effect a situation where ViolaWWW runs on a client workstation and accesses an executable and related dataset on a server machine. This demonstration used a version of plot.v accessing an HDF dataset through a VIS executable, both of which were accessed via NFS on a server residing on a network. The video carefully describes the changes made to plot.v and testPlot.hmml from the [Viola-5/12/93] codeset and shows the resulting display. A frame from that video is shown below.



**Figure 21. External VIS Executable and HDF File**

546. To reiterate a statement that Wei wrote to Doyle, "And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it." It is important to recognize that Wei told Doyle *exactly* how to modify Viola to accomplish what I have demonstrated in the above-cited demonstration. The back-end is running VIS on a remote server. The front-end, the client workstation, has

passed VIS an X window ID which it uses to draw the graphics directly onto the window *violaWWW* has opened for it.

547. One other gem that Wei passed on to Doyle is that, in general, *ViolaWWW* can use computational resources of a server anywhere on a network and thus can perform client-server operations. That is described in [Wei94] where he teaches how a chess board application embedded in the *ViolaWWW* browser can front-end a chess server. He states, "Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess board application can front-end a chess server, connected to it using the socket facility in *viola*." The chess application is shown below as Figure 22.

548. It is my opinion that the combination of prior art references [Wei94] and the collection of reciprocal emails between Wei and applicant Doyle of the patents-in-suit render obvious all of the limitations of all asserted claims of the '906



Figure 22. Distributed Chess App

and '985 patents. Thus, it would have been obvious for one of ordinary skill in the art at the time of the alleged invention to combine them to implement the functionality of the alleged invention.

549. In Claim Chart Exhibit 6, Invalidity Claim Chart Based on [Wei94], I have shown on an element by element basis that [Wei94] in combination with the email correspondence between Pei Wei and Michael Doyle described above constitutes invalidating prior art for all asserted claims of the '906 and '985 patents.

#### I. Claim construction

550. I understand that the Court has construed the claim term "automatically [invoking / invoke] [the / said] executable application" and "executable application is automatically invoked by the browser" as "the executable application is launched without user activation". Should Eolas or Eolas' experts attempt to interpret this construction in a way that I believe is incorrect, I may cite additional items for claim limitations related to "object."<sup>11</sup> By way of example, in [vplot-petra] in the directory \vplot\off, there is a file called x29.geom. In several of my Viola Videos accompanying this report (see Appendix C), such as Viola Video 11, 12, and 13 by way of example only, I show how the x29.geom file can be accessed from a menu in order to display a jet fighter object in place of the default grid object. To the extent it is found that this claim has no requirement for "without any intervening activation of the object by the user," I reserve the right to identify files like x29.geom as satisfying these limitations related to "object." Other exemplary files

---

<sup>11</sup> Limitations for "object" or relating to "object" include at least 906-1.f; 906-6.f; 985-1.f; 985-16.f; 985-20.g; 985-24.i; 985-28.i; 985-36.f; 985-40.g; 985-9.a; 985-36.f; 985-40.g; 906-1.f; 906-6.f; 985-36.f; 985-40.g; 985-1.f; 985-16.f; 985-20.g; 985-24.b; 985-24.i; 985-28.b; 985-28.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i; 906-1.g; 906-6.g; 985-1.f; 985-16.f; 985-20.g; 985-24.j; 985-28.j; 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.h; 985-20.i; 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-36.h; 985-40.i; 985-24.b; 985-28.b; 985-24.b; 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i.

I may point to are the other files located in that same vplot\off directory, or other analogous files located elsewhere within the Viola prior art.

551. Further, I understand that the Court has construed "executable application" to mean "any computer program code, that is not the operating system or a utility, that is launched to enable an end user to directly interact with data." To the extent that Eolas or Eolas' experts attempt to interpret this construction in ways that I believe are incorrect, I may cite Viola .v script files as the claimed executable application. Examples of files I may cite include, within [Viola-5/12/93], files located within viola\apps, such as doodle.v, and chier.v. I may also cite similar files located elsewhere within the Viola prior art.

552. In addition, I have reconsidered all other claim limitations in view of the claim construction order. I continue to reserve the right to supplement my analysis of Viola prior art accordingly should I find that Eolas or Eolas' experts take positions that I believe to be inconsistent with my understanding of the claim construction order.

553. To the extent the plaintiff argues that the Viola prior art code bases or references, including [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], [Viola-March Beta], or [Wei94], do not anticipate the claimed invention by contending that they do not include an executable application, the plaintiff is incorrect. For the reasons shown in my analysis above and in the attached charts, Viola code bases and references include express teachings that ViolaWWW could interoperate with executable applications, and Pei Wei's demonstration of Viola shows an example of this functionality with the vplot executable application. The videos I am submitting along with this report show other examples, including VIS, MMPEG, and XV. In any case, to the extent one argues that such an express teaching is missing, it would nonetheless have also been obvious to one of ordinary skill in the art to combine the Viola application in each of the Viola prior art references with any

type of executable application, such as Vplot itself, which makes the asserted claims obvious as well. (See, e.g., [vplot-sun] or [vplot-petra].)

## 2. MediaView

554. MediaView prior art includes at least the following publications: [Phillips91c]; [Phillips91b]; [Phillips91a]; [Phillips91d]. Other publications which I authored, including [Phillips86], [Phillips92], [Phillips88], and [Phillips89a], also inform my work in connection with MediaView and were authored by me, and can also be considered MediaView prior art.

555. MediaView prior art also includes the compact disc labeled "LANL Software and Visualization Sampler" [LANL92] from Los Alamos National Laboratory, a publication for which I was editor and which was distributed at the SIGGRAPH'92 conference in Chicago in 1992. At that conference I also gave a live demonstration of MediaView as a speaker in a panel on Digital Publication. The reference [LANL92] was in the possession of and was produced by Dan Sadowski, who attended the session.

556. Another piece of MediaView prior art is a 1991 nomination for an award from The Smithsonian Institution National Museum of American History for the Media, Arts and Entertainment Section of the Computerworld Smithsonian Awards. As part of the award process I gave a live public demonstration of MediaView at the National Museum of American History in Washington in the Fall of 1991. I am enclosing with this report [Smithsonian91a], which were my responses to a series of questions from Smithsonian that I answered in 1991, and [Smithsonian91b], which is a scan of a formal document I received in recognition for my work in connection with MediaView.

557. An additional piece of MediaView prior art is a March 1993 video recording, [LANL93], of a demonstration I performed showing a wide range of MediaView capabilities, including those described above as the "Live Equation" and



“3D Dataset Viewer” functions, including the use of Mathematica and embedded videos.

558. All of these publications are prior art because they were published before the invention by the applicants of the patents-in-suit, and more than one year prior to the date of application of the patents-in-suit.

559. MediaView prior art also includes at least three versions of the software: version 2.1 (on my NeXT), 2.1+ [LANL92], and version 3.0 (on my NeXT). The core architecture of these applications is largely unchanged between these versions for purposes of this report. One difference, however, is that version 3.0 included another executable application, namely RenderMan. MediaView was thus known and used by others in this country before the invention by the applicants of the patents-in-suit, were in public use in this country more than one year prior to the date of application of the patents-in-suit, and were made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed.

560. As I noted above, MediaView was also demonstrated and used publicly including at the Smithsonian Institute in 1991. MediaView was also demonstrated at EDUCOM '89, SIGGRAPH '90, and SIGGRAPH '92, the latter conference being one that Doyle and Ang attended to present the Visual Embryo project through the CAVE. Sadowski, who was working for Macromind (which would later be acquired by Adobe), attended SIGGRAPH '92 and located a copy of the MediaView 2.1 program that was provided through the conference. See Sadowski Tr. 59:18-22 and DS 1 and 1.1. Don Greenberg, who is still at Cornell University, also worked with me on demonstrations of MediaView, as did several graduate student researchers working with him. [Phillips92] In fact, when Greenberg took a sabbatical from Cornell to visit HP Laboratories in Palo Alto, I was invited and gave presentations to staff and visitors at HP Laboratories in both Palo Alto, California in June 1993. I

presented MediaView in June 1991 at HP's facility in Fort Collins, Colorado, where HP was developing high-end work stations. While the June 1993 presentation at HP in Palo Alto was done using a loaned NeXT cube, my wife and I drove the NeXT cube that has been inspected repeatedly by Plaintiffs to Fort Collins in our Isuzu Trooper. A 1991 personal journal entry for this trip is here:

Tues, June 18 or Wed, June 19  
 Drove to Ft. Collins for a  
 H-P meeting →

We sold the Trooper for stability reasons upon our return. These presentations turned into consulting work I did for HP Laboratories to port MediaView to work on HP's workstations.<sup>12</sup> See, e.g., ADBE0196694, ADBE0196689, ADBE0196691, ADBE0196692, which are documents from my archives of this work.

561. MediaView was not a secret or suppressed either, as it was the Los Alamos National Laboratory policy to widely disseminate my work and get the laboratory known for its cutting edge research. Reporters who had heard of my work came to the laboratory and my home to interview me about MediaView, as is reflected in the NeXTWORLD article from the Fall of 1992, the R&D Magazine article published in July 1994 [Studt94], which has an article about MediaView and shows a RenderMan object from it on the cover, [Forslund92] and [ADBE0196688 (LA-UR – 90-2614)]. I demonstrated MediaView widely and without any confidentiality

---

<sup>12</sup> Given the length of time involved, I cannot recall the names of Dr. Greenberg's graduate students, though trying to refresh my recollection I did recall James Ferwerda, who is at Rochester Institute of Technology now, and he refreshed my recollection about Patrick Heynen, who I learned works for Apple. I also recall Joel Birnbaum, who was the executive in charge of HP Laboratories and who was present and spoke during a presentation at HP Laboratory in Palo Alto, California, and Jim Blinn, who was present at a presentation at HP's facility in Fort Collins, Colorado. See, e.g. RLP 10 (materials related to HP presentations and consulting).

restrictions, including to the authors of these articles. MediaView was even cited in text books and surveys on multimedia and distributed computing, including [Umar93] and [Earnshaw96]

562. I also have prepared a series of videos demonstrating MediaView's prior art capabilities. *See* Appendix C, at MV\_overview.wmv, Embedded Hypermedia\_2.wmv, RM1.wmv, RM3.wmv, Mathematica\_Examples.wmv, and Various Examples.wmv. As noted a couple times above, I also recently located a video demonstration of MediaView prepared by Los Alamos National Laboratory in March of 1993. [LANL93].

563. In my opinion, the MediaView prior art (all versions) anticipates claim 1 (and dependent claims 2, 3, and 11), and claim 6 (and dependent claims 7, 8, and 13) of the '906 patent. It is also my opinion that MediaView anticipates claim 1 (and dependent claims 2 and 4-11), claim 16 (and dependent claims 17 and 19), claim 20 (and dependent claims 21 and 23), claim 24 (and dependent claims 25 and 27), claim 28, claim 36 (and dependent claims 37 and 39), and claim 40 (and dependent claims 41 and 43) of the '985 patent.

564. MediaView satisfies all of the '906 and '985 limitations for these claims including those where there is a sub-requirement that an "object is external to a hypermedia document" or "file containing enabling information."<sup>13</sup>

565. For example, and using the numbering scheme of the claim charts appended to this document, in '906-1.f, the limitation is: "wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies the location of at least a portion of an object external to the first distributed hypermedia document."

---

<sup>13</sup> Limitations for or related to "object is external to a hypermedia document" or "file containing enabling information" include 906-1.f; 906-6.f; 985-36.f; 985-40.g; 985-1.f; 985-16.f; 985-20.g; 985-24.b; 985-24.i; 985-28.b; 985-28.i.

566. For that limitation for MediaView, it is stated, MediaView discloses that the embed text format specifies the location of an object. From [Next89], "A Text object or any subclass allows a 'graphic character' to be embedded in the text stream." In MediaView "graphic characters" are subclasses of a ViewCell, whose address is the data contained in the info field described above in connection with Parsing. The ViewCell character directly specifies the processing to be done for the hypermedia component. Thus, if the ViewCell character represents an embedded RenderMan viewer, its type is an instance of the \_3DButton class and its function is performed by the \_3DAction method.

567. The ViewCell character appearing in a MediaView document satisfies every sense of this limitation, in "that it specifies the location of at least a portion of an object external to the first distributed hypermedia document." For example, when a document containing a \_3DButton class object is read by MediaView from a network location, the information contained in the associated ViewCell object points to a *ribShape* object, which is yet to be read in and activated during the image rendering process. That *ribShape* object contains all the 3D coordinate data that was initially read from a RIB file at the time the \_3DButton class object was created.

568. It is further my opinion that MediaView renders obvious claims 3, 18, 22, 26, 30, 38, and 42 of the '985 patent, which relate specifically to HTML.

569. In my invalidity claim chart based on MediaView prior art, appended hereto as Claim Chart Exhibit 7, I have substantiated these opinions of anticipation and obviousness on an element-by-element basis based on the references [Phillips91a], [Phillips91b], and [Phillips91c], but my analysis is also informed by other MediaView prior art, and I reserve the right to opine on anticipation and/or obviousness based on any portion of MediaView prior art.

### 3. MediaView and Mosaic

#### a. Mosaic prior art

570. It is further my opinion that MediaView prior art in combination with Mosaic prior art renders obvious all claims of the patents-in-suit.

571. "Mosaic prior art" includes all versions of Mosaic up through and including version 2.4. Specifically, the named inventors admit in their patent specification that Mosaic version 2.4 is prior art and the code base upon which they based their modification. I understand that several of these codebases have been produced at the following bates numbers, and therefore "Mosaic prior art" also includes these codebases: PA-NAT-00000044 ("Mosaic 2.4"); PA-NAT-00000045 and PA-NAT-00000046 ("Mosaic 2 prerelease 5"); PA-NAT-00000047 ("Mosaic 2 prerelease 1"); PA-NAT-00000048 ("Mosaic 2 prerelease 2"); PA-NAT-00000049 ("Mosaic 2 prerelease 3"); PA-NAT-00000050 ("Mosaic 2 prerelease 4"); PA-NAT-00000051 ("Mosaic 1.2"); PA-NAT-00000052 ("Dandelion Patch "); PA-NAT-00000054 ("DX227 Mosaic 2.4").

572. The codebase Mosaic 2 prerelease 5 was dated October 10, 1993. Accordingly, it was known or used by others in this country before the invention by the applicants of the patents-in-suit, was in public use in this country more than one year prior to the date of the application of the patents-in-suit, and was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed.

573. Mosaic codebases were widely released and posted to FTP locations (*see* by way of example [Andreessen93a]). This includes Mosaic 2 prerelease 5, for which publically released documentation and release information can be found at <http://dandelionpatch.mit.edu/afs/sipb.mit.edu/project/www/doc/Mosaic-2.1/help-on-version-2.0.html> and

<http://dandelionpatch.mit.edu/afs/sipb.mit.edu/project/www/doc/Mosaic-2.1>,

available at [PA-NAT-00000052]. The public announcement of Mosaic 2 prerelease 5 can be seen at [www-talk-00292656]. Accordingly, Mosaic 2 prerelease 5 was published before the invention by the applicants of the patents-in-suit and was a printed publication more than one year prior to the date of application of the patents-in-suit.

574. For the same reasons as described above, earlier versions of Mosaic were also prior art. These include Mosaic 2 prerelease 1, Mosaic 2 prerelease 2, Mosaic 2 prerelease 3, Mosaic 2 prerelease 4, and Mosaic 1.2, all identified above by bates number.

575. Mosaic version 2.4 is prior art because, as explained above, the named inventors acknowledge that it is prior art the '906 patent specification. Also, it was announced and publically posted and released on April 11, 1994, which can be seen at a www-talk posting by Eric Bina on that date (<http://www.intercom.co.cr/www-archives/1994-q2/0162.html>). Accordingly, Mosaic 2.4 was known or used by others in this country before the invention by the applicants of the patents-in-suit, was in public use in this country more than one year prior to the date of the application of the patents-in-suit, was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed, was published before the invention by the applicants of the patents-in-suit, and was a printed publication more than one year prior to the date of application of the patents-in-suit.

576. "Mosaic prior art" also includes documentation about Mosaic, including [Andreessen93a], [Andreessen93b], and [Collage92]. Each of these references corroborates that the software systems described were known or used by others in this country before the invention by the applicants of the patents-in-suit, were in public use in this country more than one year prior to the date of application of the patents-in-suit, and were made in this country before the invention by the applicants

for the patents-in-suit and was not abandoned, suppressed, or concealed. Further, each of these references was published before the invention by the applicants of the patents-in-suit, and was a printed publication more than one year prior to the date of application of the patents-in-suit.

**b. Obviousness based on MediaView and Mosaic and other web browsers**

577. In my opinion, MediaView prior art and Mosaic prior art render obvious all claims of the patents-in-suit.

578. I have constructed claim charts for all of the asserted claims for the prior art references represented by MediaView (*see* Claim Chart Exhibit 7) and Mosaic (*see* Claim Chart Exhibit 8). Claim Chart Exhibit 8 for Mosaic focuses on the references [Andreessen93a], [Andreessen93b], [Collage92], my personal experience having used Mosaic, and Mosaic software releases. However, my analysis is also informed by other Mosaic prior art, and I reserve the right to opine on anticipation and/or obviousness based on any portion of Mosaic prior art.

579. I have found that both references satisfy all or nearly all of the limitations of the '906 and '985 patents. With detailed citation to the claim charts, I will discuss where each reference could be combined with another reference that is clearly in satisfaction or the other of the references plays a complementary role in the combined satisfaction.

580. Several limitations in the '985 patent refer to the requirement that "the text formats are HTML tags."<sup>14</sup> The MediaView documents were implemented using an alternative to HTML tags, which was a form of rich text format tags. Thus, strictly speaking, these limitations are not satisfied. However, the actual language used to markup the page was merely a design choice by a person of ordinary skill in the art

---

<sup>14</sup> This can be found in at least the claim limitations 985-3.a; 985-18.a; 985-22.a; 985-26.a; 985-38.a; 985-42.a.

at the time. I note, for example, that in [Phillips91c] it is expressly noted at pg. 82 that "The most obvious and important enhancement is a hyperlinking capability. This has been designed and will be implemented in the next few months. Its design draws upon the rich NeXT development environment, in particular the suite of BTree classes that are available." *See also* [Phillips91b] at pg. 12. HTML is a form of hypertext (hence its name, Hypertext Markup Language). Thus, it would have been obvious to a person of ordinary skill in the art to implement hyperlinks in the MediaView using HTML tags. Furthermore, in this report I have pointed out that while at CERN, Tim Berners-Lee implemented his web browser/editor using NeXTSTEP by sub-classing the Text class to form the HyperText class. That browser parsed HTML. I believe the above arguments render the HTML limitations obvious.

581. Additionally, CERN maintained a page of systems and applications related to the World Wide Web and specifically noted the MediaView system. [W3C-overview]. A person of ordinary skill in the art would have known, from the CERN page, that MediaView was a form of browser, and that it could have been implemented using HTML tags as opposed to the more general purpose tags which were used. Furthermore, Mosaic, as is discussed below, was implemented as a web browser that parsed HTML tags, but a person of ordinary skill would know it could have been modified to incorporate the same inline tagging found in MediaView to create plugins or custom components without browser bloat.

582. All of the limitations listed above are completely satisfied by MediaView and/or MediaView in combination with Mosaic.

583. Similarly, I now discuss how MediaView provides all limitations that are arguably missing from Mosaic.

584. Mosaic satisfies all of the '906 and '985 limitations except for those where there is a sub-requirement that there is interaction with the object at a "first



location"<sup>15</sup> in the hypermedia document, and that there be "automatic invocation of executable application" or that "automatic invocation does not require interactive action by the user."<sup>16</sup>

585. For example, from '906-1.h, the limitation is: "wherein said embed text format is parsed by said browser to automatically invoke said executable application to execute on said client workstation in order to display said object and enable an end-user to directly interact with said object within a display area created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window."

586. For that limitation for Mosaic, it is stated, from [NCSA92], "NCSA Mosaic initially supports the X bitmap (XBM) and GIF image formats directly (an example can be seen in Figure 5) and provides interfaces to external viewers to handle other multimedia data formats (e.g. JPEG, XWD, TIFF, RGB, MPEG, DVI, PostScript, and several types of audio).<sup>1</sup> Mosaic parses a file to discover tags indicating these media types and invokes appropriate external viewers. Media of type XBM and GIF are embedded inline by the HTML img tag, a text format."

587. Other embed text formats point to hypermedia objects that are external to the browser file and that cause the invocation of an external helper program. The MIME type of the object is used to locate an appropriate executable application. Helper applications display the hypermedia object and enable direct interaction with the hypermedia object.<sup>17</sup>

---

<sup>15</sup> Limitations related to interaction at a "first location" can be found in at least 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i.

<sup>16</sup> Limitations related to "automatic invocation of executable application" or "automatic invocation does not require interactive action by the user" can be found in at least 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.l; 985-28.l; 985-36.h; 985-40.i; 985-11.a.

<sup>17</sup> To the extent that Eolas incorrectly contends that the asserted claims cover content in a separate or "pop up" window launched from a browser, this would make the claims invalid as anticipated and obvious based on at least the Mosaic prior art, which as I describe here and also admitted by the

588. Further, interaction with the hypermedia object is achieved through the helper application control panel and its window and no results appear at the first location in the hypermedia document.

589. I also note the "automatic invocation" sub-requirements.<sup>18</sup> For example, from '906-1.h the limitation is: "wherein said embed text format is parsed by said browser to automatically invoke said executable application to execute on said client workstation in order to display said object and enable an end-user to directly interact with said object within a display area created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window."

590. For that limitation for Mosaic, it is stated: "From [Collage92], "NCSA Mosaic initially supports the X bitmap (XBM) and GIF image formats directly (an example can be seen in Figure 5) and provides interfaces to external viewers to handle other multimedia data formats (e.g. JPEG, XWD, TIFF, RGB, MPEG, DVI, PostScript, and several types of audio).<sup>1</sup> Mosaic parses a file to discover tags indicating these media types and invokes appropriate external viewers. Media of type XBM and GIF are embedded inline by the HTML img tag, a text format." Other embed text formats point to hypermedia objects that are external to the browser file and that cause the invocation of an external helper program. The MIME type of the object is used to locate an appropriate executable application.

591. Further, helper applications display the hypermedia object and are invoked by the user, not automatically.

---

named inventors in the specification and prosecution history of the patents-in-suit, discloses displaying interactive objects in a separate window.

