Exhibit M

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent of:	Michael D. Doyle, David C. Martin, Cheong S. Ang
Patent No.:	5,838,906
Issued:	November 17, 1998
Filed:	October 17, 1994
Assignee:	Regents of the University of California
For:	DISTRIBUTED HYPERMEDIA METHOD FOR AUTOMATICALLY INVOKING EXTERNAL APPLICATION PROVIDING INTER-ACTION AND DISPLAY OF EMBEDDED OBJECTS WITHIN A HYPERMEDIA DOCUMENT

REQUEST FOR *EX PARTE* REEXAMINATION

UNDER 35 U.S.C. § 302 AND 37 C.F.R. § 1.510

Mail Stop *Ex Parte* Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

December 22, 2005

Dear Sir:

Reexamination of U.S. Patent No. 5,838,906 ("the '906 patent"; copy attached in double column format as Exhibit A) is requested pursuant to 35 U.S.C. § 302 and 37 C.F.R. § 1.510 based on the prior art printed publications cited herein. Copies of the cited references are attached as Exhibits to this Request. Because the filing date of the '906 patent is prior to November 29, 1999, the statutory *inter partes* reexamination option is not available, and therefore *ex parte* reexamination is requested. (*See* MPEP § 2601.) The '906 patent is still in force. In accordance with 37 C.F.R. §§ 1.33(c) and 1.510(b)(5), this request is being served in its entirety on the attorney of record in the '906 patent.

I. INTRODUCTION

The patent owner has compromised the pending Director-instituted reexamination of the '906 patent (control no. 90/006,831, "the '831 reexamination") by failing to tell the Examiner about prior art in its possession that would have materially changed the prior art landscape in front of the Examiner. The patent owner has prior art that (1) fills in the gaps perceived by the Examiner in the art before him, (2) invalidates each of the claims of the patent, and (3) refutes contentions made by the patent owner and its expert to the Examiner. The present reexamination request places this previously unconsidered art before the Examiner for consideration, thereby presenting substantial new questions of patentability.

The Withheld Art Fills in Gaps Noted by the Examiner

The patent owner possessed but failed to tell the Examiner about a prior art reference (Exhibit B, hereinafter "Janssen," which was a public posting to the www-talk email list) that fills in the gaps the Examiner noted in Raggett II (Exhibit C, hereinafter "Raggett II"). The withheld Janssen prior art reference could hardly be more closely tied to Raggett II. The Janssen reference responded to and incorporated the Raggett II reference, quoting parts of it, and expanded upon it in ways that provide precisely what the Examiner deemed to be missing from Raggett II.

There is no doubt that the patent owner was aware of this Janssen reference that expands upon Raggett II. Exhibit B is a bates numbered version of the Janssen posting, which was provided to the patent owner long ago during its patent infringement action against Microsoft Corporation (No. 99-626, N.D. Ill.). Yet, there is no indication in the public record that the patent owner brought this key reference to the Examiner's attention.

This previously unconsidered Janssen posting is highly material because it fills each perceived gap identified by the Examiner in the Raggett II posting, and refutes positions the patent owner took with respect to the Raggett II posting. For example, the Examiner agreed with the patent owner that the art taught away from combining the Toye prior art patent with Raggett

II and the other cited art. The Janssen reference, however, completely fills this gap because it explicitly incorporates, responds to, and expands upon the Raggett II posting. In other words, the Janssen posting does not merely teach the combination with Raggett II that the Examiner found lacking; it effects and constitutes that combination.

The Withheld Prior Art Invalidates the '906 Patent Claims

In addition to filling in the gaps noted by the Examiner, the Janssen reference discloses each other limitation in the broader claims. When understood in the context of the software systems it expressly concerns (e.g., the Mosaic browser and X Window System), the Janssen reference anticipates claims 1 and 6 of the '906 patent. Further, combined with this related art and Raggett II, it establishes a prima facie case of obviousness of claims 1 and 6; and, combined with other related prior art, it renders obvious the remaining claims. As such, the Janssen posting raises substantial new questions of patentability.

The patent owner also withheld an important prior art publication co-authored by two of the patent applicants. Specifically, patent applicants Doyle and Ang (with others) co-authored a publication, dated before the critical date, describing a version of the "Visible Embryo Project" software shown in Figure 9 and discussed at length in the '906 patent. This withheld prior art 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. Despite being co-authored by two of the three patent applicants, being described in part in the '906 patent, and being published more than one year before the application filing date, the patent owner did not submit this prior art publication to the Patent Office.

To illustrate the importance of these withheld Janssen and Doyle-Ang references, the following features chart summarizes two approaches to mapping prior art to claims 1-10 of

the '906 patent. Detailed claim charts are included at the end of this request for *ex parte* reexamination.

Feature	Prior Art
browser (all claims)	Janssen **OR** Mosaic Admitted Prior Art ("APA") (as part of combination)
launching separate executable application (all claims)	Janssen **OR** Mosaic APA (as part of combination)
EMBED tag (all claims)	Janssen **OR** Raggett II (as part of combination)
automatic invocation of separate application (all claims)	Janssen **OR** Raggett II (as part of combination)
in-line display in browser of foreign-format data (all claims)	Janssen **OR** Raggett II (as part of combination)
interactive processing of foreign-format data (all claims)	Janssen **OR** Janssen (as part of combination)
interactive control of separate application via ongoing inter-process communication (claims 2-5 and 7-10)	X Window System art (as part of combination)
distributing separate application between client and server (claims 4, 5, 9, and 10)	Doyle-Ang reference about Visible Embryo Project software (as part of combination)

Table 1: Summary of Prior Art Mappings

The Withheld Prior Art Refutes Positions Taken by the Patent Owner

These previously unconsidered prior art references (Janssen and Doyle-Ang) relate to software for the X Window System, just like the Raggett II prior art reference and the embodiments described in the '906 patent. A person of ordinary skill in the art necessarily would have read and understood these prior art references in the context of software for the X Window System, including the Mosaic browser and other available applications for the X Window System. Therefore, to set the necessary technical background for a proper understanding of these X Window System/Mosaic prior art references, Requester submits herewith further prior art references that document the capabilities of the X Window System and of certain interactive Mosaic helper applications (also called external viewers or simply viewers) available for the X Window System in September 1993. Section IV provides citations to some

specific portions of this extensive documentation that are relevant to the claims of the '906 patent.

The patent owner gave the Examiner none of this necessary background information. Instead, it made assertions to the Examiner that could not have been made had it provided the documentation now submitted by the Requester.

For example, in the '831 reexamination, the patent owner and its expert purported to summarize the state of the art for Mosaic browser technology and Mosaic helper applications. (*See, e.g.*, Response received October 12, 2004, pages 1, 5-7, 11-13, and 15; Response received May 11, 2004, pages 1, 4-9, 11-15 and 18; Declaration of Edward Felten signed October 6, 2004, ¶¶ 18-20; and Declaration of Edward Felten signed May 7, 2004, ¶¶ 8-11, 16, 26, 31, 33, 36-55, 60, and 61.) They urged that the Raggett materials describe a browser operating with non-interactive rendering applications, which would simply return a static image when invoked, and then terminate. They did not tell the Examiner, however, that several interactive helper applications (such as Xv, Xdvi, and Ghostview) were default helper applications for Mosaic in September 1993. These helper applications by default provide interactive, continuing processing of image data, as opposed to processing in which a viewer starts, accepts a file by standard input, and returns a static image, with no interactivity. The submitted interactive helper application documentation and other background X Window System documentation, which the patent owner and its agents failed to submit to the Patent Office, call into question the completeness and accuracy of representations of the patent owner and its agents.

In sum, by providing a more complete prior art landscape that the patent owner chose not to present, the present reexamination request fills in the perceived prior art gaps noted by the Examiner in the pending reexamination, and presents several substantial and compelling new questions of patentability.

II. IDENTIFICATION OF CLAIMS FOR WHICH REEXAMINATION IS REQUESTED

Reexamination of claims 1-10 of the '906 patent is requested. Claims 1 and 6 are the only independent claims. Claims 1-5 are method claims, and each of claims 2-5 depends consecutively from the preceding claim. Claims 6-10 are apparatus counterparts to claims 1-5, respectively. None of apparatus claims 6-10 is patentably distinct from its counterpart method claim, as admitted by the applicants during prosecution. (*See* Amendment mailed June 2, 1997, in the '906 patent file history, page 26 (the apparatus claims are "of the same scope as claims 1-5"); *see also* Office action mailed February 26, 2004, in the '831 reexamination, pages 7 and 9.)