<sup>18</sup> Limitations related to "automatic invocation of executable application" or "automatic invocation does not require interactive action by the user" can be found in at least 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.l; 985-28.l; 985-36.h; 985-40.i; 985-11.a.

592. All of the limitations listed above, which are not completely satisfied by Mosaic, are satisfied by MediaView.

593. It would have been obvious to one of ordinary skill in the art at the time of the alleged invention to combine MediaView and Mosaic. First of all they were contemporaneous developments, with early versions of Mosaic appearing in the 1992 timeframe shortly after 1991 MediaView publications. Furthermore, in 1991-1992, Tim Berners-Lee at CERN developed a browser-editor on a NeXT workstation by developing a Hypertext subclass of the Text class provided by the NeXT development system. In fact, Berners-Lee expressed interest in obtaining MediaView source code to produce an enhanced browser.

594. Moreover, the World Wide Web consortium archived a CERN web site that was maintained by Tim Berners-Lee. That site address is:

<http://www.w3.org/History/19921103-hypertext/hypertext>. A link on that site called "Products" points to a page with the title "Systems and Applications." An introductory paragraph states: "Here is a list of existing systems, past present and future, which we have come across. These include both academic research systems and commercially available systems." One entry on that list is MediaView, with pointers to *Note from Dick Phillips* and *README*. The note in turn pointed to a Purdue University Internet repository where a MediaView application and document package could be obtained. The motive for Berners-Lee to maintain such a list was to bring as much information together as possible, all to make the web grow. His last line on the list was "Thus spreads the web..."

595. Finally, it was in fact the case that even at least one of the inventors and his co-worker appears to have done just this – examined MediaView in the context of Hypertext systems, and informed co-inventor Martin to do the same. [McRae93], *see also* McRae Tr. 80:6-82:5. Thus, the combination of MediaView with hypertext technology, and particularly a web browser, was clearly suggested by the prior art.

596. In this regard, namely the combination of MediaView with any web browser, and in view of the testimony by the inventors suggesting that the developers of Mosaic had access to the invention based on a presentation by the inventors (see, e.g., 8/11/11 Doyle Tr. beginning at 361 and Martin Tr. beginning at 482) it would have also been obvious to combine the functions and features of MediaView with the functions and features of Viola, discussed throughout this report and accompanying exhibits, as well as Tim Berners-Lee's CERN World Wide Web Browser, whether the first version that did not support inline images or the later 2.0 version, which did (see Figure 23 below). I reserve the right to supplement my report with regard to the CERN browser and the combination with MediaView with testimony from Mr. Berners-Lee once he has testified.

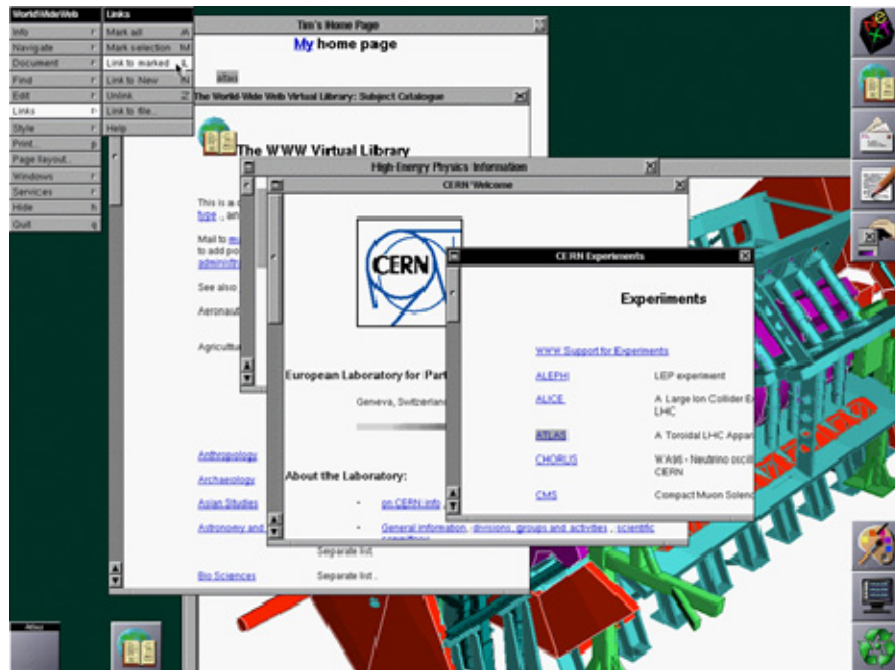


Figure 23. CERN Browser/Editor Version 2

597. Like MediaView, the CERN browser was written using NeXTSTEP which, as Berners-Lee notes, “I could do in a couple of months what would take more like a year on other platforms.” Berners-Lee added hypertext to the project by sub-classing the NeXTSTEP Text class to create a HyperText class.

598. It would have been straightforward to combine MediaView with the WorldWideWeb browser to create a highly synergistic new application. MediaView already had the capability to parse an “open document” request by examining the document’s suffix to determine which method to invoke. For a \*.mdvw suffix MediaView would invoke a method to read and render a saved MediaView file. Also, for a \*.cmbr (corpus member) suffix a different method would be invoked. Finally, for files with \*.rtf and \*.wn suffixes MediaView would read and render Rich Text Format and WriteNow (a NeXT-developed word processor) documents.

599. Thus, for a document with an \*.html suffix, MediaView would invoke an object of Berners-Lee’sHyperText class to request retrieval of that document from its URL-specified location. Then, the parsed information received from the HyperText object would be rendered on the display by a MediaViewText subclass of a Text object. Any embedded (inline) graphical entity discovered during parsing would be represented and displayed by creating a MediaViewCell class object that is inserted in the corresponding location in the text stream.

600. More importantly, since MediaView already had the ability to embed interactive applications in the midst of a document, if an <EMBED>(some application)</EMBED> HTML+ style tag were encountered during HyperText parsing, an instance of that application would be associated with a MediaViewCell and inserted in the corresponding location in the text stream. Whether to manually or automatically launch the application is simply a matter of design choice, but MediaView already automatically launched its custom components (or executable applications).

601. If the application type was not a priori known by MediaView, the already-existing custom component capability could be used to create a library of custom components that would be available for automatic installation in the document, just as, for example, images were invoked automatically upon parsing of an HTML document in both the (later) CERN and Mosaic web browsers using handling code for those processes that had been developed by others. (See, e.g., Bina Tr. at 33:8-35:6; McRae Exhibit 37. In the 1993 time period, for example, it would have been straightforward to create custom components for xv, vis, xplot, and mpeg, among others, together with the similar custom components that already existed for MediaView, including the Mathematica, 3D viewer, and the video and audio custom components.

602. A text stream that represents a hyperlink would be represented as a NeXTSTEP Button subclass object and be displayed in a distinctive color. Clicking it would invoke another instance of a HyperText object, which when processed, would create a separate MediaView document but would be displayed in the same browser window. The original (clicked) document is still easily accessible because of the scrolling summary view that was part of the original MediaView design.

603. Moreover, as I noted in my opening report, I was contacted by Berners-Lee concerning sharing the source code for the MediaView system. [7/20/2011 Report on Invalidity at ¶ 268] Berners-Lee posted a short description and hosted notes about MediaView on the CERN website he maintained [Martin Exs. 14, 15, 16], and this description was accessed by people, including the inventors on the patent, when looking for information about web browser development. [Martin Ex. 20 and McRae Ex. 26]. There was not only a teaching, suggestion and motivation to combine the CERN and Mosaic browsers with MediaView features, such as inline embedded objects and distributed applications, but there is empirical evidence that the inventors themselves accessed this information.

604. It is therefore my opinion that the above-mentioned combination of the prior art references MediaView and Mosaic satisfies all of the limitations of all asserted claims of the '906 and '985 patents, and therefore render them obvious.

#### **4. Mosaic, HTML+, and disclosure and testimony of Bill Janssen**

##### **a. HTML+ and disclosure and testimony of Bill Janssen as prior art**

605. It is further my opinion that Mosaic prior art in combination with what was generally known to persons having ordinary skill in the art, as demonstrated through postings to the publically available www-talk forum, renders obvious all claims of the patents-in-suit.

606. I have constructed claim charts for all of the asserted claims for the prior art references represented by Mosaic prior art. (Claim Chart Exhibit 8.) While Mosaic does not disclose all claim elements of the asserted claims, it is my opinion that the asserted claims would have been obvious to a person having ordinary skill when viewed in combination with prior art www-talk postings, including postings related to the HTML+ specification.

607. The HTML+ specification to which I refer is [Ragget93a]. The postings to which I refer include certain exhibits to the deposition of Bill Janssen, taken on May 11, 2011, including Exhibit 6 [PH\_001\_0000598210], Exhibit 8 [PH\_001\_0000598248], Exhibit 9 [PA-00306624], and Exhibit 10 [PH\_001\_0000588858], which were all public postings from mid-1993. Accordingly, these references were published before the invention by the applicants of the patents-in-suit. These references show that the subject matter discussed in them were known or used by at least Dave Raggett, Bill Janssen, and other members of the www-talk distribution list in this country prior to the invention by the applicants of the patents-in-suit. In addition, these references were published more than one year prior to the date of application of the patents-in-suit. This reference corroborates that the software methods disclosed in them were in public use in this country more than one year

prior to the date of application of the patents-in-suit. Furthermore, this reference corroborates that before the invention by the applicants for the patents-in-suit, at least some of the programming techniques discussed were made in this country by individuals who did not abandon, suppress, or conceal them.

608. Finally, the deposition of Bill Janssen and its exhibits (including Exhibit 11) demonstrate what was known or used by others in this country before the invention by the applicants of the patents-in-suit, what was in public use in this country more than one year prior to the date of application of the patents-in-suit, and at least some of his testimony indicates what was made in this country before the invention by the applicants for the patents-in-suit without being abandoned, suppressed, or concealed.

609. The U.S. Patent and Trademark Office considered a portion of these materials, and the Examiner found that they did not anticipate claims of the asserted patents because he "can find no evidence within the references of Janssen and Raggett that specifically teach of 'automatically invoking the executable application to execute on said client workstation in order to display said object.'"

(PH\_001\_0000786952). I note that, at the time of the '858 Reexamination Proceeding, the Examiner did not have the benefit of reviewing testimony of Mr. Janssen that has been given in connection with this litigation. I have reviewed this testimony, and have considered it in forming my opinions offered in this report. I explain in my claim chart why these materials do, in fact, render obvious the claims of the asserted patents. I also explain why under the system contemplated by these prior art references that invocation of the executable application would be "automatic."

**b. Obviousness based on these references**

610. The primary differences between the claimed invention and the Mosaic prior art is that Mosaic did not display interactive objects inline, and Mosaic did not automatically invoke helper applications – i.e., those where there is a sub-



requirement that there is interaction with the object is at a "first location"<sup>19</sup> in the hypermedia document, or those requiring that there be "automatic invocation of executable application," or "automatic invocation does not require interactive action by the user."<sup>20</sup>

611. It would have been obvious to configure Mosaic to overcome both of these differences in view of the general state of the art.

612. As to displaying interactive objects inline (interaction with the object is at a "first location"<sup>21</sup>), for Mosaic prior art, only media of type XBM and GIF are embedded inline. This was by the HTML img tag, at the first location in the hypermedia document. However, it was obvious and widely known to persons of ordinary skill at the time how to cause the object to be displayed at the first location in a hypermedia document. There were numerous posts to the www-talk interest group in mid-1993 about the subject. The issue was often referred to as "inlining" or "embedding" and the EMBED tag was proposed in the HTML+ standard to handle that situation. That means that when the presence of a hypermedia object was discovered during parsing, its representation was displayed in the browser window at the page position where it was parsed. Among many possibilities, the hypermedia object could be video, mathematical equations and running applications.

613. This issue was recently discussed at length during the Janssen Deposition. Janssen was one of the frequent contributors to www-talk. The deposition text and the referenced exhibits clearly show that persons of ordinary skill

---

<sup>19</sup> Limitations related to interaction at a "first location" can be found in at least 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i.

<sup>20</sup> Limitations related to "automatic invocation of executable application" or "automatic invocation does not require interactive action by the user" can be found in at least 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.l; 985-28.l; 985-36.h; 985-40.i; 985-11.a.

<sup>21</sup> Limitations related to interaction at a "first location" can be found in at least 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i.

in the 1993 time frame had a firm grasp on the principles of UNIX interprocess communication and embedded X-Window widgets that made it obvious to modify Mosaic to satisfy the claimed limitations related to inline embedding.

614. An early Q and A exchange during that deposition establishes the role and openness of the forum follows:

Q. In 1993 what was www-talk?

A. Well, my recollection is that in 1993, maybe even earlier, maybe in -- even in 1992, there was a mailing list about the emerging World Wide Web, which had become a major topic. It had a number of people from around the world. I was on that mailing list. I regularly read the mail and contributed to it. And it was basically for discussion of the Web and various topics, like HTTP and HTML and ways in which it might change or grow or evolve.

Q. Were Web browsers discussed on www-talk as well?

A. I'm sure they were, yes.

Q. Okay. In 1993 was the existence of www-talk a secret?

A. Not at all.

\*\*\*\*\*

Q. Okay. If you could turn to Janssen Exhibit 6. I want to read some of what you wrote to www-talk on April 27th, 1993, in response to Mark Andreessen's comments, and then I'll have some questions for you. This is toward the bottom of Janssen Exhibit 6. The second paragraph from the bottom, starting at line 3. You wrote, "HTML should allow embedded animations (very useful in documentation of gestural interfaces, for one thing) or graphs or even other text. I'm basing my expectation of the power and utility of this mechanism on my experience with Andrew and Slate, both of which have this feature. Typically what is shown for embedded sound, for example, is a small (tiny) control panel which allows control over the speed and amount of the sound inset that is actually played, along with a label." Did I read that accurately?

A. You did.

Q. So what were you communicating back to Mark Andreessen via this post to www-talk on April 27th, 1993?

A. I was communicating back my hypothesis that he was trying to avoid work for his team back in Mosaic by not allowing more complex features than the HTML standard, and that since we had wide exposure to other toolkits which

allowed embedded applications to exist inside the text flow, for example, toolkits like Andrew, for example, from the late '80's, that it was certainly technically feasible to do that, and that there was no particularly good reason to leave it out of the HTML standard because the hypertext systems of the day were already incorporating features like this. And if we didn't put it into the World Wide Web, the World Wide Web would be seen as an inferior hypertext system.

Q. Would it be fair to say on April 27th, 1993, you and Mark Andreessen were continuing a public discussion on www-talk about embedding external data objects in a Web page?

A. I would certainly characterize it as such.

\*\*\*\*\*

Q. So in 1993 X11 was the windowing system for workstations?

A. A very widespread, widely used windowing system, yes.

Q. And using X11, if I understood your explanation of what was being discussed on www-talk on April 29th, 1993, you could have the browser pass the window ID to XV, and then XV would allow a user to manipulate an image directly with inline of the Web page?

A. That's a pretty good summary, yeah.

Q. In 1993 did you believe that having browsers and external viewers cooperate with each other was an easy project?

A. I'd call it straightforward, because I'd certainly already done it several times in different kinds of browsers. And I knew other people who had done it. For example, the Andrew project at CMU, the Slate project referred to apparently did it, although I don't remember the Slate project. So yes. I would say straightforward, not easy.

\*\*\*\*\*

615. Next there was a question about the contents of a www-talk posting by Janssen on June 14th 1993 directed to possible uses of an EMBED tag that was proposed by Dave Raggett. The posting is:

" Yes, that sounds good. My favorite way to handle this is to have the browser create and manage an X sub-window over the area where the inset is to be displayed, and pass the window ID of the sub-window to the subprogram which understands the inset format, with the understanding that that program is to handle all events and refresh on the sub-window; but the browser gets to handle configuration and window movement."

616. Then the deposition continues with:

Q. All right. Is what you wrote to www-talk on June 14th, 1993, as shown in Janssen Exhibit 10 and illustrated in Janssen Exhibit 11 - could program A in your example be a Web browser, and could program B in your example be a separate imaging program such as XV?

A. Yes.

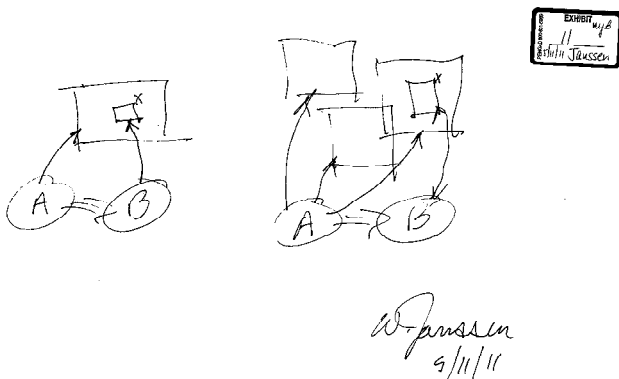
\*\*\*\*\*

Q. Let me try again. Does the combination of a www-talk postings on June 14th, 1993, and the HTML+ standard dated July 23rd, 1993, describe a technique of how a browser can parse an EMBED tag and then automatically invoke a different stand-alone program to display an image in a sub window that appears within the larger browser window and enables an end user to interact with that image at the same location as where the EMBED tag was in the HTML file?

A. In my opinion, yes

\*\*\*\*\*

617. The illustration in Janssen Exhibit 11 is reproduced below.



618. The concepts described above were well-known to persons of ordinary skill in the art. For example, from [www-talk-00293128], a public email exchange between named inventor Michael

Doyle and Viola developer Pei Wei from August 31, 1994, Wei teaches exactly how to cause an external binary executable to produce graphical or textual output in a sub-window embedded in a browser window. He states:

"Well. Viola's model was \*demonstrated\* in 1993, \*released\* freely in 1994. . . . And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point

is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it."

619. When Wei says "fires up a back-end," one of ordinary skill in the relevant era would know immediately to use the UNIX routines `vfork`, to spawn a new process, and `execv`, to launch the executable and pass it an X window ID to use for output.

620. As to Mosaic's other difference – that Mosaic did not automatically invoke helper applications ("automatic invocation of executable application," and "automatic invocation does not require interactive action by the user."<sup>22</sup>); automatic invocation also would have been obvious to a person of ordinary skill in the art, as shown by the discussion above.

621. Prior art Mosaic 2.4, helper applications display the hypermedia object and the applications are invoked by the user, not automatically. However, it was obvious and widely known to persons of ordinary skill at the time how to automatically initiate invocation of an executable application. The earlier discussion of inline embedding involved a determination of the type of external application to be invoked and the location of an external dataset to be accessed. The default invocation would normally be automatic.

622. It would have been obvious to combine Mosaic with the HTML+ disclosure from [Raggett93a], because the HTML+ disclosure was proposing supplements to the HTML standard for use with browsers like Mosaic. In fact,

---

<sup>22</sup> Limitations related to "automatic invocation of executable application" or "automatic invocation does not require interactive action by the user" can be found in at least 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.l; 985-28.l; 985-36.h; 985-40.i; 985-11.a.

Mosaic was already implementing tags from HTML+. Earlier, I described that with the release of Mosaic 2.0 for X Windows, [Andreessen93a], it became possible to embed Motif widgets inside Mosaic documents. This capability arose due to partial implementation of the HTML+ specification, discussed above. In particular it was pointed out that documents could now specify interactive fill-out forms – with input elements including text entry areas, toggle buttons, selection lists, popup menus, etc. –and Mosaic will instantiate such fill-out forms as sets of Motif widgets embedded inside the document. Expanding Mosaic's support of HTML+ to other tags, such as the EMBED tag, would have been obvious.

623. It also would have been obvious to combine the www-talk postings set from in the Janssen Dep. Exhibits, including Exhibit 6 [PH\_001\_0000598210], Exhibit 8 [PH\_001\_0000598248], Exhibit 9, and Exhibit 10 [PH\_001\_0000588858], with Mosaic. First, these www-talk email threads were focused on additions to the HTML standard that was used by web browsers like Mosaic. Second, Marc Andreessen, a primary developer of Mosaic, was actually involved, contributed to, and read these www-talk email threads. Third, these email exchanges describe X window concepts, and Mosaic operated in an X environment. Fourth, Bill Janssen's deposition testimony shows that persons of ordinary skill in the art would have known to combine the ideas from these emails with Mosaic.

## **5. Mosaic in combination with Chris McRae's posting to www-talk**

### **a. McRae's posting as prior art**

624. The reference [www-talk-00293020] was published to the www-talk forum on June 26, 1993. Accordingly, it (and the subject matter it disclosed) was known or used by others in this country before the invention by the applicants of the patents-in-suit, was published before the invention by the applicants of the patents-in-suit, was in public use in this country more than one year prior to the date of

application of the patents-in-suit, and was a printed publication more than one year prior to the date of application of the patents-in-suit.

625. In addition, this posting provides evidence that the named inventors did not themselves invent the subject matter sought to be patented, because it provides evidence that Chris McRae conceived of the invention.

**b. Obviousness based on these references**

626. "Mosaic prior art" and Chris McRae's June 26, 1993 posting render obvious all claims of the patents-in-suit. The motivation to combine the references may be found, among other places, in Bina Tr. and Exs. 4 and 7; McRae Tr. and Ex. 37 (messages from April-June 1993); Martin Tr. including 425:13-426:7 and 532:10-22

627. As I explained above, in an email thread directed to Mosaic, Chris McRae [www-talk-00293020], disclosed a scenario involving inlining xv. He asks Andreessen, "Suppose someone (such as ourselves) were able to integrate tootalk functionality into mosaic, \*and\* could provide an HDF- and tootalk-capable version of xv (or some such image manipulation package). Would the technology fit into your development strategy for mosaic?" Such a proposition clearly presaged a scenario where, within the purview of a Mosaic browser, one had an embedded interactive object managed by xv, ToolTalk interprocess communication, and access to HDF formatted data in a distributed processing environment.

628. The XV program being discussed is described in [Bradley92]. The description of the program states, "XV is an interactive image manipulation program for the X Window System. It can operate on images in the GIF, JPEG, PBM, PGM, PPM, X11 bitmap, Sun Rasterfile, and PM formats on 1-, 4-, 6-, 8-, 16-, 24-, and 32-bit X displays". Thus, after an inline invocation of XV, the control window shown Figure 12 would appear outside the browser window but graphical output would appear within.