III. PROCEDURAL BACKGROUND

The original application for the '906 patent was filed on October 17, 1994, as U.S. Patent Application Serial No. 08/324,443. The '906 patent is assigned to The Regents of the University of California ("The Regents") and is exclusively licensed to Eolas Technologies Incorporated ("Eolas"). Eolas and The Regents are pursuing a patent infringement action against Microsoft Corporation. (No. 99-626, N.D. Ill.) Certain aspects of this case have been considered by the Court of Appeals for the Federal Circuit. *See Eolas Techs. Inc. v. Microsoft Corp.*, 399 F.3d 1325 (Fed. Cir. 2005).

The Patent Office instituted the '831 reexamination on October 30, 2003. Initially, it rejected claims 1-3 and 6-8 of the '906 patent as being unpatentable over a combination of prior art including the Raggett II posting and Mosaic browser technology admitted to be prior art ("APA Mosaic"). The Patent Office added two other references to reject claims 4, 5, 9, and 10. The patent owner's admissions concerning prior art appear in the '906 patent itself as well as in various patent owner submissions to the Patent Office during the '831 reexamination.

The Patent Office considered at length the Raggett II reference in the '831 reexamination. Raggett II is a message entitled "HTML+ support for eqn & Postscript," authored by Dave Raggett and distributed to subscribers of the www-talk email list on June 14, 1993. The www-talk email list was a public, archived, and indexed discussion forum, whose contents were widely disseminated and publicly available on the Internet prior to the October 17, 1993, critical date of the '906 patent. The Raggett II posting thus qualifies as prior art against the '906 patent as a printed publication under 35 U.S.C. § 102(b). (M.P.E.P. § 2128.)

The Mosaic web browser application was itself (or "natively") able to display and process certain formats of data, but not all data formats. Raggett II describes techniques for supplementing a Mosaic browser in order to support data in a format that is "foreign" to the browser. It describes using an "EMBED" tag "to embed foreign-formats inline in the HTML+ source" (Ex. C, page 1), and gives the following example:

> <H2>A example of an equation</H2>
> <EMBED TYPE="text/eqn">zeta (s) ~=~ sum from k=1 to inf
> k sup -s~~~ (Re s > 1) </EMBED>

(*Id.*)

Raggett II describes the foreign-format data as being either internal or external to the HTML document. The above example shows the former, in which equation data is internally embedded within the EMBED tag. Or, "you can also put the foreign data in a separate file referenced by a URL." (Ex. C, pages 1 and 2.) Thus, Raggett II describes the EMBED tag in an HTML+ document including a URL that specifies the location of foreign data external to the HTML+ document.

Whether internal or external, the foreign-format data in Raggett II has type information associated with it. The Mosaic browser uses this type information to identify the format of the data. In the above quoted example set forth in Raggett II, the browser identifies the

format of the data from the "TYPE" attribute – "TYPE=text/eqn". (Ex. C, page 1.) In this example, the type attribute is "specified as a MIME content type." (*Id.*)

Once the foreign format is identified using the associated type information, a program needs to be identified that is capable of processing data in that format. Raggett II describes several techniques for doing this. The one of most interest here is to automatically invoke a separate program (i.e., a program other than the browser) for processing the data, which separate program is associated with the foreign-format data's type information. As a specific example of this technique, Raggett II describes a separate application "that take[s] a sequence of bytes and return[s] a pixmap." (*Id.*) Raggett II provides a general framework for calling such a separate program that can process the foreign-format data ("API for rendering foreign formats") and elaborates:

> Browsers can then be upgraded to display new formats without changing their code at all. All you would need is a way of binding the MIME content type to the function name for that format, e.g., via X resources or a config file. The functions could be implemented as separate programs driven via pipes and stdin/stdout or as dynamically linked library modules (Windows DLLs).

(*Id.*)

In the '831 reexamination, the Examiner agreed with the patent owner that Raggett II was lacking in an important respect. Specifically, the Examiner found that Raggett II was limited to invoking separate programs that translated the foreign-format data once, returned to the browser a static image, and then terminated. In other words, the Examiner determined that Raggett II did not disclose or suggest the separate program continuing to run to allow interactive processing of the foreign-format data.

What Eolas did not tell the Examiner, however, is that interactive Mosaic "helper" applications were common in the prior art. As described below, these undisclosed interactive programs, separate from the Mosaic browser, were designed to continue running and enable interactive processing and display of data. Indeed, the Janssen expansion of Raggett II describes

a technique for invoking such interactive programs. Thus, the patent owner's characterization of the separate programs available to the Mosaic browser, as contemplated in Raggett II, was at best incomplete and at worst misleading.

Without the benefit of a proper prior art landscape, on September 28, 2005, the PTO mailed a Notice of Intent to Issue Ex Parte Reexamination Certificate confirming the patentability of the '906 patent claims.

IV. SUMMARY OF THE PRIOR ART

The prior art describes what the '906 patent claims. For example, it describes what the '831 reexamination Examiner, provided with an incomplete and misleading description of the prior art, found lacking in Raggett II. Persons of skill in the art, as Mr. Janssen's posting demonstrates, knew of the availability of interactive Mosaic helper applications and therefore understood Raggett II quite differently than how the patent owner presented the prior art to the Patent Office Examiner. Below is a summary of previously unconsidered prior art demonstrating that the '906 patent claims what was already known in the art.

A. The Janssen Posting

On June 14, 1993, Bill Janssen and Dave Raggett held a public conversation over the Internet about techniques for doing what the '906 patent later claimed.

Within hours of its posting, Mr. Janssen publicly responded to Raggett II with a message he distributed to subscribers of the www-talk email distribution. For the same reasons that the Raggett II posting qualifies as prior art, the Janssen posting qualifies as prior art under 35 U.S.C. § 102(b).[1]

[1] Both postings were made to the WWW-Talk email list, which is a public, archived and indexed discussion forum whose participants included those who were developing and standardizing Internet technologies at the time (e.g., Marc Andreessen, who led development of the Mosaic browser). The postings were widely disseminated and publicly available through the Internet and through other means at least from June 14, 1993, and they continue to be available on-line at

The Janssen posting ("Janssen") is "In reply to" the message "Dave_Raggett: 'HTML+ support for eqn & Postscript.'," which is the Raggett II posting. (Ex. B.) Janssen quotes Raggett II:

> *> Browsers can then be upgraded to display new formats without changing their*
> *> code at all. All you would need is a way of binding the MIME content type*
> *> to the function name for that format, e.g. via X resources or a config file.*
> *> The functions could be implemented as separate programs driven via pipes and*
> *> stdin/stdout or as dynamically linked library modules (Windows DLLs).*

(*Id.*)

Janssen first affirms the quoted content of Raggett II – "Yes, that sounds good."

Then Janssen expands upon Raggett II:

> 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.

(*Id.*)

"My favorite way to handle this": With this language, Janssen acknowledges the functionality described in the Raggett II posting and introduces an alternative mechanism for implementing such functionality.

---

http://ksi.cpsc.ucalgary.ca/archives/WWW-TALK/www-talk-1993q2.index.html and elsewhere. As such, they constitute "printed publications" within the meaning of 35 U.S.C. § 102(b) because each was a "contribution" to "electronic bulletin boards, message systems, and discussion lists" that were "accessible to persons concerned with the art to which the document relates" when they were posted to the WWW-Talk list. (*See* M.P.E.P § 2128, which provides, in the section entitled "ELECTRONIC PUBLICATIONS AS PRIOR ART: Status as a 'Printed Publication,'" that "An electronic publication, including an on-line database or Internet publication, is considered to be a 'printed publication' within the meaning of 35 U.S.C. § 102(a) and (b) provided the publication was accessible to persons concerned with the art to which the document relates.") The Raggett II and Janssen postings enjoy prior art effect from the date of their posting (i.e., June 14, 1993). (*See* M.P.E.P. § 2128, which provides, in the section entitled "ELECTRONIC PUBLICATIONS AS PRIOR ART: Date of Availability," that "Prior art disclosures on the Internet or on an on-line database are considered to be publicly available as of the date the item was publicly posted.")