629. Also as I described above, the ToolTalk utility being discussed is described in [Sun91] and is an interapplication communications system developed by Sun Microsystems (SunSoft) in order to allow applications to communicate with each other at runtime. Applications supporting ToolTalk can construct "high-level" messages and hand them off to the system's ToolTalk server, which determines the proper recipients and (after applying permission checks) forwards the message to them. The ToolTalk service enables independent applications to communicate with each other without having direct knowledge of each other. Applications create and send ToolTalk messages to communicate with each other. The ToolTalk service receives these messages, determines the recipients, and then delivers the messages to the appropriate applications.

630. Also as I described above, it was well known that HDF formatted data in a distributed processing environment could be provided by Collage, as described in [Collage92]. Among Collage's many features is the ability to establish communication with remote processes, e.g. a simulation running on a supercomputer. These remote processes can be controlled remotely, and images and data can be transported to and from the remote process. Moreover, one can perform most of these operations not only on one machine, but on any machine that is participating in a collaborative session with NCSA Collage. Consequently, collaborators using Mosaic clients and involved in a Collage session can, for example, open and view an HDF (Hierarchical Data Format) file that was produced by a supercomputer computation.

631. In Claim Chart Exhibit 10, I explain on an element-by-element basis why Mosaic and Chris McRae's posting renders obvious all asserted claims of the patents in suit.



## 6. Mosaic in combination with Adobe PDF related postings to www-talk.

### a. Adobe PDF is prior art

632. As I noted above with particular www-talk postings by McRae, I understand that McRae produced several emails published by others to the www-talk mailing list more than a year before the patents were filed. Accordingly, these messages (and the subject matter they disclosed) was known or used by others in this country before the invention by the applicants of the patents-in-suit, was published before the invention by the applicants of the patents-in-suit, was in public use in this country more than one year prior to the date of application of the patents-in-suit, and was a printed publication more than one year prior to the date of application of the patents-in-suit.

633. Eolas's Local Patent Rule 3-2 Infringement Contentions accused Adobe Acrobat of infringement. While Eolas now asserts that it has withdrawn these infringement contentions, I will use them and the materials cited below to show that the identity of the features of modern versions of Acrobat are the same as the identity of the features of the 1993 version of Acrobat. *See, e.g.,* 906 – Adobe – PDF – Authoring Tools and Players (Final).pdf at 1 and 105; 985 – Adobe – PDF – Authoring Tools and Players (Final).pdf at 1 and 183. The functionality alleged to infringe is virtually every feature of Adobe's Acrobat Reader. (*See, e.g.,* 906 – Adobe – PDF – Authoring Tools and Players (Final).pdf at 51 and 61 (61 shown below).)



634. These same features were available in Adobe's Acrobat software more than a year before the patents were filed. As such, if it infringes now, as Eolas alleges, it infringed prior to October 17, 1993 and it is 102(b) prior art. Below is a screen capture from Adobe's Acrobat Reader user manual from prior to October 17, 1993.



635. The zoom and page navigation controls were present in 1993, as were the thumbnails and other navigation and data manipulation options. (ADBE0196072-113; *see also* [Adobe93].)

636. In fact, over the course of discovery, I understand that that it was found that the inventors themselves signed a non-disclosure agreement with Adobe in August of 1993, suggesting that Acrobat could be 102(f) prior art, meaning the inventors derived their invention from Adobe and the www-talk group. (*See* ADBE0195776-777.) The inventors and individuals who worked with them at UCSF remarked on the Adobe Acrobat and PDF software to the www-talk group. (*See* 1993-06-21 0920 email from McRae; 1993-07-20 1314 email from Martin). Compare, for example, [Adobe93] at 79 to the EMBED tag disclosed in the patents: they are remarkably similar. *See also* Bina Tr. and Exs. 4 and 7; McRae Tr. and Ex. 37 (messages from April-June 1993); Martin Tr. including 425:13-426:7 and 532:10-22 (additional motivation to combine).

**b. Obviousness based on Mosaic with Acrobat and PDF www-talk postings**

637. Beyond the inventors' knowledge of Adobe's Acrobat and PDF standards, upon Adobe's announcement of the alleged infringing technology, the www-talk group was abuzz with discussions of incorporating the functionality into web browsers, such as Mosaic, prior to October 17, 1993. (*See* 1993-06-21 0920 email from McRae; 1993-07-16 0950 email Cailliau; 1993-07-16 1812 email from Kehoe; 1993-07-19 1309 email from Kehoe; 1993-07-19 1107 email from Altis; 1993-07-19 1552 email from Janssen; 1993-07-19 1355 email from Altis; 1993-07-19 1614 email from Kehoe; 1993-07-20 1848 email from Heaney.) Even the inventors noted the strong interest in the technology. (*See* 1993-07-19 0855 email from Martin.)

638. As is noted elsewhere in my report too, the Adobe book referenced for example in Dr. Martin's July 19, 1993 email copying co-inventors Doyle and Ang, [Adobe93], the ability to add a special tag to a document with text formats was known. [Adobe93] at 79. It was also known that it was also known that the special tag could refer to another data object, such as an image, sound, or a movie. [Adobe93] at 78-79.

639. It would have been obvious in its own right to include a special tag like that disclosed in [Adobe93] in the Mosaic browser, given the interest in the community of making such mixed-media compound documents, as well it would have been obvious to a person of ordinary skill in the art at the time of the invention to modify the Mosaic browser as discussed in the www-talk publications to include the Adobe Acrobat functionality into the browser using a helper, DLL or other external executable application, as is discussed in the www-talk mailings. From a reading of the postings, it is clear there were two choices for implementing this idea: (1) incorporating the processing of the PDF file directly into the browser, or (2) using a separate executable application to process the PDF file. As the postings suggest, it was not as advantageous to do (1) because the web browser would have to be modified every time a new file type was added to the web, and the application for handling the processing already existed. Reusing the existing application was a natural design choice for a person of ordinary skill in the art, and techniques for making APIs and drivers for interprocess (and intra process) communications were already known, as is detailed elsewhere in my report. (See, e.g., discussion of MediaView.) The issue distills down to whether the PDF file (for example) will be displayed in the browser window or in a separate browser window, which is hardly the type technological advancement beyond the level of ordinary skill in the art at the time, particularly in view of operation of the prior art browser, OLE technology, and

the HyperCard technology described in my report. (See also, e.g., 6/15/11 Raggett Depo. at 21:18-29:24 (rough) Exs. 4-8.)

640. Furthermore, Eolas's now abandoned Patent Rule 3-2 infringement contentions are themselves evidence of the obviousness of the patent claims in view of Acrobat as well as in combination with the known environment in which individuals contributing to the www-talk discussion and persons of ordinary skill in the art, which is evidenced by Bina Exhibits 4 and 7, as well as the April 1993 discussion in McRae Exhibit 37. In my opinion, this combination also renders the claims obvious and my analysis of the dependent claims would be the same as it is for my charts concerning Mosaic.

## **7. HyperCard (including Director and Quicktime) and Viola**

### **a. HyperCard as prior art**

641. HyperCard software version 2.0 was released by 1990, version 2.1 was released by 1991, and version 2.2 was distributed by December 1993. Corroboration of this can be found, by way of example, in contemporaneous trade publications such as InfoWorld, such as those found in the bates range PA-00321271 through PA-00321298, or by inspection of dates listed in the software.

642. Accordingly, all these versions (and any earlier versions) were known or used by others in this country before the invention by the applicants of the patents-in-suit.

643. Also, at least HyperCard versions up through and included HyperCard 2.1 were in public use in this country more than one year prior to the date of application of the patents-in-suit, because at least these versions were widely distributed by Apple Computer and/or its subsidiaries by 1991, and used by customers of the product.

644. Also, these software versions corroborate that before the invention by the applicants for the patents-in-suit, HyperCard was made in this country by at least

Apple Computer and/or its subsidiaries, who did not abandon, suppress, or conceal it.

645. The reference [Goodman90] is prior art and corroborates what was in the prior art. This reference was a printed publication before the invention by the applicants of the patents-in-suit. In addition, this reference corroborates that Hypercard software including Hypercard 2.0 was known or used by others in this country before the invention by the applicants of the patents-in-suit. Further, this reference corroborates that Hypercard was in public use and on sale in this country more than one year prior to the date of application of the patents-in-suit and it was a printed publication more than one year prior to the date of application of the patents-in-suit. Finally, this reference corroborates that before the invention by the applicants for the patents-in-suit, Hypercard and programs implemented in Hypercard were made in this country by individuals who did not abandon, suppress, or conceal it.

646. [Morgan92] was a printed publication before the invention by the applicants of the patents-in-suit. In addition, this reference corroborates that Hypercard and the TCP XCMD Toolkit software were known or used by others in this country before the invention by the applicants of the patents-in-suit, that Hypercard and the TCP XCMD Toolkit software were in public use in this country more than one year prior to the date of application of the patents-in-suit, and that Hypercard and the TCP XCMD Toolkit software were made in this country by individuals who did not abandon, suppress, or conceal it. This reference was a printed publication more than one year prior to the date of application of the patents-in-suit.

647. [Powers90] was a printed publication before the invention by the applicants of the patents-in-suit. In addition, this reference corroborates that Hypercard and the XCMDs described therein were known or used by others in this

country before the invention by the applicants of the patents-in-suit. This reference corroborates that Hypercard and the XCMDs described therein were in public use in this country more than one year prior to the date of application of the patents-in-suit. This reference was a printed publication more than one year prior to the date of application of the patents-in-suit. This reference corroborates that before the invention by the applicants for the patents-in-suit, Hypercard and the XCMDs described therein were made in this country by individuals who did not abandon, suppress, or conceal it.

648. Finally, [Sadowski11] is a declaration from Dan Sadowski. This declaration shows what was known or used by others in this country before the invention by the applicants of the patents-in-suit, what was in public use in this country more than one year prior to the date of application of the patents-in-suit, and what was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed.

**b. QuickTime as prior art**

649. The reference [Drucker92] was a printed publication before the invention by the applicants of the patents-in-suit. In addition, this reference corroborates that Hypercard and Quicktime software and XCMDs (including the MOVIE XCMD) were known or used by others in this country before the invention by the applicants of the patents-in-suit. This reference corroborates that Hypercard and Quicktime software and XCMDs (including the MOVIE XCMD) were in public use in this country more than one year prior to the date of application of the patents-in-suit. This reference was a printed publication more than one year prior to the date of application of the patents-in-suit. Finally, this reference corroborates that before the invention by the applicants for the patents-in-suit, Hypercard and Quicktime software and XCMDs (including the MOVIE XCMD) were made in this country by individuals who did not abandon, suppress, or conceal it.

650. Similarly, the reference [Borrell93] was a printed publication before the invention by the applicants of the patents-in-suit. In addition, this reference corroborates that Hypercard and Quicktime software and XCMDs (including the QtMovie XCMD) were known or used by others in this country before the invention by the applicants of the patents-in-suit. This reference corroborates that Hypercard and Quicktime software and XCMDs (including the QtMovie XCMD) were in public use in this country more than one year prior to the date of application of the patents-in-suit. This reference was a printed publication more than one year prior to the date of application of the patents-in-suit. Finally, this reference corroborates that before the invention by the applicants for the patents-in-suit, Hypercard and Quicktime software and XCMDs (including the QtMovie XCMD) were made in this country by individuals who did not abandon, suppress, or conceal it.

651. Exemplary software that I reviewed related to the MOVIE XCMD was provided with Claris HyperCard 2.1 and can be found at [PA-NAT-00000010.] This software corroborates that Hypercard and Quicktime software and XCMDs were known or used by others in this country before the invention by the applicants of the patents-in-suit. HyperCard 2.1 including Claris QuickTime XCMDs were widely distributed by Apple Computer and/or its subsidiaries by 1991. Also, this software corroborates that Hypercard and Quicktime software and XCMDs were in public use in this country more than one year prior to the date of application of the patents-in-suit. HyperCard 2.1 including Claris QuickTime XCMDs were widely distributed by Apple Computer and/or its subsidiaries by 1991. It also corroborates that before the invention by the applicants for the patents-in-suit, Hypercard and Quicktime software and XCMDs were made in this country by individuals who did not abandon, suppress, or conceal it. HyperCard and Claris Quicktime XCMDs were made in this country by at least Apple Computer and/or its subsidiaries, who did not abandon, suppress, or conceal it.

652. Exemplary software I reviewed related to the QTMovie XCMD was from "Interactive Speech Systems:SE stack" in Apple Chronicle CD, dated March 19, 1992, from [PA-NAT-00000068; PA-NAT-00000069]. This software corroborates that Hypercard and Quicktime software and XCMDs (including QTMovie) were known or used by others in this country before the invention by the applicants of the patents-in-suit. Apple QuickTime XCMDs were known or used at least by Apple Computer, its subsidiaries, and/or Apple Developer's Association. (See, e.g., [Drucker92] and [Borrell93].) The Apple Chronicle CD-ROM was known or used by at least 1992, when it was publicly distributed with Apple Computers. In addition, this software corroborates that Hypercard and Quicktime software and XCMDs (including QTMovie) were in public use in this country more than one year prior to the date of application of the patents-in-suit. Apple QuickTime XCMDs were made available for public use by at least Apple Computer, its subsidiaries, and/or Apple Developer's association. This also corroborates that before the invention by the applicants for the patents-in-suit, Hypercard and Quicktime software and XCMDs (including QTMovie) were made in this country by individuals who did not abandon, suppress, or conceal it. Apple QuickTime XCMDs were made by at least Apple Computer, its subsidiaries, and/or Apple Developer's Association.

653. Other exemplary software I reviewed related to the QTMovie XCMD was obtained from :Unsupported Stuff:XCMDs:QTMovie Stack in "QuickTime: The Official Guide for Macintosh Users," dated April 26, 1994. [PA-NAT-00000077]. This software corroborates that Hypercard and Quicktime software and XCMDs (including QTMovie) were known or used by others in this country before the invention by the applicants of the patents-in-suit. This corroborates that before the invention by the applicants for the patents-in-suit, Hypercard and Quicktime software and XCMDs were made in this country by individuals who did not



abandon, suppress, or conceal it. Apple QuickTime XCMDs were made by at least Apple Computer, its subsidiaries, and/or Apple Developer's Association.

654. In addition, I prepared videos showing HyperCard and the MOVIE XCMD. (See Appendix C at hypercard\_movie\_2.wmv, hypercard\_movie\_5.wmv ). I also prepared videos showing HyperCard and the QTMovie XCMD. (See Appendix C at hypercard\_movie\_1 - B.wmv, hypercard\_movie\_3.wmv, hypercard\_movie\_4.wmv, hypercard\_movie\_6- A.wmv, hypercard\_movie\_6-B.wmv, hypercard\_movie\_7.wmv ).

**c. Obviousness based on HyperCard/QuickTime and Viola**

655. I have constructed claim charts for all of the asserted claims for the prior art references represented by Viola (Claim Chart Exhibits 1 through 6) and HyperCard-QuickTime (Claim Chart Exhibits 11-12). I have found that both references satisfy all or nearly all of the limitations of the '906 and '985 patents and thus render all asserted claims of these patents obvious. This opinion applies to the references used in either Claim Chart Exhibit 11 or Claim Chart Exhibit 12, in combination with the Viola codebases/references used in Claim Chart Exhibit 1, 2, 3, 4, 5, or 6. That is, my opinion is that HyperCard and QuickTime (whether using the QTMovie XCMD or the Movie XCMD) in combination with Viola (whether using [Viola-5/12/93], [Viola-5/27/93] , [Viola-Alpha], [Viola-Feb Beta], [Viola-March Beta], or [Wei94] and the other references cited in Claim Chart Exhibit 6), render obvious all asserted claims of the patents in suit.

656. With detailed citation to the claim charts, I will discuss where each reference could be combined with another reference that is clearly in satisfaction or the other of the references plays a complementary role in the combined satisfaction.

657. Viola (including [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], [Viola-March Beta], or [Wei94] and the other references cited in Claim Chart Exhibit 6) satisfies all of the '906 and '985 limitations including those

where there is a sub-requirement that "Text formats are HTML tags"<sup>23</sup>, "Object is displayed at first location in hypermedia document"<sup>24</sup>, and "Interaction with object is at first location in hypermedia document"<sup>25</sup>. I list the foregoing sub-requirements of limitations because there is possibly not complete satisfaction of them by the companion prior art reference, HyperCard-QuickTime.

658. For "HTML tag" sub-requirements, ViolaWWW running on the client workstation can receive hypermedia document files that contain text formats in the form of HTML tags from a network server (e.g., a file server or HTTP server) over a distributed hypermedia network environment. Examples of such documents can be found in all the Viola codebases that I am opining on, including for example [Viola-5/12/93] in \viola\docs. For example, the testall.html file contains HTML tags, such as TITLE and H1.

659. For the "first location" sub-requirements, in one example, the object is displayed and interactive processing of the object occurs at the first location within the portion of the hypermedia document displayed in the ViolaWWW window. For example, the vplot application (e.g., [Vplot-Sun] or [Vplot-petra]) displays the object as a grid (the default grid) inside the ViolaWWW window. The object is displayed at the first location in the portion of the testPlot.hmml hypermedia document being displayed in the ViolaWWW window. (See, e.g., [Viola-5/12/93] at \viola\docs\testPlot.hmml.). Also, for example, the vplot executable application enables the user to directly interact with the object using the slider bars to rotate the

---

<sup>23</sup> Limitations related to "HTML tags" can be found in at least the claim limitations: 985-3.a; 985-18.a; 985-22.a; 985-26.a; 985-38.a; 985-42.a.

<sup>24</sup> Limitations related to displaying an object at a "first location" include at least: 985-24.b; 985-28.b; 985-36.h; 985-40.i.

<sup>25</sup> Limitations related to interaction with objects at the "first location" can be found in at least the claim limitations: 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i.

object around the X, Y and Z axes. (See, e.g., [Viola-5/12/93] at apps\plot.v, src\cl\_slider.c, src\cl\_client.c.) This interaction with the object occurs at the first location within the portion of the HMML document displayed in the ViolaWWW window. (See, e.g., [Viola-5/12/93] at docs\testPlot.hmml, apps\plot.v).

660. All of the limitations listed above and all limitations of the asserted claims are completely satisfied by Viola.

661. HyperCard-QuickTime satisfies all of the '906 and '985 limitations except possibly for those where there is a sub-requirement that "Text formats are HTML tags"<sup>26</sup>, "Object is displayed at first location in hypermedia document"<sup>27</sup>, and "Interaction with object is at first location in hypermedia document".<sup>28</sup>

662. For the "HTML tag" sub-requirements, I have noted that the text format tags used by HyperCard, while not HTML, are nonetheless tags that HyperCard can recognize to direct the way it lays out each card associated with each segment in the stack. HTML was known to the HyperCard developers but storing and reading binary data to and from the data and/or resource fork achieved an efficiency and speed not possible by parsing raw text. All the limitations relating to this sub-requirement are satisfied by Viola.

663. For the "first location" sub-requirements, HyperCard-QuickTime does not satisfy follow the Court's discussion of this limitation, since there is not an inline sequential correspondence between the location of the embed text format and the object in the HyperCard stack. However, as broad as [Martin11] has applied this limitation, HyperCard-QuickTime does disclose a HyperTalk script stored in the data

---

<sup>26</sup> Limitations related to "HTML tags" can be found in at least the claim limitations: 985-3.a; 985-18.a; 985-22.a; 985-26.a; 985-38.a; 985-42.a.

<sup>27</sup> Limitations related to displaying an object at a "first location" include at least: 985-24.b; 985-28.b; 985-36.h; 985-40.i.

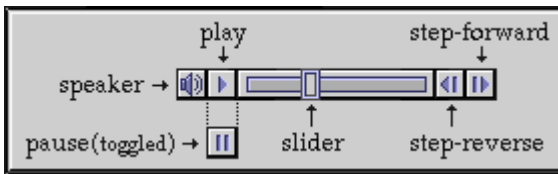
<sup>28</sup> Limitations related to interaction with objects at the "first location" can be found in at least the claim limitations: 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i.

fork<sup>29</sup> of a segment of a HyperCard document as one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. Such a script could be:

```
on mouseUp
  QTMovie OpenMovie, Direct, fileName, location, [,options...]
end mouseUp
```

664. The object in this script is a file, *fileName*, which is specified directly and is stored elsewhere as a separate file. The object is displayed in a location that generally corresponds to the first location (e.g. the segment of the stack corresponding to the card). Interaction with the object at the first location is provided by a standard controller which is associated with the QTMovie XCMD. Such a controller is shown below, annotated to indicate purpose.

665. However, under the plain meaning and understanding the Court's discussion of this limitation, the "first location" limitation is satisfied by the combination of HyperCard-QuickTime and Viola.



666. Therefore, all of the limitations listed above, not completely satisfied by HyperCard-QuickTime, are completely satisfied by Viola.

667. To the extent it is found that Viola does not satisfy sub-requirements directed to "type-information,"<sup>30</sup> HyperCard and Quicktime provide disclosure of

<sup>29</sup> In my earlier report (paragraphs 616 and 639) I incorrectly noted that the HyperTalk script was stored in the resource fork. I noticed this error upon review of Daniel Sadowski's deposition transcript. This is not to say that all scripts are stored in the data fork - Lingo scripts, for example, can be stored in the resource fork.

type information through the QTMovie and Movie HyperTalk script syntax for invoking XCMDs.

668. In addition, to the extent it is found that Viola does not satisfy sub-requirements directed to the notion of "objects,"<sup>31</sup> HyperCard and Quicktime provide disclosure of these claim limitations because the QTMovie and Movie HyperTalk script syntax calls for identifying for display a movie object through its *filename*.

669. It would have been obvious to one of ordinary skill in the art at the time of the alleged invention to combine Viola and HyperCard-QuickTime. First of all, they were contemporaneous developments, with the popular HyperCard IIGS version appearing in 1991, HyperCard 2.1 appearing by 1991, and early releases of Viola appearing the same year. In fact, the motivation to combine was especially strong for Wei because he began developing Viola after being inspired by HyperCard. Gillies and Cailliau quote Wei on this discovery: "HyperCard was very compelling back then, you know graphically, this hyperlink thing, it was just not very global and it only worked on Mac...and I didn't even have a Mac" (p. 213). Only having access to X terminals, he (in 1990) created the first version of *Viola* for them: "I got a HyperCard manual and looked at it and just basically took the concepts and implemented them in X-windows[sic]" (p. 213).

670. This can also be seen in the Viola prior art. In [Viola-5/12/93], Wei notes the similarity of Viola and HyperCard: "Viola is desigend to aid the

---

<sup>30</sup> Limitations related to "type-information" include at least 906-1.g; 906-6.g; 985-1.f; 985-16.f; 985-20.g; 985-24.j; 985-28.j; 985-10.a; 906-1.g; 906-6.g; 985-1.g; 985-16.g; 985-20.h; 985-24.k; 985-28.k; 985-36.g; 985-40.h.

<sup>31</sup> Limitations for "object" or relating to "object" include at least 906-1.f; 906-6.f; 985-1.f; 985-16.f; 985-20.g; 985-24.i; 985-28.i; 985-36.f; 985-40.g; 985-9.a; 985-36.f; 985-40.g; 906-1.f; 906-6.f; 985-36.f; 985-40.g; 985-1.f; 985-16.f; 985-20.g; 985-24.b; 985-24.i; 985-28.b; 985-28.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i; 906-1.g; 906-6.g; 985-1.f; 985-16.f; 985-20.g; 985-24.j; 985-28.j; 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.h; 985-20.i; 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-36.h; 985-40.i; 985-24.b; 985-28.b; 985-24.b; 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i.

development and support of interactive/hyper media applications for the Unix/X platform. Its functionality is similar to HyperCard [1] and Tcl/Tk [2]."

671. In a montage's of early Viola versions [Viola-Montage], it is stated "This is a montage of the first version of viola (0.8) released in 1991. This version is now unsupported... It was a spare time project done between classes while the author was an undergraduate. It was heavily influence by HyperCard(tm) concepts."

672. In addition, it was known in the prior art that HyperCard could be used as a client to the World Wide Web, just as ViolaWWW was. For example, [Morgan] states on page 427 that it "would be possible to create WAIS, Gopher, or World Wide Web clients simply by modifying the scripts given in this article." That article shows how to connect a HyperCard stack to the TCP/IP Internet, which is the home of the World Wide Web.

673. Finally, HyperCard and web browsers like ViolaWWW were alternative types of hypertext/hypermedia systems that were known in the art. I describe in my report above how hypermedia browsers operated in networked environments, and also describe that World Wide Web browsers were just one example of a hypermedia browser operating on a particular network (the Internet) using a particular set of protocols.

674. It is my opinion that the combination of prior art references Viola and HyperCard-QuickTime satisfies all of the limitations of all asserted claims of the '906 and '985 patents, and that it would have been obvious for one of ordinary skill in the art to combine them to implement the functionality of the alleged invention. I further note in this regard that Eolas's Patent Rule 3-2 Infringement Contentions accused the Quicktime authoring tool of infringement, and the identity of the functionality provided by Quicktime (e.g. to create interactive movies) is unchanged since more than a year before the patents were filed. Thus Eolas's Patent Rule 3-2 Infringement Contentions themselves are evidence of the obviousness of the patent claims.

**d. Obviousness based on HyperCard/QuickTime and Mosaic**

675. In a subsequent section of my report, directed to HyperCard/Director, I state that HyperCard/Director in combination with Mosaic renders obvious all claims of the patents-in-suit. In this regard I note that Eolas accused Director of infringement too. (See, e.g., 906 - Adobe - Authoring Tools and Players (Final).PDF at 3-5 and 111 of 158; 985 - Adobe - Authoring Tools and Players (Final).PDF at 3-5, 173-174, 192-193 of 247.)" The same analysis and reasoning that I applied to arrive at that conclusion likewise applies to HyperCard/QuickTime (whether using the QTMovie XCMD or the Movie XCMD) and Mosaic, and accordingly I believe that this combination renders obvious all asserted claims of these patents. Specifically, claim charts for all of the asserted claims for the prior art references represented by Viola (Claim Chart Exhibits 1 – 6) and HyperCard/QuickTime (Claim Chart Exhibits 11 and 12) are attached. This opinion of obviousness applies to the references used in Claim Chart Exhibits 11 or 12, in combination with the Mosaic references used in Claim Chart Exhibit 8.

676. I incorporate by reference here my discussion below for why Mosaic satisfies all of the '906 and '985 limitations for which there is possibly not complete satisfaction by HyperCard/Director. That discussion applies to HyperCard/QuickTime with equal force, but with the QTMovie or MOVIE XCMD syntax for QuickTime instead of the playmovie XCMD syntax for Director.

**e. Director as prior art**

677. Based on Dan Sadowski's declaration [Sadowski11], MacroMind released a product called the HyperCard Driver in 1988 which allowed the playback of movies created in VideoWorks within a HyperCard stack. ([Sadowski11] at 11-12, 42, 46, 61.) In 1989, MacroMind re-branded its VideoWorks software as Director. ([Sadowski11] at 16.) Between 1990 and 1991, MacroMind released numerous versions of Director including Director versions 2 and 3. ([Sadowski11] at 17, 21.)

Director version 3.1 was released sometime between May of 1992 and early 1993 and Director 3.1.3 was also released in June of 1993. ([Sadowski11] at 23, 61-62.) Based on the release dates of the various versions of Director, it was in use by others in this country before the invention by the applicants of the patents-in-suit. Moreover, Director was also in public use in this country more than one year prior to the date of application of the patents-in-suit.

678. In addition, included in the releases for Director 2, 3.X, and 3.1.3 is the Director Player HyperCard Manual. ([Sadowski11] at 61.) The manual was implemented as a HyperCard Stack. ([Sadowski11] at 61-62.) This manual corroborates that Hypercard and Director software and XCMDs (including the playMovie XCMD) were in use by others in this country prior to the invention by the applicants of the patents-in-suit. It also corroborates that HyperCard and Director software and XCMDs (including the playMovie XCMD) were in public use more than one year prior to the date of application of the patents-in-suit. In addition, the manual corroborates that before the invention by the applicants for the patents-in-suit, Hypercard and Director software and XCMDs (including the playMovie XCMD) were made in this country by individuals who did not abandon, suppress, or conceal it.

679. As part of my analysis, I also reviewed the Director Player HyperCard Manual which was shipped with Director version 3.1.3. ([Sadowski11] at 62 and Exhibit T.) I used HyperCard version 2.1 to open the Director Player HyperCard Manual, and review the embedded interactive Director movies. The systems and software that I reviewed may be inspected by contacting counsel of record. My review corroborates that HyperCard and Director software and XCMDs were known or used by others in this country before the invention by the applicants of the patents-in-suit. HyperCard 2.1 was widely distributed by Apple Computer and/or its subsidiaries by 1991. Similarly, Director was distributed by MacroMind by 1990,



and Director 3.1.3 by June 1993. Also, my review corroborates that Hypercard and Director software and XCMDs were in public use in this country more than one year prior to the date of application of the patents-in-suit. It also corroborates that before the invention by the applicants for the patents-in-suit, Hypercard and Directors software and XCMDs were made in this country by individuals who did not abandon, suppress, or conceal it. In particular, HyperCard and Director XCMDs were made in this country by at least Apple Computer and MacroMind and/or its subsidiaries and successors, who did not abandon, suppress, or conceal it.

680. Finally, I prepared videos showing HyperCard and the playMovie XCMD. (See Appendix C at hypercard\_movie\_8 (MM Dir).wmv).

**f. Obviousness based on HyperCard/Director and Mosaic**

681. It is further my opinion that HyperCard-Director prior art<sup>32</sup> in combination with Mosaic renders obvious all claims of the patents-in-suit.

682. Claim charts for all of the asserted claims for the prior art references represented by Mosaic (Claim Chart Exhibit 8) and HyperCard-Director (Claim Chart Exhibit 13) are attached.

683. I have found that both references satisfy all or nearly all of the limitations of the '906 and '985 patents and thus render all asserted claims of these patents anticipated or obvious. This opinion applies to the references used in Claim Chart Exhibit 8 in combinations with the HyperCard-Director codebases/references used in Claim Chart Exhibit 13. That is, my opinion is that HyperCard-Director in combination with Mosaic renders obvious all asserted claims of the patents in suit.

684. HyperCard-Director satisfies all of the '906 and '985 limitations except possibly for those where there is a sub-requirement that "Text formats are HTML

---

<sup>32</sup> This prior art can be found in the Director product itself, particularly the HyperCard stack with Director product information and demonstrations. See Sadowski Declaration.

tags,"<sup>33</sup> "Object is displayed at first location in hypermedia document,"<sup>34</sup> and "Interaction with object is at first location in hypermedia document."<sup>35</sup>

685. For the "HTML tag" sub-requirements, I have noted that the text format tags used by HyperCard, while not HTML, are nonetheless tags that HyperCard can recognize to direct the way it lays out each card associated with each segment in the stack. HTML was known to the HyperCard developers but storing and reading binary data to and from the data and/or resource fork achieved an efficiency and speed not possible by parsing raw text. All the limitations relating to this sub-requirement are satisfied by Mosaic.

686. For the "first location" sub-requirements, at least as broad as Eolas alleges infringement for modern browser mechanisms that do not require an inline embed text format such as the DOM and JavaScript solutions for embedding content, HyperCard-Director discloses a HyperTalk script stored in the data fork of a segment of a HyperCard document as one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. Such a script could be:

```
on presentMovie
    global returnSound
    playMovie "Cement Column"
    lock screen
    play returnSound
    pop card
```

---

<sup>33</sup> This can be found in at least the claim limitations 985-3.a; 985-18.a; 985-22.a; 985-26.a; 985-38.a; 985-42.a.

<sup>34</sup> Limitations related to displaying an object at a "first location" include at least: 985-24.b; 985-28.b; 985-36.h; 985-40.i.

<sup>35</sup> Limitations related to interaction with objects at the "first location" can be found in at least the claim limitations: 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i.

```
unlock screen with wipe right  
end presentMovie
```

687. The object in this script is a file with the name "Cement Column," which is specified directly as an argument to the playMovie XCMD and is stored elsewhere as a separate file. The object is displayed in a location that corresponds to the first location (e.g. the segment of the stack corresponding to the card). Interaction with the object at the first location is provided by mouse clicks which are associated with the playMovie XCMD.

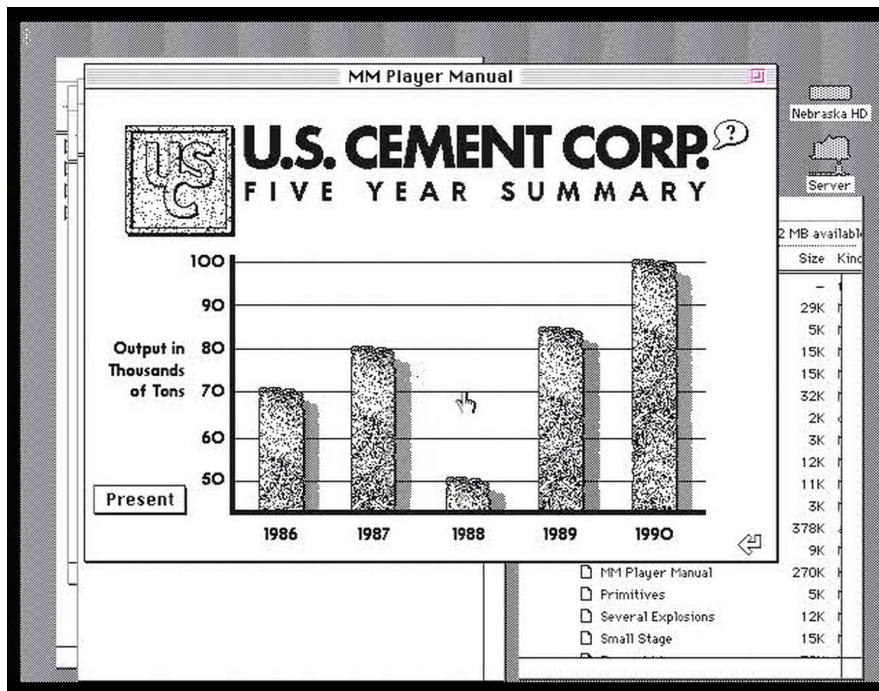
688. I understand that the Court's construction is consistent with my earlier application and understanding of how a person of skill in the art would have understood the limitation, in other words that strict adherence to the inline and locational aspect of the embed text format and object is required, thus the limitations relating to these sub-requirements are satisfied by additional teachings of Mosaic, which did preserve this relationship.

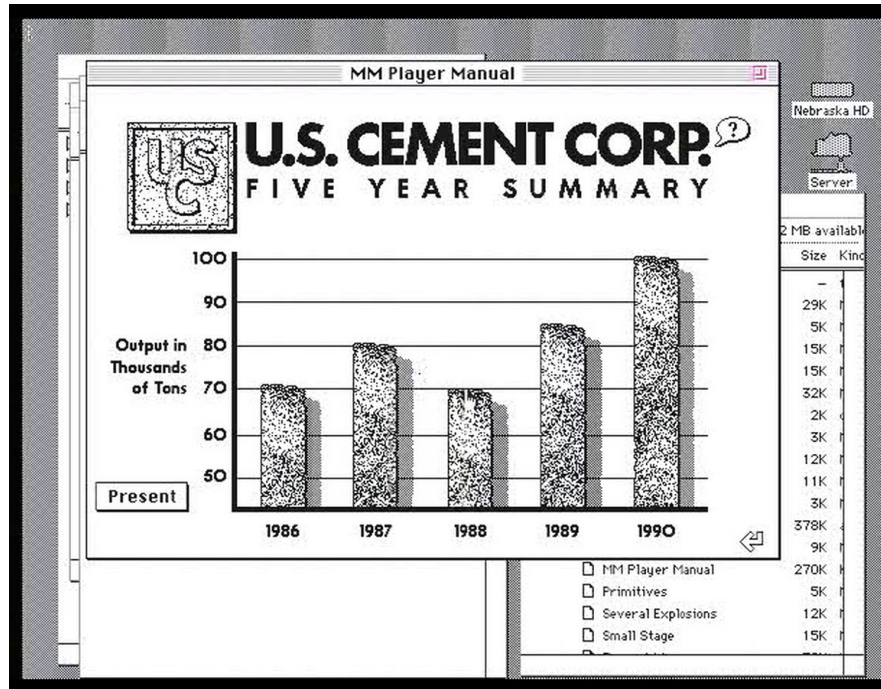
689. In this regard, a person of ordinary skill in the art could have used a file format like Mosaic used (e.g. HTML), which required an inline tagging mechanism at the time, and could have used for example a special tag like the image tag or an anchor tag as a mechanism for identifying the movie to include. I note the substantial discussion on www-talk related to this subject that is discussed elsewhere in my report concerning HTML+, Viola, MediaView and Adobe Acrobat (particularly [Adobe93] at 78-79 (suggesting making a compound PDF file with sound or video and using a special tagging mechanism very similar to the EMBED tag in the patent). In fact, I understand Viola was modeled after HyperCard functionality, Tim Berners-Lee at CERN (and later the W3C) kept a webpage of technologies including citations to information relating to many of these technologies (*see* 6/24/11 Doyle Depo. Exhibit 3), even early HyperCard manuals refer to the system as a form of "browsing" (*see* [HyperCardBasics90] at 5).

690. Therefore, all of the limitations listed above, not completely satisfied by HyperCard-Director alone, are satisfied as obvious in view of Mosaic.

691. To the extent it is found that Mosaic does not satisfy sub-requirements directed to "automatic invocation of executable application" or "automatic invocation does not require interactive action by the user," again HyperCard-Director provides disclosure of type information through the playMovie and Movie HyperTalk script syntax for invoking XCMDs.

692. For example, the screenshots below are taken from videos submitted with this report. They show interaction with an object, a Director Movie, Cement Column, in HyperCard through the use of a mouse.





693. The interaction in this instance is the resizing of the chart for the year 1988 from a magnitude of 90 thousand tons to 70 thousand tons through a mouse click.

694. It would have been obvious to one of ordinary skill in the art at the time of the alleged invention to combine Mosaic and HyperCard-Director. First, they related browser technologies directed toward hypertext / hypermedia systems, as is mentioned above. Second they were nearly contemporaneous with the HyperCard IIGS version appearing in 1991, the HyperCard 2.1 appearing by 1991, and early releases of Mosaic appearing the same year. Also and early versions of Mosaic being released in December 1992 and April 22, 1993. Third, it is precisely the type of information and reflects the habits and techniques of those of skill in the art to which the patent pertains that they would have looked at and shared resources and ideas of ways to improve the state of technology at the time as it pertained to Web technology. Indeed, even the inventors were aware of these resources, as they modified the Mosaic source, they were aware of the www-talk mailing list, and people they worked with informed them to visit these materials. [McRae93]

In addition, it was known in the prior art that HyperCard could be used as a client to the World Wide Web, just as Mosaic was. For example, [Morgan] states that it "would be possible to create WAIS, Gopher, or World Wide Web clients simply by modifying the scripts given in this article." That article shows how to connect a HyperCard stack to the TCP/IP Internet, which hosted the World Wide Web.

695. It is my opinion that the combination of prior art references Mosaic and HyperCard-Director satisfies all of the limitations of all asserted claims of the '906 and '985 patents, and that it would have been obvious for one of ordinary skill in the art to combine them to implement the functionality of the alleged invention. I further note in this regard that Eolas's Patent Rule 3-2 Infringement Contentions accused the Director authoring tool of infringement, and the identity of the functionality provided by Director (e.g. to create interactive movies) is unchanged since more than a year before the patents were filed. In fact, as Sadowski noted, Shockwave, which is still alleged to infringe in [Martin11] is just another Director projector that can be embedded in a hypermedia document. Thus Eolas's Patent Rule 3-2 Infringement Contentions themselves are evidence of the obviousness of the patent claims.

**g. Obviousness based on HyperCard/Director and Viola**

696. In the previous section of my report, directed to HyperCard/QuickTime, I stated that HyperCard and QuickTime (whether using the QTMovie XCMD or the Movie XCMD) in combination with Viola (whether using [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], [Viola-March Beta], or [Wei94] and the other references cited in Claim Chart Exhibit 6), render obvious all asserted claims of the patents in suit.

697. The same analysis and reasoning that I applied to arrive at that conclusion likewise applies to HyperCard/Director and Viola, and accordingly I believe that this combination renders obvious all asserted claims of these patents. Specifically, claim charts for all of the asserted claims for the prior art references

represented by Viola (Claim Chart Exhibits 1 – 6) and HyperCard-Director (Claim Chart Exhibit 13) are attached. This opinion of obviousness applies to the references used in Claim Chart Exhibit 13, in combination with the Viola codebases/references used in any one of Claim Chart Exhibit 1, 2, 3, 4, 5, or 6.

698. I incorporate by reference here my discussion above for why Viola (including [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], [Viola-March Beta], or [Wei94] and the other references cited in Claim Chart Exhibit 6) satisfies all of the '906 and '985 limitations, including those for which there is possibly not complete satisfaction by HyperCard/QuickTime. That discussion applies to HyperCard/Director with equal force, but with the playmovie XCMD syntax for Director instead of the QTMovie / MOVIE XCMD syntax for QuickTime.

#### **h. Claim construction issues related to HyperCard**

699. I understand that the court has construed the claim term "workstation" to mean "as a computer system connected to a network that serves the role of an information server." To the extent that Eolas opines on the extent to which a personal computer and workstation are/are not obvious variants of one another, or are/are not equivalent, I reserve the right to address this issue as well.

700. In addition, I have reconsidered all other claim limitations in view of the claim construction order and continue reserve the right to supplement my analysis of HyperCard prior art accordingly should Eolas or Eolas' experts attempt to interpret the claim construction order in a way that I believe to be incorrect.

### **8. Viola and Cohen**

701. U.S. Patent No. 5,367,621 to Cohen [Cohen94] ("Cohen") is prior art because it is a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent. Cohen issued as a patent on Nov. 22, 1994 from U.S. Application No. 07/755,709 filed on Sep. 6, 1991 by

Amy S. C. Cohen, Christopher F. Gleason, Donald R. Hyatt, Michael E. Moran, Jeffrey N. Stevens, and Alan J. Wecker.

702. In my opinion, to the extent that Viola prior art is not found anticipatory, the combination of Viola prior art and Cohen render obvious all claims of the patents-in-suit.

703. This opinion applies to Cohen in combination with any one of [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], [Viola-March Beta]. This opinion also applies to Cohen in combination with [Wei94] and the other references cited in Claim Chart Exhibit 6.

704. I have constructed claim charts for all of the asserted claims for the prior art references represented by Viola (Claim Chart Exhibits 1 through 6) and Cohen (Claim Chart Exhibit 14). I have found that both references satisfy all or nearly all of the limitations of the '906 and '985 patents. With detailed citation to the claim charts, I will discuss where each reference could be combined with another reference that is clearly in satisfaction or the other of the references plays a complementary role in the combined satisfaction.

705. Viola (including [Viola-5/12/93], [Viola-5/27/93], [Viola-Alpha], [Viola-Feb Beta], [Viola-March Beta], or [Wei94] and the other references cited in Claim Chart Exhibit 6) satisfies all of the '906 and '985 limitations including those where there is a sub-requirement that "Text formats are HTML tags",<sup>36</sup> "Embed text format specifies location of object directly"<sup>37</sup>, "Object is displayed at first location in hypermedia document"<sup>38</sup>, "Executable application enables direct interaction with object"<sup>39</sup>, "Enabling end-user to directly interact with object"<sup>40</sup>, "Executable

---

<sup>36</sup> Limitations related to HTML include 985-3.a; 985-18.a; 985-22.a; 985-26.a; 985-38.a; 985-42.a.

<sup>37</sup> Such limitations include 985-9.a.

<sup>38</sup> Such limitations include 985-24.b; 985-28.b; 985-36.h; 985-40.i.

<sup>39</sup> Such limitations include 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-36.h; 985-40.i.



application enables interactive processing of object"<sup>41</sup>, "Interaction with object is at first location in hypermedia document"<sup>42</sup>, and "Ongoing inter-process communications."<sup>43</sup> I list the foregoing sub-requirements of limitations because there is possibly not complete satisfaction of them by the companion prior art reference, Cohen.

706. For the "HTML tag" sub-requirements, ViolaWWW running on the client workstation can receive hypermedia document files that contain text formats in the form of HTML tags from a network server (e.g., a file server or HTTP server) over a distributed hypermedia network environment. Examples of such documents can be found in all Viola codebases that I am opining on, including for example [Viola-5/12/93] in \viola\docs. For example, the *testall.html* file contains HTML tags, such as TITLE and H1.