"is to have the browser create and manage an X sub-window over the area where the inset is to be displayed": With this language, Janssen describes the browser creating a new window ("X sub-window") for display of the foreign-format ("inset format") data. With this language, Janssen also describes the browser placing the sub-window inline with the rest of the data being displayed by the browser ("over the area where the inset is to be displayed"); i.e., the sub-window is placed at a location "inset" in the browser-controlled window.

"and pass the window ID of the sub-window to the subprogram which understands the inset format": With this language, Janssen describes how the browser hands over information to a separate program ("the sub-program") that is responsible for processing data in the foreign format. Specifically, the browser provides this separate program with a resource identifier ("window ID") that the sub-program uses to identify the sub-window for two purposes, which are described next.

"with the understanding that that program is to handle all events and refresh on the sub-window": With this language, Janssen reinforces that the separate program handles everything within the sub-window relating to interactive processing and display of the foreign-format data. Specifically, the separate program handles all keystrokes, all mouse button presses, all messages that the sub-window has become exposed, and all other events on the sub-window ("handles all events"). In other words, the separate program is not one that simply performs a translation and then terminates and disappears. On the contrary, Janssen describes this sub-program as one that takes over from the browser all event-handling throughout the life of the sub-window. Aside from event handling, the sub-program uses the window ID to identify the sub-window when drawing or redrawing ("handle all ... refresh") to the display.

"but the browser gets to handle configuration and window movement": With this language, Janssen further reinforces that the browser and separate program are operating side by side, with the browser handling two functions in relation to the sub-window. The browser handles the movement and the configuration of the sub-window within the browser-controlled

window, leaving foreign-format data processing and refresh operations on the sub-window to the separate program. Thus, when the browser changes the position, size, or border of the sub-window ("configuration and window movement"), the sub-window retains its place in the Web page display.

The above reading of the plain language of Janssen is confirmed by the state of the art in mid 1993. In particular, it is confirmed by the availability of interactive helper applications for Mosaic in the X Window System environment (e.g., Xv and Ghostview) as described below. In other words, the X Window System documentation and interactive helper application documentation submitted with this Request, and not previously considered by the Examiner, demonstrate conventional ways of doing what Janssen describes. As this patent owner has said, the hypothetical person of ordinary skill in the art "does things in a conventional way." (Response received October 12, 2004, page 25.)

B. Interactive Mosaic Helper Applications / External Viewers

In September 1993, the Mosaic browser running in the X Window System environment was designed to deal with certain foreign-format data by launching a pre-designated helper application associated with the particular foreign format encountered by the browser. For example, for data in the 'ps' format, the browser's default was to launch the Ghostscript application. This functionality is described in detail below.

Many of these helper applications were interactive. Once launched, they stayed running and handled the user's interactions with the particular data type in which they specialized. This functionality is detailed below for three different prior art interactive helper applications.

Mosaic's Launching of Helper Application Based On Particular Data Type Encountered

As noted, prior art versions of the Mosaic browser for the X Window System automatically associated particular helper applications with foreign-format data types by default.

Prior art Exhibits D – H documented this functionality for two prior art versions of Mosaic – version 1.2 and version 2.0 prerelease 4, as described below.

Mosaic 1.2: Source code and documentation for Mosaic 1.2 for the X Window System are prior art because they were available for download by anonymous file transfer protocol ("FTP") download at least as early as June 1993. (Ex. D.) Exhibit E shows the contents of a compressed archive file entitled, "xmosaic-1.2.tar.z." The compressed archive file includes a file entitled, "xresources.h," which was used to provide certain settings for XMosaic 1.2. (Ex. E.) Among other things, the xresources.h file associated external viewers with the following types. (Ex. E, page 3 of the file xresources.h.)

Data Type	External Viewer
gif	Xv
jpeg	Xv
tiff	Xv
dvi	Xdvi
mpeg	mpeg_play

Table 2: Viewers Launched for Data Types in Mosaic 1.2

Mosaic 2.0, prerelease 4: Source code and documentation for Mosaic 2.0 prerelease 4 were available for download by anonymous FTP download in September 1993. (Ex. F, page 1.) In Mosaic 2.0 prerelease 4, the mechanism for associating data types with external viewers was changed to one in which MIME types map to viewers. (Ex. F, pages 1-2.) The mappings for file extensions, MIME types, and viewers were customizable. (*Id.*) Viewers were not just for images. Users could associate MIME types for other kinds of content (e.g., video animations, scientific data) with applications appropriate for the content. By default, however, certain file extensions were mapped to MIME types, as shown in the following table. (Ex. G.)