707. For the "Embed text format specifies location of object directly" sub-requirements, the VOBJF embed text format specifies the location of an object directly. For example, *testPlot.hmml* includes a VOBJF tag specifying the location of an object based on a filepath location in which the object can be found:

```
<VOBJF>/home/wei/viola/apps/plot.v<\VOBJF>  
(See, e.g., [Viola-5/12/93] at viola\docs\testPlot.hmml.)
```

708. For the "first location" sub-requirements in one example, the object is displayed and interactive processing of the object occurs at the first location within the portion of the hypermedia document displayed in the ViolaWWW window. For example, the *vplot* application (e.g., [Vplot-Sun] or [Vplot-petra]) displays the object

---

<sup>40</sup> Such limitations include 985-24.b; 985-28.b.

<sup>41</sup> Such limitations include 985-24.b.

<sup>42</sup> Such limitations include 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i

<sup>43</sup> Such limitations include 906-3.a; 906-8.a.

as a grid (the default grid) inside the ViolaWWW window. The object is displayed at the first location in the portion of the testPlot.hmml hypermedia document being displayed in the ViolaWWW window. (See, e.g., [Viola-5/12/93] viola\docs\testPlot.hmml.). Also, for example, the vplot executable application enables the user to directly interact with the object using the slider bars to rotate the object around the X, Y and Z axes. (See, e.g., [Viola-5/12/93] at apps\plot.v, src\cl\_slider.c, src\cl\_client.c.) This interaction with the object occurs at the first location within the portion of the HMML document displayed in the ViolaWWW window. (See, e.g., [Viola-5/12/93] at docs\testPlot.hmml, apps\plot.v)

709. For the "Executable application enables direct interaction" sub-requirements, the vplot executable application enables the user to directly interact with the object using the slider bars to rotate the object around the X, Y and Z axes. (See, e.g., [Viola-5/12/93] apps\plot.v, src\cl\_slider.c, src\cl\_client.c.)

710. For the "Enabling end-user to directly interact" sub-requirement, the vplot executable application enables the user to directly interact with the object using the slider bars to rotate the object around the X, Y and Z axes. (See, e.g., [Viola-5/12/93] apps\plot.v, src\cl\_slider.c, src\cl\_client.c.).

711. For the "Executable application enables interactive processing" sub-requirement, the vplot executable application enables the user to directly interact with the object using the slider bars to rotate the object around the X, Y and Z axes. (See, e.g., [Viola-5/12/93] at apps\plot.v, src\cl\_slider.c, src\cl\_client.c.).

712. For the "Ongoing inter-process communications" sub-requirements, ViolaWWW browser can communicate with an application using inter-process communication. As noted in Viola's documentation, "... viola is message driven. Messages can originate from window system events and user-GUI interaction, generated by timers, interprocess communication, and objects' scripts." (See, e.g., [Viola-5/12/93] at viola\docs\violaBrief.hmml.). In one example, the ViolaWWW

browser communicates with vplot using inter-process communication. (See, e.g., [Viola-5/12/93] at `viola\apps\plot.v`, `viola\src\cl_TTY.c.`) (See also `main.c` of vplot source code in [vplot-petra].

713. All of the limitations listed above and all limitations of the asserted claims are completely satisfied by Viola.

714. Cohen satisfies all of the '906 and '985 limitations except for those where there is a sub-requirement that "Text formats are HTML tags"<sup>44</sup>, "Embed text format specifies location of object directly"<sup>45</sup>, "Object is displayed at first location in hypermedia document"<sup>46</sup>, "Executable application enables direct interaction with object"<sup>47</sup>, "Enabling end-user to directly interact with object"<sup>48</sup>, "Executable application enables interactive processing of object"<sup>49</sup>, "Interaction with object is at first location in hypermedia document"<sup>50</sup>, and "Ongoing inter-process communications."<sup>51</sup>

715. For the "HTML tag" sub-requirements, I have noted that, while the use of HTML tags was not disclosed by Cohen, it would have been obvious to one having ordinary skill in the art to use HTML instead of GML, the Generalized Markup Language developed by IBM. Cohen was filed in 1991, at a time when HTML

---

<sup>44</sup> Limitations related to "HTML tags" can be found in at least the claim limitations: 985-3.a; 985-18.a; 985-22.a; 985-26.a; 985-38.a; 985-42.a.

<sup>45</sup> Limitations in which the embed tag specifies the location of a object "directly" include 985-9.a.

<sup>46</sup> Limitations related to displaying an object at a "first location" include at least: 985-24.b; 985-28.b; 985-36.h; 985-40.i.

<sup>47</sup> Such limitations can be found at 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-36.h; 985-40.i.

<sup>48</sup> Such limitations can be found at 985-24.b; 985-28.b.

<sup>49</sup> Such limitations can be found at 985-24.b.

<sup>50</sup> Limitations related to interaction with objects at the "first location" can be found in at least the claim limitations: 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i..

<sup>51</sup> Such limitations can be found at 906-3.a; 906-8.a.

standardization activities were at their peak. All the limitations relating to this sub-requirement are satisfied by Viola.

716. For the "Embed text format specifies location of object directly" sub-requirements, I have noted that the embed text format in Cohen is an instance of an LID tag, which does not directly specify the location of an object. Rather, it references an LDESC tag which *does* directly specify the location of an object. The LDESC tags are kept in the prologue of the document. This was actually a feature, because it allowed an author to define the embed text format once, and then re-use that text format within the hypermedia document using a shorter link identification tag, without needing to re-type the full LDESC text format. Therefore, it is my opinion that it would have been obvious to use LDESC text formats within the document's formatted text stream such that the embed text format was at a first location in the hypermedia document. All the limitations relating to this sub-requirement are satisfied by Viola.

717. For the "first location" sub-requirements, Cohen discloses that the object is displayed on an auxiliary display device and not at a first location in the hypermedia document, i.e. the location of an LID tag determined by parsing. All the limitations relating to this sub-requirement are satisfied by Viola.

718. For the "Executable application enables direct interaction" sub-requirements, Cohen does not explicitly disclose direct interaction with an object, except for the obvious interaction of starting, pausing and stopping the presentation of multimedia objects. Cohen does suggest, however, the possibility of direct interaction because it discloses a variety of multimedia objects, including those that inherently require user interaction, such as spreadsheet objects. ([Cohen] at col. 2, line 63 - col. 3, line 2). All the limitations relating to this sub-requirement are satisfied by Viola.

719. For the "Enabling end-user to directly interact" sub-requirement, Cohen does not explicitly disclose direct interaction with an object, except for the obvious interaction of starting, pausing and stopping the presentation of multimedia objects. Cohen does suggest, however, the possibility of direct interaction because it discloses a variety of multimedia objects, including those that inherently require user interaction, such as spreadsheet objects. ([Cohen] at col. 2, line 63 - col. 3, line 2). All the limitations relating to this sub-requirement are satisfied by Viola.

720. For the "Executable application enables interactive processing" sub-requirement, Cohen does not explicitly disclose direct interaction with an object, except for the obvious interaction of starting, pausing and stopping the presentation of multimedia objects. Cohen does suggest, however, the possibility of direct interaction with executable applications such as video applications. Videos are interactively processed by the video.exe executable applications and underlying handlers. All the limitations relating to this sub-requirement are satisfied by Viola.

721. For the "Ongoing inter-process communications" sub-requirements, Cohen does not explicitly disclose direct interaction with an object, except for the obvious interaction of starting, pausing and stopping the presentation of multimedia objects. Cohen does suggest, however, the possibility of direct interaction because it discloses a variety of multimedia objects, including those that inherently require user interaction, such as spreadsheet objects. (col. 2, line 63 - col. 3, line 2). All the limitations relating to this sub-requirement are satisfied by Viola.

722. All of the limitations listed above, not completely satisfied by Cohen, are completely satisfied by Viola.

723. To the extent it is found that Viola does not satisfy sub-requirements directed to "type-information,"<sup>52</sup> Cohen provides clear disclosure of type information through its "OBJTYPE" attribute of the LDESC tag (which can take on any of the value PROGRAM/ANIMATION/VIDEO/AUDIO/GRAPHIC/IMAGE), and/or the DATA attribute of the LDESC tag (which can be used to specify the executable application that handles an object).

724. In addition, to the extent it is found that Viola does not satisfy sub-requirements directed to the notion of "objects,"<sup>53</sup> Cohen provides clear disclosure of such objects. By way of example, Cohen discloses that "[m]any different kinds of multimedia objects can be linked into a softcopy book. Multimedia objects such as high resolution, photographic quality graphics, motion video, or sound can be supported by the invention. In addition, other functions which can be included in an expanded definition of multimedia can also be presented, such as a spread sheet, or an engineering diagram using a computer aided design data base." Examples provided by Cohen include animation objects (e.g., "family\_clip.anm"), video objects ("family\_clip.vid"), audio objects (e.g., "trumpet.aud"), and graphic objects (e.g., "population.gph").

725. It would have been obvious to one of ordinary skill in the art at the time of the alleged invention to combine Viola and Cohen. Indeed, they were

---

<sup>52</sup> Limitations related to "type-information" include at least 906-1.g; 906-6.g; 985-1.f; 985-16.f; 985-20.g; 985-24.j; 985-28.j; 985-10.a; 906-1.g; 906-6.g; 985-1.g; 985-16.g; 985-20.h; 985-24.k; 985-28.k; 985-36.g; 985-40.h.

<sup>53</sup> Limitations for "object" or relating to "object" include at least 906-1.f; 906-6.f; 985-1.f; 985-16.f; 985-20.g; 985-24.i; 985-28.i; 985-36.f; 985-40.g; 985-9.a; 985-36.f; 985-40.g; 906-1.f; 906-6.f; 985-36.f; 985-40.g; 985-1.f; 985-16.f; 985-20.g; 985-24.b; 985-24.i; 985-28.b; 985-28.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i; 906-1.g; 906-6.g; 985-1.f; 985-16.f; 985-20.g; 985-24.j; 985-28.j; 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.h; 985-20.i; 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-36.h; 985-40.i; 985-24.b; 985-28.b; 985-24.b; 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-24.b; 985-28.b; 985-36.h; 985-40.i.

contemporaneous developments, with Cohen issuing November 22, 1994 and beta releases of Viola and [Wei94] appearing the same year.

726. Also, both were directed to the same purpose. Cohen was directed to a platform for browsing electronic on-line text books. ([Cohen] at col. 1, lines 24-32.) Viola was likewise designed to be a platform for electronic on-line text books. This is evident from the "docs" subdirectories of the various Viola codebases, which include HMML and HTML documents that were styled as book chapters. This is also evident based on the trial testimony of Pei Wei, who testified that his job was to develop Viola to create a browser that O'Reilly and Associates could use to publish on-line text books. (Trial testimony from Pei Wei, Microsoft Corp., No. 99-C-626 (Jul. 28, 2003) [EOLASTX-E-0000000644].

727. It is my opinion that the above-mentioned combination of prior art references Viola and Cohen satisfies all of the limitations of all asserted claims of the '906 and '985 patents, and therefore render them obvious.

728. I understand that the court has construed the terms "embed text format, located at a first location in said first distributed hypermedia document" and "embed text format [which] correspond[s/ing] to [a / said] first location in the document" To have their plain meaning. To the extent that Eolas or Eolas' experts attempt to characterize this construction in a way I believe is incorrect, I may opine that the LDESC and LID tags disclosed in Cohen satisfy claim limitations that include these terms.

729. In addition, I have reconsidered all other claim limitations in view of the claim construction order and continue to reserve the right to supplement my analysis of Cohen prior art accordingly should Eolas or Eolas' experts attempt to interpret the claim construction order in a way that I believe to be incorrect.

## **9. Any of the above-identified prior art, and distributed applications**

### **a. VIS**

730. VIS and publications about VIS are described in the patent as prior-art. It is described at [10:17-27] as, "Application client 210 is embodied in software presently under development called 'VIS' and 'Panel' created by the Center for Knowledge Management at the University of California, San Francisco, as part of the Doyle Group's distributed hypermedia object embedding approach described in 'Integrated Control of Distributed Volume Visualization Through the World-Wide-Web,' by C. Ang, D. Martin, M. Doyle; to be published in the Proceedings of Visualization 1994, IEEE Press, Washington, D.C., October 1994."

731. In the latter reference [Ang94] VIS is described as, "VIS is a highly modular distributed visualization tool, following the principles of client/server architecture (figure 1), and consisting of three cooperating processes: VIS, Panel, and VRServer(s). The VIS module handles the tasks of transformation, texture-mapping, isosurface extraction, Gouraud shading, and manages load distribution in volume rendering. VIS produces images that are drawn either to its own top-level window (when running stand-alone) or to a shared window system buffer (when running as a cooperative process). The Panel module provides a graphical user-interface for all VIS functionality and communicates state changes to VIS. The VRServer processes execute on a heterogeneous pool of general purpose workstations and perform volume rendering at the request of the VIS process."

732. [Ang94] is prior art because it was published before the invention by the applicants of the patents-in-suit.

733. Another paper describing VIS was published in February 1993, by Michael D. Doyle, Cheong Ang, Rakesh Raju, Gary Klein, Betsey S. Williams, Thomas DeFanti, Arsdeshir Goshtasby, Robert Grzeczuk, and Adrienne Noe. The paper was entitled "Processing of Cross-Sectional Image Data for Reconstruction of Human Developmental Anatomy from Museum Specimens." [Doyle93]. This paper discloses various aspects of distributed computing. For example, it describes



"Remote-access visualization and database tools are under development to allow real-time interaction with these enormous datasets by distributing certain computational tasks to super computers." Also, "In order to maintain the quick response needed for effective real-time interaction, the computational load of this application was distributed so that the interface panel, seen at the bottom of the screen, and the 3-D surface model were running on the CPU of the Silicon Graphics workstation. Computation of the high-resolution oblique section image displayed in the right window took place on the Convex supercomputer. Both of these operations occurred simultaneously, communicating through a high-speed fiber optic network." Figure 3 of that paper shows a "Screen image of the 3-D visualization program demonstrated at SIGGRAPH '92, in Chicago. This application was distributed between a Silicon Graphics Crimson VGXT workstation and a Convex supercomputer."

734. [Doyle93] is prior art because it was published before the invention by the applicants of the patents-in-suit.

735. The '906 patent's description of VIS, along with the publications described above, show that VIS was in public use more than one year prior to the date of the application for the patents-in-suit, and is therefore prior art.

736. In addition, I prepared videos demonstrating prior art VIS. (See Appendix C at VIS - A.wmv, and VIS - B.wmv.)

737. In my opinion, VIS teaches all of the asserted claim elements related to distributed processing, as shown by the disclosure I cite above.<sup>54</sup>

#### **b. Collage**

---

<sup>54</sup> The asserted claim elements related to distributed processing include 985-36.i; 985-40.j; 985-36.j; 985-40.k; 985-20.b; 985-40.b; 906-11.a; 906-13.a; 906-11.b; 906-13.b; 906-11.c; 906-13.c; 906-11.d; 906-13.d; 906-11.e; 906-13.e; 985-36.i; 985-40.j.

738. NCSA Collage [Collage92] was a scientific data analysis tool from the Macintosh group of the Software Development Group at NCSA. In a networked environment, this tool provided the capability to distribute most of these data analysis and visualization functions synchronously among a number of users. This was the foundation for the collaborative aspects of this tool's functionality. Ultimately Collage was made available not only on the Macintosh but on X-Windows and PC-compatible platforms as well.

739. Mosaic clients can communicate directly with Collage, as well as with Polyview, NCSA's collaborative tool for three-dimensional geometric and polygonal data analysis. Scientific data contained within a hypermedia document could be sent across the network to one of these programs for graphical and statistical inspection, and analysis.

740. Among Collage's many features was the ability to establish communication with remote processes, e.g. a simulation running on a supercomputer. These remote processes could be controlled remotely, and images and data could be transported to and from the remote process. Moreover, one could perform most of these operations not only on one machine, but on any machine that is participating in a collaborative session with NCSA Collage.

741. Consequently, collaborators using Mosaic clients and who were involved a Collage session could, for example, open and view an HDF (Hierarchical Data Format) file that was produced by a supercomputer computation. Members of the session could (non-destructively) annotate the displayed image to point out significant features.

742. Collage was prior art to the patents-in-suit.

743. According to the above description, Collage was known or used by others in this country before the invention by the applicants of the patents-in-suit.

744. In addition, as described above, Collage was described in the above-cited printed publication before the invention by the applicants of the patents-in-suit.

745. From the foregoing citation, Collage was in public use in this country more than one year prior to the date of application of the patents-in-suit.

746. Also, from above, Collage was described in a printed publication more than one year prior to the date of application of the patents-in-suit.

747. Finally, from above, Collage was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed.

748. In my opinion, and in view of my description above, Collage teaches all of the asserted claim elements related to distributed processing.<sup>55</sup>

### **c. Scientific Visualization Workbench**

749. The Scientific Visualization Workbench system was known or used by others in this country before the invention by the applicants of the patents-in-suit. An overview from [Phillips88] describes its main features as, "A system for visualization of data from supercomputer simulations has been developed for use by scientists and engineers at Los Alamos National Laboratory. The scientific visualization workbench, as the system is called, is based on an industry standard workstation (a Sun 3/160C), the NeWS window system, and a video/graphics add-in card, which is supported by NeWS. Also involved is a frame buffer attached to a 48-Mbit/s Cray channel and a video link from the frame buffer to the Sun workstation. Computation and graphics preparation are performed on the Cray under the control of a NeWS client program. Images are rasterized and sent to the workstation at the rate of 25 frames / sec. The workbench permits the scientist to converse with applications

---

<sup>55</sup> The asserted claim elements related to distributed processing include 985-36.i; 985-40.j; 985-36.j; 985-40.k; 985-20.b; 985-40.b; 906-11.a; 906-13.a; 906-11.b; 906-13.b; 906-11.c; 906-13.c; 906-11.d; 906-13.d; 906-11.e; 906-13.e; 985-36.i; 985-40.j.

running on a Cray but still retain the convenience and flexibility of a personal workstation environment. Moreover, the video link allows animated graphics images to be displayed on the workstation at a much higher rate than would be possible with a conventional network connection, i.e., Ethernet. Also, NeWS affords the convenience of a modern window system. Live video can be displayed in a window anywhere on the screen, while other windows may contain conventional text or graphics."

750. Further, from [Phillips88], "The operating system is UNICOS (a UNIX-like operating system developed by Cray Research), and the control and display functions are both incorporated into the workstation. From the workstation end, the user controls Cray activity over an Ethernet link through a NeWS process. This process communicates with a Cray-based NeWS client program, which in turn directs the frame buffer display software described above. There is a separate video link that transmits the frame buffer video signal to the Parallax processor. This is currently an NTSC video signal, derived by encoding the RGB output of the frame buffer. The signal is channelized onto a (ubiquitous) cable video network and decoded with a tuner at the user's location. The trade-off of image quality for bandwidth extracted by NTSC encoding was mentioned earlier; a later generation of the system will hopefully involve direct transmission of RGB video to the workstation."

751. Again, from [Phillips88], "Launching the workbench is simple. Once in the NeWS environment, the user opens an authorized session on the Cray through a terminal emulator window on the workstation. By pressing a (software) button on the workstation screen, the user begins the client process, which in turn establishes the client/server path on the network. At this point NeWS is in control of the entire distributed environment. The user issues commands via software sliders, buttons, and text fill-in dialogs. Figure 4 shows a typical screen layout for the workbench. In

the upper right, for example, there is a partly visible control panel for setting video parameters such as contrast and brightness. The image in the lower right shows a video image annotator and its associated controls. Other controls are used for such operations as video image sequencing and speed, video frame capture, graphics overlay, etc."

752. As noted above, the Scientific Visualization Workbench was described in a printed publication before the invention by the applicants of the patents-in-suit.

753. Also, the Scientific Visualization Workbench was in public use in this country more than one year prior to the date of application of the patents-in-suit.

754. In addition, the Scientific Visualization Workbench was described in the above-cited printed publication more than one year prior to the date of application of the patents-in-suit.

755. Finally, the Scientific Visualization Workbench was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed.

756. As a result of the foregoing observations, and in view of my description above, the Scientific Visualization Workbench teaches all of the asserted claim elements related to distributed processing.<sup>56</sup>

**d. Pei Wei's August 1994 paper [Wei94]**

757. In Pei Wei's 1994 August paper, he describes a distributed processing application of Viola by, "Here's another example of a mini interactive application that is embedded into a HTML document. It's a chess board in which the chess pieces are actually active and movable. And, illegal moves can be checked and denied straight off by the intelligence of the scripts in the application. Given more work, this chess

---

<sup>56</sup> The asserted claim elements related to distributed processing include 985-36.i; 985-40.j; 985-36.j; 985-40.k; 985-20.b; 985-40.b; 906-11.a; 906-13.a; 906-11.b; 906-13.b; 906-11.c; 906-13.c; 906-11.d; 906-13.d; 906-11.e; 906-13.e; 985-36.i; 985-40.j.

board application can front-end a chess server, connected to it using the socket facility in viola." [Wei94]

758. Pei Wei further characterized this application in August 31, 1994, when he explained to named inventor Michael Doyle the following:

Well. Viola's model was \*demonstrated\* in 1993, \*released\* freely in 1994. . . . And, as for the plotting demo, it actually is really just a front-end that fires up a back-end plotting program (and the point is that that back-end could very well be running on a remote super computer instead of the localhost). For that demo, there is a simple protocol such that the front-end app could pass an X window ID to the back-end, and the back-end draws the graphics directly onto the window violaWWW has opened for it.

759. By these descriptions he teaches how a front end, the viola application, can work with a back end server application. Thus data would be passed to the server, which would process it and send data back to the viola application.

760. Pei Wei's August 1994 paper was published before the filing date of the patents-in-suit, as corroborated by references cited in the report above.

761. As a result of the foregoing observations, and my description above, Pei Wei's 1994 paper anticipates all the asserted claim elements related to distributed processing.<sup>57</sup>

**e. Mathematica**

762. The first Mathematica reference manual, [Wolfram88], was published in 1988 and the software it describes was bundled with NeXTSTEP 1.0 that same year. Mathematica had two main components; a Front End and a Kernel. Both components were designated as Version 1.2. The Front End is a graphical user interface (GUI) that

---

<sup>57</sup>The asserted claim elements related to distributed processing include 985-36.i; 985-40.j; 985-36.j; 985-40.k; 985-20.b; 985-40.b; 906-11.a; 906-13.a; 906-11.b; 906-13.b; 906-11.c; 906-13.c; 906-11.d; 906-13.d; 906-11.e; 906-13.e; 985-36.i; 985-40.j.

accessed the Kernel, which was also known as the Mathematica server. MediaView, [Phillips91c], provided access to both the GUI and the server but only directly accessed the server through the Live Equation feature. As stated in [Phillips91c] this was done by using the UNIX pipes facility.

763. Access to the Mathematica is described in Chapter B.7, titled "External Interface." There is states, "While the computational aspects of Mathematica are essentially identical on all kinds of computers that run Mathematic, the external interface to Mathematic inevitably varies somewhat from one computer system to another."

764. Further, it states, "Mathematica is usually divided into two parts: the kernel that actually does computations, and a front end that deals with interaction with the user. The kernel is set up to be as similar as possible on every different kind of computer: the front end varies from one computer to another, taking advantage of particular features of each computer."

765. Finally, it is pointed out, "Under multi-tasking operating systems (such as UNIX), the front end is usually a separate process, which communicates with the kernel through a bidirectional pipe. The front end and kernel need not both be on the same computer; they can be running on separate computers, connected by a network or other communications link." Additional information on invocation of the Mathematica server is found in my report above.

766. Based on the foregoing discussion, Mathematica was known or used by others in this country before the invention by the applicants of the patents-in-suit, was in public use in this country more than one year prior to the date of application of the patents-in-suit, and was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed.

767. In addition, Mathematica was described in publications, such as the above-cited publication, before the invention by the applicants of the patents-in-suit, and more than one year prior to the date of application of the patents-in-suit.

768. I also recall knowing about, using, and creating software that operated with Mathematica more than one year prior to the date of application of the patents-in-suit.

769. As a result of the foregoing observations, and in view of my descriptions above, Mathematica teaches all of the asserted claim elements related to distributed processing.<sup>58</sup>

**f. RenderMan**

770. NeXTSTEP version 3.0, released on September 8, 1992, included Pixar's RenderMan software. From [NeXT92], "RenderMan software provided the industry standard for creating computer-based pictures that contain all the qualities of real life, such as shadings, reflections, texture and motion blur. NeXTSTEP Release 3.0 incorporated two components of Pixar's complete family of RenderMan products: full Photorealistic RenderMan and Interactive RenderMan."

771. Further, from [NeXT93a], "The 3D Graphics Kit enables NeXTSTEP applications to model and render 3-dimensional scenes. Much as the Application Kit's 2D graphics capabilities are based on the Display PostScript interpreter, the 3D Kit's capabilities are based on the Interactive RenderMan renderer. There are both similarities and differences in the inner workings of the two implementations. One similarity is that both are implemented with a client-server model, in which client applications send drawing code to the Window Server, which does the actual drawing. One difference in the implementations is in the code generated for drawing.

---

<sup>58</sup> The asserted claim elements related to distributed processing include 985-36.i; 985-40.j; 985-36.j; 985-40.k; 985-20.b; 985-40.b; 906-11.a; 906-13.a; 906-11.b; 906-13.b; 906-11.c; 906-13.c; 906-11.d; 906-13.d; 906-11.e; 906-13.e; 985-36.i; 985-40.j.



For 2D drawing, a View sends PostScript code to the Window Server's Display PostScript interpreter. For 3D drawing, a View sends RenderMan Interface Bytestream (RIB) code to the Window Server's Interactive RenderMan renderer."

772. Also, from [NeXT93a], "The photorealistic renderer supports the full RenderMan standard (with a few minor exceptions), so the images it generates display the detail and features specified in the original model. The photorealistic renderer operates as a separate process. It starts when invoked by a 3D Kit client and stops when the image based on that RIB has been rendered. For speedier rendering, the 3D Kit supports photorealistic rendering in multiple processes on multiple hosts.

773. Based on the foregoing discussion, RenderMan was known or used by others in this country before the invention by the applicants of the patents-in-suit, was in public use in this country more than one year prior to the date of application of the patents-in-suit, and was made in this country before the invention by the applicants for the patents-in-suit and was not abandoned, suppressed, or concealed.

774. In addition, RenderMan was described in publications before the invention by the applicants of the patents-in-suit, and more than one year prior to the date of application of the patents-in-suit.

775. As a result of the foregoing observations, RenderMan teaches all of the asserted claim elements related to distributed processing.<sup>59</sup>

**g. Motivations to combine**

776. In general, all of the browsers discussed in this report were extensible in some way – i.e., they were designed to interoperate with other applications in different ways. With Mosaic, it was helper applications. With Viola it was with

---

<sup>59</sup> The asserted claim elements related to distributed processing include 985-36.i; 985-40.j; 985-36.j; 985-40.k; 985-20.b; 985-40.b; 906-11.a; 906-13.a; 906-11.b; 906-13.b; 906-11.c; 906-13.c; 906-11.d; 906-13.d; 906-11.e; 906-13.e; 985-36.i; 985-40.j.

embedded objects such as vplot. With MediaView it was with Mathematica, RenderMan and custom components. With HyperCard it was with XCMDS.

777. Therefore, for any of these applications, it was a matter of choosing the platform on which to run it, and, usually, what network data resource to access to satisfy relevant data aspects of the browser application. As I've described above, it is my opinion that there are several prior art distributed computing applications that are obvious to combine with equally obvious browser combinations. These are:

- Mosaic, Viola, MediaView, or HyperCard with VIS. In fact, in several of my demonstration videos, I show how easy it was to combine Viola with VIS (Viola videos 9, 15, 19, and 24) (*See Appendix C.*)
- Mosaic, Viola, MediaView, or HyperCard with Collage. I note here that Mosaic did in fact interoperate with Collage. Furthermore, with Viola and Mosaic both being contemporaneous web browsers whose developers communicated with one another, it would have been particularly obvious to configure Viola to interoperate with Collage.
- Mosaic, Viola, MediaView, or HyperCard with Scientific Visualization Workbench. I note here that I developed software and authored papers on both MediaView and the Scientific Visualization Workbench.
- Mosaic, Viola, MediaView, or HyperCard with Pei Wei's August 1994 paper. In fact, the combination of any Viola codebase with the chess application disclosure would have been obvious because all the Viola tools necessary to create the chess application were already implemented and in place.
- Mosaic, Viola, MediaView, or HyperCard with Mathematica. I note here that MediaView did in fact interoperate with Mathematica.
- Mosaic, Viola, MediaView, or HyperCard with RenderMan. I note here that MediaView did in fact interoperate with RenderMan.

#### **h. Claim construction**

778. I understand that the Court has construed "distributed application" to mean "an application that is capable of being broken up and performed among two or more computers."

779. To the extent Eolas or Eolas' experts attempt to characterize this construction in a way I believe is incorrect, I may supplement my analysis to identify other applications in addition to those discussed in this section as satisfying claim limitations with this term. For example, I may opine that any browser interaction with a server or with an application residing (in whole or in part) on a server satisfies these claim elements.

### **10. Invention date**

#### **a. Opinions**

##### **i. Overview**

780. I understand from counsel that "conception" is the formation in the mind of the inventor of a definite and permanent idea of the complete and operative invention, as it is therefore to be applied in practice. I understand that an idea is sufficiently definite for conception when the inventor has a specific, settled idea – a particular solution to the problem at hand, not just a general goal or research plan he hopes to pursue.

781. I understand from counsel that an actual "reduction to practice" requires: (1) that the party constructed an embodiment or performed a process that meets every element of the claim, and (2) the embodiment or process works for its intended purpose.

782. I understand from counsel that "diligence" requires an accounting of facts showing that the named inventors were working towards a reduction to practice after an alleged conception, where the work relied upon is directly related to the reduction to practice of the invention at issue.

783. I also understand from counsel Eolas was required under local Patent Rule 3-2(b) to produce all documents showing the conception, reduction to practice, design, and development of each claimed invention, which were created on or before the date of application for the patent in suit or their priority dates.

784. I understand from counsel that Eolas produced materials under this Rule at the bates range EOLASTX-E-0000000001 and EOLASTX-0000000001 through EOLASTX-0000000312.

785. I have reviewed the documents at this bates range in forming my opinions about invention date.

786. In addition, I have reviewed Appendix A to the patent, which the patent specification indicates "includes NCSA Mosaic version 2.4 source code along with modifications to the source code to implement the present invention." ('906 Patent at 8:9-12.)

787. In reviewing these materials, I have concluded that at least the claim elements I discuss below are not entitled to the benefit of an earlier invention date.

788. First, there is no evidence that the named inventors reduced to practice the claim limitations related to "type information"<sup>60</sup> before the filing of the '906 patent.

789. Second, there is no evidence that the named inventors exercised diligence towards an actual reduction to practice of type information during 1994.

790. Third, there is no evidence of either conception or reduction to practice of claim limitations related to "enabling an end-user to directly interact" or "direct interaction"<sup>61</sup> with the object before the filing of the '906 patent.

---

<sup>60</sup> Limitations related to "type-information" include at least 906-1.g; 906-6.g; 985-1.f; 985-16.f; 985-20.g; 985-24.j; 985-28.j; 985-10.a; 906-1.g; 906-6.g; 985-1.g; 985-16.g; 985-20.h; 985-24.k; 985-28.k; 985-36.g; 985-40.h.

<sup>61</sup> This can be found in at least the following limitations: 906-1.h; 906-6.h; 985-1.h; 985-16.h; 985-20.i; 985-36.h; 985-40.i; 985-24.b; 985-28.b;

ii. **Opinion related to type information**

791. The '906 patent discloses an embodiment in which "type information" takes the form of a "TYPE" attribute located within the EMBED tag ('906 at 12:59-65):

```
<EMBED  
  TYPE = "type"  
  HREF="href"  
  WIDTH=width  
  HEIGHT=height  
>
```

792. The patent states that "[t]he TYPE element is a Multipurpose Internet Mail Extensions (MIME) type. Examples of values for the TYPE element are 'application/x-vis' or 'video/mpeg'. The type 'application/x-vis' indicates that an application named 'x-vis' is to be used to handle the object at the URL specified by the HREF. Other types are possible such as 'application/x-inventor', 'application/postscript' etc." ('906 Patent at 12:67 - 13:3.)

793. This shows that the named inventors considered the TYPE attribute to provide an extensible mechanism by which an author of a hypermedia document could specify the executable application that is to be automatically launched. They considered "type information" to be something that could readily be changed in order to have Mosaic automatically launch a different executable application.

794. An alternative approach would be to make modifications directly within the Mosaic source code that would result in Mosaic invoking a particular program, and then compiling the modified source code. This is known as "hardcoding." If one wanted Mosaic to instead invoke a different executable application, one would have to again modify Mosaic's source code such that Mosaic would invoke the other application, and then recompile the result to create a different binary.

795. To the extent one finds that (or to the extent Eolas argues that) TYPE information means the former rather than the latter, it is my opinion that there is no

evidence of a reduction to practice for "type information" before the filing of the '906 patent, nor is there evidence of any diligence towards such a reduction to practice during the 1994 time period.

**iii. Opinion related to enabling an end-user to directly interact**

796. I understand that this claim term as been agreed by the parties to mean "allowing a user to directly interact with the object."

797. A person of ordinary skill in the art would understand this to mean that in order to satisfy this claim limitation, a system must allow a user to interact with the object at the location at which the object is displayed. This is in contrast to a system that requires a user to interact with the object through a separate control panel in a separate window.

798. This also seems to reflect the understanding of the inventors, who amended the claims to recite "direct interaction" in order to reflect their view that interactive processing allows "a user to manipulate and control the object being displayed in the display area created in the browser controlled window." [858-PH], Applicant Remarks (September 27, 2007), at 10.

799. In my opinion, there is no evidence that the inventors either conceived of or reduced to practice this claim limitation before the filing date of the '906 patent, because all of the produced conception and reduction to practice documents only show interaction through a separate control panel.

**b. The '906 patent and appendices**

800. The '906 patent states that Appendix A "includes NCSA Mosaic version 2.4 source code along with modifications to the source code to implement the present invention." ('906 Patent at 8:9-12.)

801. In my original invalidity report I stated that "this source code shows that the named inventors did not reduce to practice type information by the time they filed the '906 patent." Since then, having read Mr. Ang's testimony regarding Exhibit

27, I now understand that the source code enables the selection of one of three external applications to be launched, according to the value of a type code in the range from 0 to 2. This is accomplished by specifying three character constant variables at compile time by the following statements:

```
char *vis_program = "/home/user/bin/panel";  
char *mpeg_program = "/usr/openwin/bin/xbiff";  
char *wt_program = "/usr/openwin/bin/xbiff";
```

802. Even so, this approach still represents a type of hard coding since a recompilation of the code would be required if any other application types were to be included. Thus, this implementation is not extensible. The particular application and path must be specified in the code before the browser application is compiled, so if additional applications were to be included in the future, they would require a modification of the source code and recompilation of the browser application code.

803. In addition, the '906 patent provides evidence that the named inventors did not reduce to practice claim limitations related to "enabling an end-user to directly interact." The embodiment disclosed in the patent shows that interaction with the object displayed in the image window 352 occurs in a separate panel window 354. (See, e.g., '906 patent at Figure 9.) This does not disclose direct interaction with the object displayed in image window 352.

**c. Other documents upon which Eolas relies as evidence of conception and reduction to practice**

804. I will now briefly summarize each of the items that Eolas identified as supporting conception or reduction to practice.

805. **EOLASTX-E-000000001:** This is a video in which named inventor Michael Doyle shows Mosaic purportedly interoperating with VIS. It does not show or suggest any other executable applications that would handle other data types, and

does not show type information that could be used to identify an executable application, nor few of the other limitations.

806. Furthermore, this shows interaction with the object in a separate panel window, and so does not show a reduction to practice of direct interaction.

807. Finally, I note that this video does not support conception or reduction to practice of many other claim limitations, because this video alone (which does not show the source code or the HTML document being rendered) does not indicate how the software was developed or is operating. For example, it does not show the use of a client-server network environment. The "file:///localhost" that appears in the address bar suggests this was a local demonstration. There is no evidence that VIS was operating as a distributed application.<sup>62</sup> The demonstration also does not indicate what text formats are being parsed or what (if any) embed text format is being used. Also, the video does not show automatic invocation because the video cuts to a segment in which VIS has already been invoked in one example, and it is not apparent what interactions might be happening off screen in other examples.

808. **EOLASTX-000000001:** This document is a declaration submitted by Michael Doyle in connection with the reexamination [831-PH]. The declaration itself does not disclose technical or implementation details, but rather only sets forth general assertions and concepts. It does not show evidence of reduction to practice of type information or direct interaction. It also does not show many other claim limitations, such as what text formats might be parsed (such as the embed text

---

<sup>62</sup> I note that the client-server network environment discussion here depends on how the Court's constructions are applied. As I read the Court's construction of client and server, both could be on the same computer, in which case the <file:///localhost> string in address bar is irrelevant. In other words, the same computer could be a distributed client-server environment with, for example, a single browser client and multiple server processes that respond to client requests. If my understanding of the Court's construction is correct, then the demonstration would at least show a client-server environment. If my application is not correct, then it would have been obvious to deploy the prior art systems in an actual client server system, such as in the native networking facilities provided in Sun workstations, the NeXT cube, or those of Apple's network services, such as AppleTalk or NFS.



format), or whether automatic invocation was used. Appended to this document, at EOLASTX-0000000004, is an article from Dr. Dobb's journal which is dated from 1996 and therefore does not show reduction to practice of type information or direct interaction before the filing date of the '906 patent.

809. **EOLASTX-0000000014:** This document is a single page taken from the [858-PH] prosecution history. This document does not include any technical or implementation details, and therefore does not show evidence of reduction to practice of type information or direct interaction. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

810. **EOLASTX-0000000015:** This document includes an invention disclosure form from the named inventors. Although the second page of this form (at EOLASTX-0000000016) asserts that the invention was conceived on September 7, 1993, I do not see any technical or implementation details on that page. On pages EOLASTX-0000000018 - 19, the named inventors describe aspects of their implementation, but do not disclose the concept of type information. They disclose an external program, but not external programs that can be identified by type information. On page EOLASTX-0000000020, there is a reproduction of a paper or draft paper by Cheong Ang. This paper does not show a reduction to practice of type information, because it only discloses Mosaic's interoperation with a single executable application - VIS. Section 5.2 of this paper shown in general terms that the named inventors thought to extend their implementation from VIS to other programs, but does not disclose that those programs or the data objects that they process are associated with type information. I also note that on page EOLASTX-0000000024, there is an "IMG3D" tag that resembles the EMBED tag disclosed in the

patent. The IMG3D tag includes the parameters that the '906 patent identifies for the EMBED tag, except for the "TYPE=" parameter.

811. On page EOLASTX-0000000030, there is an abstract dated January 28, 1994 entitled "Medicine Meets Virtual Reality II: Interactive Technology & Healthcare - The Virtual Embryo: VR Applications in Human Developmental Anatomy." This abstract only shows Mosaic's interoperation with a single volume visualization program (presumably VIS). It does not show type information used to identify an executable applications.

812. Nothing in this document suggests direct interaction with an embedded object as opposed to interaction through a separate panel window.

813. Finally, these documents do not indicate the extent to which automatic invocation was supported.

814. **EOLASTX-0000000034:** This document appears to be an inventor's notebook. There is one entry on page EOLASTX-0000000061 of the document which is purportedly dated "9/7/93" under the heading "Mtg w/ MDD." This appears to show certain concepts that are disclosed in the '906 patent specification, such as "ToolTalk," visualization, and a "low cost PC/Windows" or "X terminal" connected over a network to a higher priced server. However, again, there is nothing on this page that shows type information used to identify an executable application, and nothing to show direct interaction with an embedded object. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used. I note that Bina Exhibits 4 and 7 clearly disclose more information about embedded, inline applications of the type alleged to infringe by Eolas (e.g. PDF-like files and videos) than the subject notebook page. Likewise, pages from McRae Exhibit 37 (e.g. pages starting with the message by Chris McRae dated June 25, 1993 through the end of the

exhibit), which consist of materials from public www-talk postings, show far more detail of the invention than the alleged evidence cited by Eolas.

815. **EOLASTX-0000000169:** This document is an email press release dated August 1994. While the document discloses various types of applications that could be used in connection with the "embedded program objects" concept, it does not disclose that these applications would be identified by any form of type information and it does not disclose direct interaction with an embedded object. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

816. **EOLASTX-0000000171:** This includes a document from Eolas that appears to be dated 12/17/96, and therefore cannot show reduction to practice of type information before the filing date of the '906 patent. To the extent that Eolas asserts this document pre-dates the filing of the '906 patent, it nonetheless does not disclose type information. At most, it shows that the inventors thought to use multiple executable applications with Mosaic (such as VIS and an MPEG movie player), but it does not show any form of type information that can be used to identify which executable application. At page EOLASTX-0000000175, the document includes an announcement dated October 27, 1993 from Christopher McRae for a conference. The document notes that named inventor David Martin will be in attendance, but does not provide any technical or implementation details of his work. Nothing in this document suggest direct interaction with an embedded object.

817. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

818. **EOLASTX-0000000178:** This is an email thread from around June 22, 1993 between two of the named inventors. Of relevance to the patents-in-suit, David Martin wrote: "A World-Wide-Web (WWW) server, with support for Hierarchical

Data Format for the exchange of scientific data and the visualization of said data via applications developed both at NCSA (e.g. Mosaic – the WWW browser – and Collage – a cooperative work system) and at UCSF (e.g. VIS)." This disclosure is vague and may not even be directed to the disclosed embodiment of the '906 patent at all, let alone show a reduction to practice of type information. Nothing in this document suggest direct interaction with an embedded object. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

819. **EOLASTX-0000000180:** This is an email thread from on or about November 3, 1993, and shows named inventor Cheong Ang running into bugs in his efforts to show VIS outputs within XMosaic. This email is focused only on configuring XMosaic to interoperate with VIS, and does not show other executable applications or type information that can be used to identify an executable application. Nothing in this document suggest direct interaction with an embedded object as opposed to interaction through a separate panel window. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

820. **EOLASTX-0000000182:** This is an email thread dated on or about November 9, 1993 in which David Martin proposes to Cheong Ang a possibility for drawing the VIS output into Xmosaic. Again, this email is focused on configuring XMosaic to interoperate with VIS, and in fact shows that this is a "hard-coded" implementation, because the *execve* function in the source code accepts "vis" as a parameter. This source code, if implemented, would need to be changed and recompiled if one wanted to use an executable application other than VIS. Nothing in this document suggest direct interaction with an embedded object as opposed to interaction through a separate panel window. It also does not show other claim

limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

821. **EOLASTX-0000000184:** This is an email dated June 23, 1993 from Martin to Doyle stating "I should probably include some description of the multimedia aspects of Mosaic and the presumed interface we would provide; for example, combining the MPEG/JPEG viewing, output from VIS, output from RightPages, examples of the Hierarchical Data Format (HDF) from NCSA being utilized in Mosaic." Again, this does not disclose any form of type information that could identify an executable application. Nothing in this document suggest direct interaction with an embedded object as opposed to interaction through a separate panel window. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

822. **EOLASTX-0000000185:** This document includes a string of emails on or about October 1993 between Doyle and the Mosaic development team at the University of Illinois in which Doyle proposes that they collaborate. Although Doyle writes that "we are about 3 weeks away from demonstrating dynamic control of 3-D medical volume visualization through Mosaic," this document does not disclose any form of type information that can identify an executable application. Furthermore, nothing in this document suggest direct interaction with an embedded object as opposed to interaction through a separate panel window. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

823. **EOLASTX-0000000189:** This is a letter dated January 21, 1994, from Joseph Hardin at the University of Illinois in which NCSA agrees to collaborate with Doyle, but does not include any technical or implementation details, and so does not show reduction to practice of type information or direct interaction. It also does not

show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

824. **EOLASTX-0000000190:** This is a draft proposal to NSF/ARPA/NASA for a Digital Library project titled "Embedded Visualization Objects for Knowledge Access, Creation and Management through the World Wide Web," dated January 25, 1994. On page EOLASTX-0000000199, this document refers to type information: "Upon receiving a service request, the W3 server will establish a connection with the client, and transmit the requested digital objects (documents, sound clips, images etc.) with their respective tags in the headers. The client will read the headers, and based on the types of data, decide the appropriate actions. For example, a W3 server will reply to an image transfer request from a client by sending a header containing /img? as the data type followed by the image data. The client will peek into the header and invoke its image handling function(s)." However, this disclosure differs from the use of type information disclosed in the '906 patent, which shows that the inventors did not have a specific, settled idea in mind, but instead had a general goal or research plan they hoped to pursue. Even by March 31, 1994, when the paper shown at EOLASTX-0000000020 was submitted, the inventors still did not have a specific, settled idea in mind for using type information; as discussed above, they still had in mind the hardcoded IMG3D tag, which did not use type information. Likewise, nothing else in in EOLASTX-0000000190 shows a reduction to practice: at best, there is a suggestion that the named inventors thought of configuring Mosaic to operate with applications other than VIS, but there is no disclosure of how type information would be used to identify which of those applications would be used for a given type of object. Nothing in this document suggest direct interaction with an embedded object as opposed to interaction through a separate panel window. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

825. **EOLASTX-0000000249:** This is an email dated April 1, 1994 in which Michael Doyle asks the other named inventors for "the earliest notation that you may have in your work notebook relating to any discussion we may have had about embedding Vis within an HTML document." This email does not provide any technical or implementation details that would show a reduction to practice of type information. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used. Furthermore, I note that Doyle asked for a date for the earliest notation for embedding Vis within an HTML document, not any other executable applications that would handle other data types, and not a notation for anything related to type information. Finally, nothing in this document suggest direct interaction with an embedded object as opposed to interaction through a separate panel window.