File Extension	MIME Type
.ps	application/postscript
.gif	image/gif
.tif or .tiff	image/x-tiff

.jpg or .jpeg	image/jpeg
.rgb	image/x-rgb
.mpg or .mpeg	video/mpeg
.dvi	application/x-dvi

Table 3: Default File Extension to MIME Type Mappings in Mosaic 2.0 Prerelease 4

Exhibit H shows default MIME types in Mosaic 2.0 prerelease 4 along with mappings to viewers, including the following mappings. (Ex. H.)

MIME Type	External Viewer
image/*	Xv
video/mpeg	Mpeg_play
application/postscript	Ghostview
application/x-dvi	Xdvi

Table 4: Default MIME Type to Viewer Mappings in Mosaic 2.0 Prerelease 4

In sum, Exhibits D – H establish that prior art Mosaic for X Window System browsers invoked whatever helper application was assigned to the particular foreign format the browser encountered while parsing an HTML document. As shown above, these helper applications included Xv and Ghostview, each of which was interactive as described below. (Xdvi was also interactive but is not addressed at this time.)

1. Xv: "Interactive Image Display for the X Window System."

Xv is a prior art application that was invoked by Mosaic for interactive manipulation of GIF, JPEG, TIFF, and X11 format images. Requester submits prior art documentation for two versions of this program. (Exs. I, J, and K.)

The software Xv version 2.2 is described in the Xv 2.2 manual entitled, "Xvdocs.ps.x." (Ex. I.) The Xv 2.2 manual is dated April 24, 1992, and included in the archive "Xv-2.21.tar." In addition to the Xv 2.2 manual, Exhibit I includes printouts showing the contents of "Xv-2.21.tar."

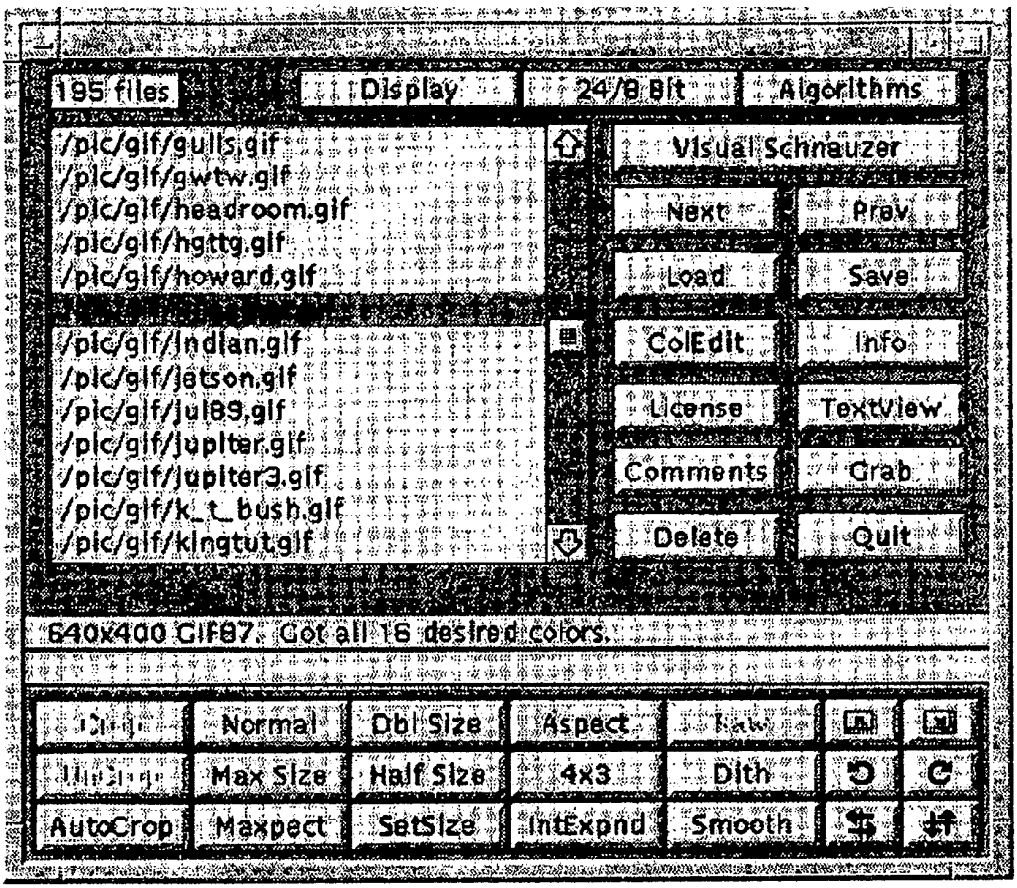
Xv version 3.0 is described in the Xv 3.0 manual entitled, "Xvdocs.ps.x." (Ex. J.) The Xv 3.0 manual is dated April 26, 1993, and included in the archive "Xv-3.00.tar." In