826. **EOLASTX-0000000250:** This is a fax dated December 13, 1993. Attached to the fax at page EOLASTX-0000000253 is the abstract document entitled "Medicine Meets Virtual Reality II: Interactive Technology & Healthcare – The Virtual Embryo: VR Applications in Human Developmental Anatomy" which I already discussed above. Also attached to the fax at page EOLASTX-0000000257 is a video transcript that discloses the use of Mosaic with VIS, but not other executable applications that would handle other data types, and does not demonstrate a reduction to practice of type information. Nothing in this document suggest direct interaction with an embedded object as opposed to interaction through a separate panel window. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

827. **EOLASTX-0000000271:** This is an email from Michael Doyle dated November 30, 1993, in which he states "I need a stereo-pair set of images which show the integrated Vis/Mosaic demo, grabbed from the screen and converted to TIF. I

need this by 12 noon today. There should be enough text around the Vis window so that it is obvious that it is embedded in a true Mosaic document." Again, this only describes VIS, not other executable applications that would handle other types of data, and does not show a reduction to practice of type information. Nothing in this document suggest direct interaction with an embedded object as opposed to interaction through a separate panel window. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

828. **EOLASTX-0000000272:** This email is a reply from David Martin to Michael Doyle, dated April 2, 1994, in response to an email I described above (EOLASTX-0000000249). David Martin states that the "earliest record I have is 9/7/93." This email does not provide any technical or implementation details that would show a reduction to practice of type information. Furthermore, I note that Michael Doyle asked for a date for the earliest notation for embedding Vis within an HTML document, not any other executable applications that would handle other data types and not for notations involving type information. Nothing in this document suggest direct interaction with an embedded object as opposed to interaction through a separate panel window. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

829. **EOLASTX-0000000273:** This is an abstract dated January 27, 1994 for a "Medicine Meets Virtual Reality II" conference. This document describes the integration "through NCSA Mosaic and the World Wide Web, of text-based, image, audio, and video data with real-time interactive control of high-performance visualizations embedded within Mosaic documents." At best, this discloses various categories of data, but does not disclose type information or executable applications that would be identified based on type information. Nothing in this document



suggest direct interaction with an embedded. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

830. **EOLASTX-0000000277:** This document is undated, and so I cannot opine on whether it shows conception or reduction to practice of type information before the filing date of the '906 patent. In any event, the page EOLASTX-0000000281 includes vague disclosure about the use of Mosaic for visualization of anatomical volume data, but there is no disclosure of type information in this document. Nothing in this document suggest direct interaction with an embedded object. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

831. **EOLASTX-0000000283:** This document is a declaration from the [906-PH] prosecution history in which the named inventors assert an invention date prior to April 14, 1994. Attachment A to the declaration is the paper that was shown at EOLASTX-0000000020, which I have already addressed above and which does not show a reduction to practice of type information or direct interaction. Attachment B is a "transcript of the audio portion and still photographs of a video tape presented to an audience of scientists prior to April 14, 1994." To the extent this refers to the video transcript at EOLASTX-0000000257, I have already discussed that above and it does not show reduction to practice of type information or direct interaction.

832. **EOLASTX-0000000286:** This is a declaration from the named inventors that was part of the prosecution history for [906-PH]. Attachment A is the transcript that was also located at EOLASTX-0000000257 and which I already discussed as not showing reduction to practice of type information. Also in the document, at page EOLASTX-0000000305, is an NSD/ARPA/NASA proposal for a Digital Library project. This is similar to EOLASTX-0000000190, and likewise does not show a reduction to practice of type information. At best, there is ambiguous disclosure that

the named inventors thought of configuring Mosaic to operate with applications other than VIS, but there is no disclosure of type information that would be used to identify which of those applications would be used for a given type of object. Nothing in this document suggest direct interaction with an embedded object as opposed to interaction through a separate panel window. It also does not show other claim limitations, such as what text formats might be parsed (such as the embed text format), or whether automatic invocation was used.

**d. Conclusions**

833. As shown by my review of the documents identified above, there is no evidence of a reduction to practice of type information before the filing date of the '906 patent.

834. As shown by my review of the documents identified above, there is hardly any discussion of type information at all, let alone facts that would show diligence towards a reduction to practice during 1994.

835. As shown by my review of the documents identified above, there is no evidence of conception or of reduction to practice for "enabling an end-user to directly interact" before the filing date of the '906 patent.

836. Finally, I describe above (in connection with Section 112) that certain claim limitations are not supported by adequate disclosure in the patent specification, at least to the extent that the claim limitations are interpreted to cover certain functionality. My review of the documents above does not show documents that provide the adequate disclosure missing from the patent specification. To the extent the claim limitations are interpreted to cover the functionality described above in connection with Section 112, it is my opinion that the documents above do not support conception, reduction to practice, or diligence of those claim limitations before the filing date of the '906 patent.

**X. Section 112**

837. Under the Patent Law, 35 U.S.C. § 112, the specification of a patent must contain, "a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same, and shall set forth the best mode contemplated by the inventor of carrying out his invention."

838. I am informed that the written description requirement must be satisfied by the specification. To satisfy the written description requirement, which is an objective test, the specification as filed with the USPTO must clearly allow persons of ordinary skill in the art to recognize that the inventor invented what is claimed.

839. I have reviewed the specification of the '906 and '985 patents (the specification for both patents is the same) and considered it in view of the asserted claims.

840. It is my opinion that the specification does not contain a written description of the inventions as claimed in the asserted claims and as interpreted by Eolas. A summary of the reasons for my opinion is provided below.

**a. The Specification Lacks An Adequate Written Description Of An "Embed Text Format" Other Than One Implemented Using Special Tags**

841. Having reviewed the specification, I find that it lacks support for an "embed text format." The specification refers to special tags or symbols (see '906 at col. 14), and during prosecution, the inventors characterized the embed text format as the EMBED tag and attempted to distinguish over the prior art by arguing that it is a "special tag" not found in the prior art. *See e.g.*, '906 patent at 12:54–13:36; '985 patent at 14:27; Applicants' Response, at 22 (Feb. 5, 2009) (identifying the EMBED tag disclosed in the specification as support for the "embed text format"); Declaration of Edward W. Felten, at ¶ 21–25 (Sept. 27, 2007) (accompanying Applicants' Response

(Sept. 27, 2007)) ("In the claimed '906 system, the browser instead used a special tag, the "embed text format", to specify that an embedded object should be included. Mosaic lacked the embed text format."); *see also* Notice of Intent to Issue a Reexam Certificate, at 8-9 (Sept. 27, 2005) ("'interactive processing' is invoked . . . in response to the browser application parsing an 'embed text format' (i.e., an 'EMBED' tag, see col. 12, line 60, '906 patent) . . ."). Regardless of what exactly the embed text format is, the specification does not provide adequate disclosure of an "embed text format" implemented using scripts, such as code written in JavaScript, as opposed to a special tag or symbol that is simply parsed by the browser application.

842. The term "embed text format" is not found in the original specification, nor is it a term of art generally known to a person skilled in the art at the time of the invention. Rather, it was coined by Eolas added by amendment long after the first patent filing. *See Applicants' Response*, at 1-2 (Aug. 6, 1996); Doyle, Tr: 319:15-17 ("Q: No, nothing specific. I'm just asking if -- did embed text format, was that a term of art or is that a term that you coined? A: That's a term that we coined in our -- in our -- or that we used in our patent specification.").

843. As I noted above, the example of an "embed text format" linked by Eolas to the disclosure in the specification is that of an "embed tag" in HTML. (*See* Figure 7B). This example is shown in Table II, ('906 patent, 12:58-65; '985 patent, 12:60-67):

```
<EMBED
  TYPE = "type"
  HREF="href"
  WIDTH=width
  HEIGHT=height
>
```

844. The "embed tag" shown above, and envisioned by the specification, is a short pre-defined tag in standard HTML format with angle brackets and attributes and contained within an HTML document. The embed tag has a TYPE attribute that

specifies the same MIME type information that web browsers at the time, such as Mosaic, used to identify helper applications, a HREF attribute, which is a standard HTML attribute commonly used at the time to specify the location of a file, and WIDTH and HEIGHT attributes, which were standard HTML attributes commonly used at the time to specify the width and height of an image.

845. As I noted above, I understand that Eolas now asserts that an "embed text format" is "text that initiates processing" and has accused code contained in a HTML document and written in JavaScript, a client-side scripting language that is interpreted by the browser, of satisfying the requirement of an "embed text format."

846. It is my view that the specification does not support the use of client-side scripting language, such as JavaScript, to implement the "embed text format." Among other reasons, explained below, I say this because the use of a scripting language in this way, and as HTML documents are interpreted in modern browsers, implies a much higher level functionality than the simple inline processing and output of predecessor web browsers – at a time when webpages themselves were much less sophisticated. Modern web browsers use a special interpreter for processing client-side scripts, such as scripts written in JavaScript, and render the HTML document in a tree-like structure called the DOM tree, which is also not disclosed or suggested by the specification. The JavaScript is often used to manipulate the DOM tree so that the web browser can render the DOM as the content shown in the web browser. This idea is not taught or suggested in the specification. In fact using scripts and self-modifying code as is implemented in modern browsers with the DOM was not what even the inventors testified they invented. See, e.g., Martin Tr. at 384:5-386:17 ("It was not typically used technology. It's a complicated technology and was not part of what we were doing...It's like trying to repair an airline in flight."); Ang Tr. at 591:19-592:5 ("Q. Did you contemplate implementing the invention using scripting language? A. When the

application of this patent, the '906 patent, was file, I was busy doing implementation with the embodiment that was described in this patent, so I wasn't having the thoughts of - of any other - I wasn't thinking about anything else but doing my job."), 679:21-680:12 ("no, the invention did not implement a script interpreter."); Martin Tr. at 460:11-16 ("Q. And did JavaScript exist at the time you came up with your inventions? A. No."). 6/30/11 Doyle Tr. at 467:11-20 ("Sometime in the Nineties I recall hearing that - that Netscape had created a language called JavaScript."), 495:16-25 ("Q. Did you disclose the appropriate software that you'd need to implement the embed text format in a different language than the HTML? A. No.").

847. JavaScript itself was not available before the filing date of the '906 patent. JavaScript was developed by an employee at Netscape and first introduced in beta releases of Netscape Navigator 2 in December 1995. *See Netscape and Sun Announce JavaScript, the Open, Cross-Platform Object Scripting Language for Enterprise Networks and the Internet*, December 4, 1995, available at:

<http://web.archive.org/web/20070916144913/http://wp.netscape.com/newsref/pr/newsrelease67.html>; Innovators on the Net: Brendan Eich and JavaScript, June 24,

1998, available at:

[http://web.archive.org/web/20080208124612/http://wp.netscape.com/comprod/columns/techvision/innovators\\_be.html](http://web.archive.org/web/20080208124612/http://wp.netscape.com/comprod/columns/techvision/innovators_be.html).

848. HTML is a simple markup language with generic semantics that are used to represent information on a document, whereas a client-side scripting language is a programming language with entirely different syntax that allows a web page to have different and changing content depending upon user input, environmental conditions and events, or other variables. *See, e.g., W3C, Client-side Scripting and HTML*, March 14, 1997, available at: <http://www.w3.org/TR/WD-script-970314>. A client-side scripting language can also be used to modify the HTML document "as it is being parsed." *Id.*

849. The Mosaic HTML parser modified by the inventors and described in their preferred embodiment was not capable of parsing HTML documents so they could be modified as they were being parsed.

850. The specification does not demonstrate to a person of ordinary skill in the art that the inventors contemplated or fully possessed, as of the filing date, an invention wherein the "embed text format" comprises a script, including JavaScript, as opposed to merely representing an HTML tag as disclosed in the specification. Thus, the written description is not satisfied by the specification as this limitation has been interpreted by Eolas.

**b. The Specification Lacks An Adequate Written Description Of An "Embed Text Format" Other Than One Located At The "First Location"**

851. I further find that the specification lacks support for an "embed text format" other than one located at the "first location" where the object is displayed. Each of the asserted claims of the '906 patent requires that the "embed text format [is] located at a first location in [the] first distributed hypermedia document," and that the object is "display[ed] . . . within a display area created at said first location within the portion of said first distributed hypermedia document being displayed . . ."

852. As discussed above, the only embodiment of an "embed text format" disclosed in the specification is the EMBED tag of Table II. *See* '906 col. 12:56-65. This tag does not include an attribute that specifies where the display area will appear in the displayed document. Like other HTML tags, the object is inserted in the same location as the tag itself. For example, the specification describes that the inventors modified the source code of the publicly available Mosaic browser, version 2.4, so that when the browser parsed an HTML file and identified the EMBED tag it would automatically launch VIS without a user clicking on a link, and the output from VIS would be displayed within the browser window at the location of the EMBED tag that was found in the HTML file. *See, e.g.,* '906 patent, 14:9-16:46. There

is no disclosure that the output from VIS would be displayed at a location other than the location of the EMBED tag found in the HTML file. It is a simple in-line processing of the input stream of text from the HTML file where the location of each element identified within the stream alone determines its location in the browser window.

853. During prosecution and the two reexaminations, the applicants attempted to distinguish their invention over the prior art by arguing that the prior art did not teach an embed text format that is located at the same location as where the object is displayed. *See, e.g.*, Applicants' Response, at 11 (June 2, 1997) ("Further, [in Mosaic] a display window is not created in the first hypermedia document at the location in the document of the embed text format as required by the claim."); Declaration of Edward W. Felten, at ¶¶ 51-52 (Sept. 27, 2007) (accompanying Applicants' Response (Oct. 1, 2007)) ("[Cohen's] LDESC tags cannot be the embed text format, because they do not satisfy the required claim element 'wherein said first distributed hypermedia document includes an embed text format, located at a first location . . .' This claim element requires that the embedded object be displayed at a location in the distributed hypermedia document (e.g., the Web page) that corresponds to the location of the embed text format within the document. . . . The LDESC tag does not appear in the document at the required location. Instead, the LDESC (link description) tag appears in the document file's prologue . . .").

854. Each of the asserted claims of the '985 patent requires that the "embed text format . . . corresponds to a first location in the document." During prosecution of the '985 patent, the applicants similarly attempted to distinguish their invention over the prior art by arguing that the prior art did not teach an embed text format that is located at the same location as where the object is displayed. *See, e.g.*, Applicants' Response, at 18 (March 11, 2005) ("Further, there is no teaching in NoteMail of parsing an embed text format at a first location and displaying and



enabling interactive processing within the first location because, in NoteMail, the location of information is specified elsewhere, by the 'Format' data type."). When asked by the Examiner to identify support for the newly amended claims that included the "correspond to" language, the applicants did not acknowledge that the "corresponds to" language was a new element that required support, and simply listed as support the specification's EMBED tag – which as discussed earlier is located at the same location as where the object is displayed.

855. It is, therefore, my opinion that one of ordinary skill in the art cannot reasonably conclude, based on the specification of the patents-in-suit, that the inventors possessed, as of the filing date, an invention wherein the "embed text format" is located at a location other than one located at the "first location" where the object is displayed. Thus, the written description is not satisfied by the specification as this limitation has been interpreted by Eolas.

**c. The Specification Lacks An Adequate Written Description Of An "Embed Text Format" That Is Identified By A Method Other Than Parsing**

856. The specification disclosure for identifying an "embed text format" is limited to a simple in-line parsing of an HTML file and the attributes of the EMBED tag. It does not disclose that the identification of an "embed text format" can be accomplished by any method other than a straight in-line parsing of a stream of text.

857. For example, the specification describes that the inventors added routines in the HTMLparse.c file of Mosaic that checks whether each item parsed (e.g. word, tag or symbol) from a hypermedia document is an EMBED tag. If an EMBED tag is identified, a standard routine in HTMLParse.c is called to assign to the tag an enumerated type, which is an identifier with a unique integer associated with it. Once all of the text in the hypermedia document has been parsed, routines in a separate HTMLformat.c file are called to create an internal object representation of the EMBED tag with values for the "width and height of a window on the display to

contain an image, position of the window, style, URL of the image object, etc." *See, e.g., '906 patent, 14:9-16:46.*

858. To the extent Eolas argues that parsing is not required to identify an "embed text format," the specification does not provide a written description of such a technique to show that the inventors were in possession of such an invention, nor does the specification provide sufficient disclosure to a person of ordinary skill in the art of an "embed text format" that is identified by any method other than parsing.

**d. The Specification Lacks An Adequate Written Description Of  
"Executable Application"**

859. I have further found that the specification lacks support for an "executable application" other than that which, at all times, runs as a separate process from the browser. In other words, the specification does not support the "executable application" running within the same process as the browser.

860. Eolas has asserted that "executable application" means "any computer program code that is not the operating system or a utility, that is launched to enable and end-user to directly interact with data."<sup>63</sup> Dr. Doyle has suggested that the executable application need only be separate from the browser at the point when the executable application is invoked by the browser. *See Doyle June 30, 2011 Deposition, Tr: 513:15-22* ("Q: Is the executable application separate from the browser in your claims? Doyle: At the moment of invocation, or at the point of invocation, of automatic invocation, in the language of the claims, the browser is a separate entity from the executable application. The – the claim is silent on what relationship exists afterwards."). In my opinion, the specification does not provide a sufficient written description to support Eolas' broad definition or Dr. Doyle's suggestion.

---

<sup>63</sup> Joint Claim Construction Statement. (D.I. 479 at 4).

861. The specification of the patents-in-suit states that the executable application is started as a "child process" of the web browser: "The external application is started as a child process of the current running process (Mosaic), and informed about the window ID of the DrawingArea created in HTML format." (*See* '906 patent, 15:20-21; '985 patent, 15:21-21).

862. It is well known in the art that a "child" process runs as a separate process from its "parent" process. For example, the specification illustrates that the "application client" is a discrete and separate process from the "browser client." *See, e.g.*, '906 patent, 8:66-67 ("Client computer 200 includes processes, such as browser client 208 and application client 210.") & Figure 5; 10:17-19. The prosecution history confirms that the "executable application" runs as a separate process from the browser at all times, and cannot be merged into one process. *See* Notice of Intent to Issue Ex Parte Reexamination Certificate, at 8 (Sept. 27, 2005) ("In contrast, the instant '906 claims explicitly requires the "interactive processing" to be enabled by an "executable application" that is a separate application from the browser application.") (emphasis original); 60 ("In such case, the browser and the "executable application" merge into one program, and therefore cannot meet the requirement for a discrete "browser application" and a discrete "executable application" as claimed by the instant '906 patent."). Therefore, the disclosure in the specification of "executable application" requires that it run *separately* from the browser itself.

863. The specification further states that once the executable application is executed, ongoing communications are performed through *inter-process* communications (as opposed to, for example, *intra-process* communications):

864. Interprocess communication between the hypermedia browser and the embedded application program is ongoing after the program object has been launched. ('906 patent, 6:56-59; '985 patent, 6:56-59).

865. The preferred embodiment uses the XEvent interprocess communication protocol to exchange information between browser client and application client as described in more detail, below. ('906 patent, 8:57-65; '985 patent, 8:57-65).

866. As shown in FIG. 10, the browser process, Mosaic, communicates with the Panel process via inter-client communication mechanisms such as provided in the X-Window environment. ('906 patent, 16:29-33; '985 patent, 16:29-33).

867. In my opinion one of ordinary skill in the art cannot reasonably conclude, based on the specification that the inventors possessed, as of the filing date, an invention wherein the "executable application" operates within the *same* process as the browser after the "executable application" has been launched. Thus, the written description is not satisfied by the specification as this limitation has been interpreted by Eolas.

## **XI. Materiality of prior art**

868. I have reviewed the file histories of the patents-in-suit as well as the reexamination records of the two reexaminations of the '906 patent. My summary of those proceedings is described above. In view of those records, as well as my analysis of the prior art references detailed herein (including my attached claim charts), it is my opinion that the references contain information material to the patentability of the asserted claims of the patents-in-suit. As discussed below, each is not cumulative of the information of record already before the PTO, and but for the applicant/patentee's failure to disclose them, the PTO would not have allowed one or more claims based on anticipation and/or obviousness.

### **1. ViolaWWW and Vplot**

869. As noted above, I have reviewed several codebases related to ViolaWWW and vplot. It is my opinion that these references – including the various

codebases and versions of vplot, xplot, [Viola-5/12/93], [Viola-5/27/93], Viola-alpha, Viola-Feb Beta, Viola-Mar Beta, Pei Wei's August 1994 paper, and the www-talk postings and email correspondence between Pei Wei and Michael Doyle – are material to patentability. That is, they are not cumulative of information previously of record before the PTO, and but for the applicant/patentee's failure to disclose them, the PTO would not have allowed the claims of the patents-in-suit because each of the above-listed prior art references anticipates and/or renders obvious (either alone or in combination) one or more of the claims in the patents-in-suit.

870. Based on my review of the factual record, it is my opinion that ViolaWWW, vplot, and the Viola references described above were publicly demonstrated and in public use prior to October 17, 1993.

871. As described more fully above, it is my understanding and opinion that Viola (including [Viola-5/12/93] and [Vplot-sun]) was publicly demonstrated and used prior to October 17, 1993. This is corroborated by a May 4, 1993 email from Dougherty to Wei and a May 8, 1993 follow-up email from Dougherty to Wei and other staff at O'Reilly Associates that acknowledges a demonstration of Viola for Sun Microsystems engineers held on May 7, 1993, as well as emails exchanged on www-talk between Wei and Doyle. It is also confirmed by the "date modified field" for the [Viola-5/12/93] archive and my sampling of the [Viola-5/12/93] files, as well as by the testimony of Wei, Silvey, Dougherty, and Jacob, all of which I discuss within this report. It is also my understanding that the Federal Circuit held that a district court erred in finding that the demonstration of Viola was not a prior art public use. *Eolas Techs., Inc. v. Microsoft Corp.*, 399 F.3d 1325, 1335 (Fed. Cir. 2005).

872. As described more fully above, it is my understanding and opinion that [Viola-5/27/93] was publicly demonstrated and used prior to October 17, 1993. This is corroborated by a May 31, 1993 email from Wei to Kempf informing him of a public FTP posting of the [Viola-5/27/93] codebase, emails exchanged on www-talk

between Wei and Doyle, and other email exchanges. It is also shown by the "date modified field" for the [Viola-5/27/93] files, and the testimony of Wei, Silvey, Dougherty, and Jacob, which I discuss within this report. It is my understanding that the Federal Circuit held that a district court erred in finding that [Viola-5/27/93] did not anticipate or render the '906 patent obvious. *Eolas Techs., Inc. v. Microsoft Corp.*, 399 F.3d 1325, 1335 (Fed. Cir. 2005).

873. As described more fully above, it is my understanding that Pei Wei's August 16, 1994 paper was published and publicly available before the invention date claimed by the applicants of the patents-in-suit. This is confirmed by www-talk posts and emails exchanged between Wei and Doyle. Moreover, it is my opinion that Wei's 1994 paper describes and illustrates features that were themselves publicly demonstrated and in public use prior to October 17, 1993. As detailed more fully above, Wei's 1994 paper describes ViolaWWW and a plotting demonstration, "XPlot," that was publicly demonstrated by at least May 1993.

874. As I have also described more fully above, it is my understanding and opinion that the vplot executable application, [Viola-alpha], [Viola-Feb Beta], and [Viola-March Beta] were known or used by others in this country before October 17, 1993.

875. Based on my review of the prosecution and reexamination records, it is my opinion that ViolaWWW, vplot, and the aforementioned references, individually and/or in combination, are not cumulative of any references previously before the PTO. References that the applicant/patentee failed to disclose to the PTO include Wei's 1994 paper, the email exchanges between Wei and Doyle indicating that Viola was publicly demonstrated and in use in at least May 1993, and any other correspondence received by Doyle regarding Viola and Wei's invention.

876. At no time during the original prosecution of the '906 patent did the applicant disclose any information regarding the vplot executable application, ViolaWWW, or any other information related to Viola to the PTO.

877. As described more fully above, during the first reexamination of the '906 patent, the patentee provided a CD containing source code for Viola, but that disclosure was incomplete and the patentee did not disclose material information related to this prior art; information that was needed to enable the Examiner to review it effectively. Specifically, the PTO was not provided any tools, software, hardware, or search terms for analyzing, compiling, or identifying the relevant portions of the source code provided. For example, without the compiled binary code, and without a computer capable of executing that code, the PTO had no practical way to see the ViolaWWW browser in operation. The patentee also failed to disclose, for example, the VOBJF tag, plot.v, vplot, or and the other relevant search terms that, given the large volume of source code provided in the submission, that were necessary for the Examiner to review the material portions of the source code.

878. Accordingly, the Examiner did not search for, locate, and review the relevant portions of the [Viola-5/12/93] and [Viola-5/27/93] source code submitted by the patentee during the first reexamination of the '906 patent. It is evident from the Examiner's description of his review of [Viola-5/27/93] that he only reviewed non-material information contained on the CD relating to "clock.v". As discussed above, "clock.v" is a script and not an executable application, as can be found elsewhere in the reference. The Examiner did not find and examine, for example, the files "testPlot.hmm1" and "plot.v," which were contained within [Viola-5/27/93]. As noted above, these files relate to the plotting demonstration illustrated in Wei's 1994 paper and that was publicly demonstrated and in public use by at least May 1993. In my opinion, if the Examiner had reviewed these files and the plotting demonstration,

he would have found every limitation of one or more of the asserted claims of the '906 patent, and as a result would not have allowed the claims.

879. The PTO likewise failed to independently examine [Viola-5/12/93] and [Viola-5/27/93] during the second reexamination of the '906 patent and the prosecution of the '985 patent, as described more fully above. In particular, during the second reexamination of the '906 patent, the Examiner relied on the analysis of [Viola-5/27/93] made by the examiner in the first reexamination, but rejected certain claims of the '906 patent as anticipated by Wei's 1994 article. In response, the patentee filed a declaration from Doyle swearing behind the date of that article. I am aware of the evidence cited above, including emails exchanged between Wei and Doyle himself, that ViolaWWW and vplot were publicly demonstrated and in use prior to the date sworn by Doyle, including, for example, the plotting demonstration given in May 1993. The patentee provided no other basis for overcoming the rejection in light of Viola, and the Examiner did not undertake any other review of [Viola-5/12/93] or [Viola-5/27/93].

880. It is therefore my opinion that the Examiner failed to review material information concerning ViolaWWW and vplot during the prosecution of both the '906 and '985 patents and during both reexaminations of the '906 patent, and this information was not cumulative of the record already before the PTO. Even though one reference, DX 37, was considered by the examiner in the first reexamination, the Examiner did not consider the material portion of the source code. The PTO has never fully considered [Viola-5/12/93] or [Viola-5/27/93] with reference to the prosecution of the '906 or '985 patents, and has not specifically considered "testPlot.hmm1" or "plot.v" at any time.

881. Based on my analysis of the references and giving the claims their broadest reasonable construction, it is my opinion that the preponderance of the evidence shows that, but for the patentee's failure to disclose material information



about ViolaWWW, vplot, and the Viola references as described above, the PTO would not have allowed one or more asserted claims of the patents-in-suit.

882. As detailed above and in the attached claim charts, [Viola-5/12/93], [Viola-5/27/93], [Viola-alpha], [Viola-Feb Beta], and [Viola-March Beta], each independently and/or in combination, discloses each and every element of claim 1 (and dependent claims 2, 3, and 11), and claim 6 (and dependent claims 7, 8, and 13) of the '906 patent, and claim 1 (and dependent claims 2-11), claim 16 (and dependent claims 17-19), claim 20 (and dependent claims 21-23), claim 24 (and dependent claims 25-27), claim 28, claim 36 (and dependent claims 37-39), and claim 40 (and dependent claims 41-43) of the '985 patent. Likewise, as detailed above in the attached claim chart, Pei Wei's 1994 paper in combination with the email correspondence between Pei Wei and Michael Doyle disclose each element of the asserted claims of the '906 and '985 patents. Moreover, the Examiner determined that Wei's 1994 paper discloses every element of claims 1-10 of the '906 patent; as I detail above, the "plotting demo" invention described and illustrated in that paper was publicly demonstrated and in public use prior to October 17, 1993. It is therefore my opinion that each of the aforementioned references individually anticipates and/or in combination renders obvious one or more of the asserted claims of the '906 and '985 patents.

883. By contrast, the PTO did not examine any references that disclosed each and every limitation of the claims. As noted above, the Examiner allowed the claims of the '906 patent based on his conclusion that neither "clock.v" in [Viola-5/27/93] nor various other cited prior art satisfied the "executable application" limitation. However, as described above, Viola, including "plot.v" in [Viola-5/27/93], teaches the use of an "executable application" such as vplot, as demonstrated by the plotting demonstration given by Wei in May 1993. In other words, the references that the applicant/patentee submitted to the PTO were less material than the references

concerning Viola and vplot that the applicant/patentee failed to disclose to the PTO because, unlike the withheld references, the submitted references do not disclose every limitation. Accordingly, had the applicant/patentee disclosed vplot, plot.v, or any other material information about Viola to the PTO, the PTO would have found the "executable application" limitation to have been demonstrated and in public use prior to October 17, 1993 and rejected the claims based on anticipation and/or obviousness. It is my opinion that, but for the applicant/patentee's failure to disclose Viola to the PTO during the original prosecution and failure to disclose material portions of [Viola-5/21/93] and [Viola-5/27/93] during either reexamination of the '906 patent or the prosecution of the '985 patent, the PTO would not have allowed one or more of the asserted claims of the patents-in-suit.

884. It is therefore my opinion that the vplot and Viola references described herein are information material to patentability.

885. In addition, based upon my review of the factual record, including the file histories and reexamination proceedings, I understand that the Adobe Acrobat related art (e.g. Acrobat/PDF related www-talk postings and the Acrobat tagging mechanism such as shown in [Adobe93]) was not disclosed to the PTO in any original prosecution or reexamination of the patents-in-suit. And as discussed above and detailed elsewhere in this report (see, e.g., paras. 585-592), the Acrobat related prior art is not cumulative of the references before the PTO because it, alone or in combination with other references, discloses every limitation of one or more of the asserted claims of the patents-in-suit as alleged by Eolas in its infringement contentions against Adobe--indeed, Eolas admits it is material by its allegations alone.

## **2. Acrobat**

886. Acrobat is an authoring environment, and the www-talk postings discuss combining the Acrobat technology with a browser, which Eolas argued was a

reason a person of ordinary skill in the art would have been motivated to combine other prior art references, and Eolas itself points to the same functionality in Acrobat as evidence of infringement that existed more than a year before the patent was filed. (See, e.g., Response to Office action 8/9/96 at 18 (interaction); Response to Office action 12/23/97 at 3, 14, 17, 18, 19, 22, 24 (user editing facilities).)

887. Moreover, the inventors were aware of the Acrobat prior art as well as the www-talk postings suggesting it be combined with a web browser as they not only were beta testers, but monitored the www-talk postings discussing the combination, and even their EMBED tag, which is an embodiment of the embed text format, is remarkably similar to the Acrobat format. (See [Adobe93] at 79.) As such, but for the failure to disclose the Acrobat related prior art, and in view of Eolas's infringement allegations, the Examiner would not have allowed those claims as anticipated and/or obvious.

### **3. MediaView**

888. Based upon my review of the factual record, including the file histories and reexamination proceedings, I understand that the MediaView reference has never been disclosed to the PTO in any original prosecution or reexamination of the patents-in-suit. And as discussed above and detailed in the attached claim charts, MediaView is not cumulative of the references before the PTO because it, alone or in combination with other references, discloses every limitation of one or more of the asserted claims of the patents-in-suit. As such, but for the failure to disclose MediaView, the Examiner would not have allowed those claims as anticipated and/or obvious.

## **XII. Secondary considerations of non-obviousness**

889. I have reviewed Plaintiff Eolas's Response to Defendants' Interrogatory No. 6, which identifies the alleged facts surrounding secondary considerations that

Eolas contends supports the non-obviousness of the Patents-in-Suit. I have also reviewed the materials cited in Eolas's Responses to Interrogatory No. 6.

890. Here, I'll confine my response to the contentions found in Eolas's Response to Interrogatory No. 6 and reserve the right to supplement my analysis based on materials and information subsequently located during discovery.

891. Generally, it is my opinion that the materials identified in Eolas's Responses to Interrogatory No. 6 do not provide sufficient evidence of non-obviousness of the Patents-in-Suit. My opinions are described in detail below.

### **1. No Evidence of Commercial Success**

892. Eolas's Response to Interrogatory No. 6 contains flaws in the evidence and its reasoning. First, Eolas confuses the success of a company or product with the success of the Patents-in-Suit. In doing so, Interrogatory Response No. 6 does not establish any connection ("nexus") between commercial success and the Patents-in-Suit. This error is the exact same error that was noted by the U.S. Patent Office during reexamination proceedings. *See e.g.*, [831-PH], September 27, 2005 Office Action at 39-44.

893. Specifically, Eolas presumes a number of products practice the Patents-in-Suit without providing any evidence to support its presumption. In particular, Eolas assumes that Microsoft's Internet Explorer is an "embodying product" of the Patents-in-Suit. In doing so, Eolas never compares the claims of the Patents-in-Suit to the Microsoft Internet Explorer, nor addresses any changes in that product and the way it operates since the prior trial. The only evidence Eolas appears to present is a jury verdict that found infringement of the '906 Patent based on evidence available at that time. I am informed by counsel that the verdict, however, was remanded for a new trial by the Federal Circuit on issues involving the validity of the '906 Patent. *See e.g., Eolas Techs. Inc. v. Microsoft Corp.*, 399 F. 3d 1325, 1335 (Fed. Cir. 2005). Microsoft and Eolas entered into a settlement agreement following the Federal Circuit decision.

In that settlement, Microsoft made no admission of liability, including no admissions regarding Eolas's allegations of infringement of the '906 Patent. *See, e.g.,* EOLASTX-0000009900-906 at 902.

894. In addition, Eolas makes no effort to address the competitive landscape at the time, Microsoft's distribution channel, or barriers competitors faced in bringing any competing browser to market, nor does Eolas address the simplicity and ease of use of the World Wide Web generally, its rapid adoption, and increased bandwidth that improved network capabilities facilitated. Additionally, with regard to the '985 Patent, the remanded jury verdict predates the '985 patent issuance.

895. Similarly, on page 10-11 of Interrogatory Response No. 6, Eolas makes the cursory statement that "the infringing products of the various defendants in this case have also been commercially successful." Here again, Eolas provides insufficient analysis or evidence to support this statement. In fact, Eolas does not specifically identify any of the allegedly infringing products, or any of the non-infringing uses of those products or any combination of alleged infringing plugin technologies. Eolas did not invent either the idea of compound documents, or of compound documents shared over a network. Also, Eolas does not bother to attribute any alleged success to an embed text format that cannot even be seen by a user.

896. On page 11, Eolas also mistakenly believes that the "willingness" of certain entities such as, Oracle America, Inc., JPMorgan Chase & Co., New Frontier Media, Playboy Enterprises, Texas Instruments, and Argosy Publishing, Inc., to license the patents-at-issue, "implies a presumption of patent validity." I understand from counsel that information relating to settlements from litigation may not be admissible at trial. The fact that certain entities chose to license the Patents-in-Suit does not by itself provide any evidence regarding the validity or commercial success of the Patents-in-Suit. To the extent it shows anything, it is an economic decision whether to continue with litigation through trial and then on appeal, where it may

cost much more than the relatively low settlement amounts I understand have been entered. Moreover, review of the license agreements reveals that the licensees to these agreements did not concede infringement of the Patents-in-Suit. Nor did Eolas warrant the validity or enforceability of any of the claims of the Patents-in-Suit. *See e.g.*, EOLASTX-0000320079-109 at 098; EOLASTX-0000320110-135 at 123; EOLASTX-0000009914-919 at 917.

897. In fact if one examines the sole embodiment of the so-called embed text format in the patent, the EMBED tag, for which no effort has been made to tie its use to the commercial success of any specific product, it very closely resembles the tags used in Adobe's PDF format before the patents-in-suit were conceived and reduced to practice. Compare, for example, '906 patent 12:58-64 (Table II) to [Adobe93] at 78-80 (where it is mentioned that PDF files may eventually support video and audio objects):

TABLE II

```

<EMBED
  TYPE = "type"
  HREF = "href"
  WIDTH = width
  HEIGHT = height
>
    
```

Table 6.20 Image attributes

Key	Type	Semantics
<b>Type</b>	name	(Required) Resource type. Always <b>XObject</b> .
<b>Subtype</b>	name	(Required) Resource subtype. Always <b>Image</b> .
<b>Name</b>	name	(Required) Resource name, used as an operand of the Do operator. Name must match the name used in the XObject dictionary within the page's Resources dictionary.
<b>Width</b>	integer	(Required) Width of the source image in samples.
<b>Height</b>	integer	(Required) Height of the source image in samples.

898. Accordingly, it is my opinion that the evidence Eolas relies upon does not establish that any product embodies the Patents-in-Suit, or that any such

embodiment is the reason why those products are commercially successful. Therefore, it is also my opinion that Eolas has not established that commercial success, if any, is directly derived from the claimed inventions of the Patents-in-Suit.

## 2. No Evidence of Inability to Design Around the Claimed Invention

899. Second, in addition to presuming infringement of the Patents-in-Suit, Eolas, without providing any support for its allegations, also presumes that there is no design-around for the Patents-in-Suit.

900. Eolas makes the mistake of presuming there is no design-around for the Patents-in-Suit when it contends that Microsoft's click to activate update to Internet Explorer "does not avoid infringement of [the Patents-in-Suit]." Here, Eolas makes the mistake of presenting only cursory arguments regarding the infringement of the Patents-in-Suit. Eolas does not present any analysis or evidence of why Microsoft's click to activate would still infringe the Patents-in-Suit or would the use of another document structure that more closely resembles the tree type structure of, for example, a PDF file. See [Adobe93] at 50. Eolas, therefore, in my opinion, failed to establish that click to activate or other design alternatives is not a design around to the Patents-in-Suit.

901. Regardless of whether or not Microsoft's click to activate design-around was well received by certain industry critics, that fact does not establish inability to design around the Patents-in-Suit. Microsoft also created a second design-around that worked in conjunction with its click to activate design-around. In 2003, Microsoft discussed its click to activate design-around with certain industry players. ADBE0193980; *see also* Sundermeyer Tr. at 25:9-25. In conjunction with its click to activate design around, Microsoft also recommended a JavaScript solution for webpages. Sundermeyer Tr. at 55:7-25, 119:19-24. The JavaScript solution provided an alternate way to code webpages. *Id.*; *see also* ADBE0193973. Though Eolas now contends that the JavaScript solution infringes the patent, this solution was

promulgated by Microsoft and others, including Adobe, as early as 2003. However, Eolas, as I understand the evidence, never bothered to tell anyone its assertion that such a solution infringed the patent. *See, e.g.* Doyle Tr. (6/24/11) Exhibits 1, 2, 6, 7.

902. With respect to the JavaScript solution, Eolas contends that it infringes the Patents-in-Suit. Again, the flaw in Eolas's contentions is that no evidence or analysis of how the JavaScript solution practices every element of even at least one claim of the Patents-in-Suit and why that particular technique, which is not visible to a user, is the basis for the commercial success of the products.

903. It is therefore my opinion that the evidence presented does not demonstrate that the industry was unable to design around the Patents-in-Suit. In my view, it is simply a matter of inertia since the Web is standards-based and requires openness and a consensus. The industry settled on standards and alternatives to them, they were used widely, Eolas remained silent, and now claims, after more than 6 years of use of the standards and alternatives and Eolas's silence, that it would not have been possible to derive yet another solution. Indeed, I note that Eolas sued the Defendants the day the '985 patent was issued.

### **3. No Evidence of Long Felt Need and Skepticism of Experts**

904. With respect to the factors of long felt need and skepticism of experts, the evidence does not support a finding of either. Eolas attempts to establish that there was long felt need for the invention of the Patents-in-Suit by relying on emails exchanged in 1993 between Dr. Doyle and the individuals connected to the Mosaic project. Given the short duration between Mosaic's release date and the cited emails, it is my opinion that the emails do not establish any long felt need for the inventions of the Patents-in-Suit. In fact, some of the combinations of products now alleged by Eolas to infringe (such as the ability of a browser to display a PDF file) were discussed on the www-talk email thread. *See, e.g.,* Doyle Tr. (6/24/11), Exhibits 4 and 5.



905. More importantly, as I have discussed in other sections of this report, the ViolaWWW browser predates both Mosaic and the Patents-in-Suit. Additionally, as discussed in this report, the ViolaWWW browser anticipates the claims of the Patents-in-Suit. Thus, it is my opinion, that any need for the inventions of the Patents-in-Suit was first met by the ViolaWWW browser, and not the alleged inventions of the Patents-in-Suit.

906. Eolas also relies on emails in attempts to establish that there was skepticism by others. In particular, Eolas relies on the statements of a single individual, Marc Andreessen, as proof that there was skepticism by experts in the field. Review of the email, however, tells a different story. Bill Janssen interpreted Marc Andreessen's comments as identifying "specific problems to be solved" regarding the embedding of "images or animations or audio or text or whatever." PH\_001\_0000003398-399 at 398. Marc Andreessen's statements when read in conjunction with Bill Janssen's suggest that the cited email was a discussion regarding considerations – design choices – that must be taken into account when embedding objects into web pages, and not an argument against the embedding of objects. It is therefore my opinion the cited evidence does not support a conclusion that there was skepticism by others.

907. Finally, Eolas contends that the outcomes of a jury trial and two reexaminations provide secondary considerations of non-obviousness. I am informed by counsel that the outcome of jury trials and patent reexaminations are not a secondary consideration of non-obviousness. I therefore do not interpret the decisions of the U.S. Patent Office and a jury as evidence supporting non-obviousness of the Patents-In-Suit.

#### **4. No evidence of "Copying by others"**

908. Although Eolas contends that "copying by others" shows that the asserted patents are non-obvious, Eolas does not appear to provide any meaningful

support or evidence for such an assertion. The assertion is simply too lacking in evidence for me to opine.

### **My Compensation**

909. I am being compensated on an hourly basis. My hourly rate is \$400. I am also being reimbursed for reasonable and customary expenses associated with my work and testimony in this case. My compensation is not contingent on the outcome of this litigation or the specifics of my testimony.

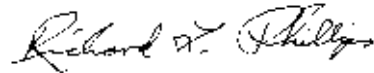
**Prior Expert Testimony**

910. 1996-97 Adobe Systems v. Quantel  
Wrote invalidity and non-infringement reports.  
Was deposed.  
Testified at trial.  
All five Quantel patents were invalidated.
911. 1990-00 Adobe Systems v. Heidelberger  
Was 30(b)(6) expert.  
Was deposed.  
Case settled.
912. 2000-02 Adobe Systems v. Macromedia  
Wrote invalidity and non-infringement reports on 3  
patents.  
Was deposed on 3 cases.  
Testified at 3 trials.  
Case decided for Adobe.
913. 2002-03 Corel Corporation v. Heidelberger  
Wrote invalidity and non-infringement reports.  
Was deposed.  
Case settled.
914. 2004-06 DreamWorks v. Sitrick  
Wrote invalidity and non-infringement reports.  
Was deposed.  
Summary judgment favoring DreamWorks.
915. 2005-06 Microsoft Corporation v. America Video Graphics (AVG)  
Wrote invalidity report.  
Case settled.

916. 2007-08 Witness Systems Inc. v. Nice Systems  
Wrote invalidity and non-infringement reports.  
Was deposed.  
Case settled.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

October 27, 2011  
Santa Fe, NM



Richard L. Phillips