addition to the Xv 3.0 manual, Exhibit J includes printouts showing the contents of "Xv-3.00.tar." Exhibit K is a www-talk posting describing the availability of Xv 3.0 in April 1993.

Xv versions 2.2x and 3.0 were publicly distributed before October 17, 1993, and their distributions included the Xv 2.2 and 3.0 manuals. The Xv 2.2 manual and Xv 3.0 manual qualify as prior art against the '906 patent as printed publications under 35 U.S.C. § 102(b). They also provide evidence of how the Janssen and Raggett II postings would have been understood prior to October 17, 1993.

According to the title pages for the manuals, Xv 2.2 and 3.0 provided "Interactive Image Display for the X Window System." (Ex. I, title page of the Xv 2.2 manual; Ex. J, title page of the Xv 3.0 manual.) "Xv is an interactive image manipulation program for the X Window System." (Ex. I, page 2 of the Xv 2.2 manual; Ex. J, page 3 of the Xv 3.0 manual.) The manuals describe Xv operating on images in numerous formats, including GIF, JPEG, TIFF, and X11 bitmap formats. (*Id.*) For an image displayed in a window on the screen, Xv versions 2.2 and 3.0 allowed a user to stretch, rotate, flip, or crop the image, magnify a portion of the image, or adjust colors of the image. (*Id.*) In other words, Xv versions 2.2 and 3.0 allowed ongoing real-time manipulation and control of the image, resulting in changes to the displayed image.

The Xv 2.2 and 3.0 manuals describe a user interacting with the Xv software to change what was displayed using the Xv controls window. The Xv controls window "contains controls to resize the current image, flip and rotate it, load and save different files, and bring up the other Xv windows." (Ex. J, page 8 of the Xv 3.0 manual; *see also* Ex. I, page 6 of the Xv 2.2 manual.) The following figure shows an Xv controls window for Xv 3.0.



(*Id.*) Commands for resizing, rotating/flipping, smoothing, or cropping are "executed by either clicking the appropriate command button, or typing the keyboard equivalent (where given) into any open Xv window." (*Id.*) Thus, Xv 2.2 and 3.0 accepted keyboard input into the Xv controls window *and* the Xv window in which the image was displayed.

Xv 2.2 and 3.0 accepted various command line options. (Ex. I, pages 38-46 of the Xv 2.2 manual; Ex. J, pages 57-71 of the Xv 3.0 manual.) For Xv 3.0, the "-viewonly" command line option is described as "For use when calling Xv from some other program. Forces all user input to be ignored." (Ex. J, page 69.) The "-viewonly" option was not offered in Xv 2.2. In Xv 3.0, it was not the default setting. In any case, if Xv 3.0 were to handle all events in an Xv window, the calling program would not use the "-viewonly" option.

The "-" command line option is described as follows: "[s]pecifying '-' all by itself tells Xv to take its input from stdin, rather than from a file. This lets you put Xv on the end of a Unix pipe." (Ex. I, page 46; Ex. J, page 71.) Whether accepting input from a file or from stdin, however, Xv 2.2 and 3.0 by default allowed interactive manipulation of the image.

Thus, the Xv 2.2 and 3.0 manuals describe Xv accepting an image file as input and drawing/redrawing an image. Xv was interactive. The Xv 2.2 and 3.0 manuals describe ongoing interactive processing of the image data as keystrokes and button inputs were handled by the Xv software.

2. Ghostview: Previewer for Postscript Files

Ghostview is a prior art application that was invoked by Mosaic for interactive display and manipulation of postscript files. Requester submits prior art documentation for this prior art helper application. (Ex. L.)

The manual in the file entitled "Ghostview.ps" (Ex. L) describes Ghostview version 1.5. The Ghostview manual is included in the archive "Ghostview-1.5.tar.gz," which indicates Ghostview.ps was last modified on July 25, 1993. Ghostview 1.5 was publicly distributed before October 17, 1993, and its distributions included the Ghostview manual. (*See* Ex. L, page 1.) The Ghostview manual qualifies as prior art against the '906 patent as a printed publication under 35 U.S.C. § 102(b); it also provides evidence of how the Janssen and Raggett II postings would have been understood prior to October 17, 1993. Exhibit L includes the Ghostview manual as well as printouts showing a directory listing of an FTP site and the contents of "Ghostview-1.5.tar.gz." Exhibit L also includes a printout of the "README" file from "Ghostview-1.5.tar.gz." The README file is dated July 25, 1993, and indicates:

> Ghostview-1.5 is available via anonymous ftp from:
> prep.ai.mit.edu:/pub/gnu/ghostview-1.5.tar.gz
> ftp.cs.wisc.edu:/pub/X/ghostview-1.5.tar.gz

(Ex. L, page 19, which is page 1 of the Ghostview 1.5 README file.)

Ghostview 1.5 provided an X Window System interface for Ghostscript, an interpreter program for postscript files. (Ex. L, page 1 of the Ghostview 1.5 manual.) The Ghostview manual describes Ghostview 1.5 and Ghostscript functioning as two cooperating programs, with Ghostview creating a viewing window and Ghostscript drawing in it. (*Id.*)

The Ghostview manual describes Ghostview 1.5 as having a main viewport (for displaying an image) and a menu box with buttons for bringing up menus. (Ex. L, pages 1-2 of the Ghostview manual.) The menus included a "page" menu for changing pages, a "magstep" menu for changing view magnification, and an "orientation menu" for changing orientation. (*Id.*) Most of the popup menu commands could be invoked from the keyboard, and other keyboard input caused scrolling of the main viewport window up, down, left, or right. (Ex. L, pages 4-5 of the Ghostview manual.) Clicking a mouse button anywhere within the viewport window caused a zoom window to pop up. (Ex. L, page 1 of the Ghostview manual.)

The Ghostview manual describes Ghostview 1.5 accepting a postscript file as input and causing drawing/redrawing of an image (in cooperation with Ghostscript). It also describes, however, Ghostview 1.5 handling keystrokes and button inputs for ongoing interactive processing of the image.

Ghostview 1.5 was invoked with the name of the file to be previewed as a parameter. (*Id.*) Alternatively, Ghostview read from stdin if the filename was "-." (*Id.*) Either way, Ghostview 1.5 provided tools for interactive manipulation of the image.

Ghostview 1.5 *passed the window ID* of a window to Ghostscript, for Ghostscript to draw an image to the window. The window ID passing mechanism is detailed below in the section about communication between X Clients in the X Window System.

C. X Window System

The Janssen posting must be understood in context. It relates to software for the X Window System. For this reason, Requester also submits sections of two prior art books about

the X Window System. (Exs. M and N.) Exhibit M includes chapters from Douglas Young, The X Window System, Programming and Applications with Xt (1990). Exhibit N includes chapters from Adrian Nye, Xlib Programming Manual for Version 11 (1990). Each of these books qualifies as prior art against the '906 patent as a printed publication under 35 U.S.C. § 102(b).

1. Overall Architecture

The Janssen posting uses the terms "X sub-window" and "window ID of the sub-window." These terms of art must be understood in the context of the X Window System.

X Server software runs on a machine with a display, a keyboard, and a mouse. (Ex. M, 1.1 and 1.3; Ex. N, 1.2.2.) Programs called X Clients interact with the X Server. (*Id.*) Mosaic and Mosaic helper applications for the X Window System are X Clients.

The X Server stores data on behalf of X Clients and often shares the data between the X Clients. (*Id.*) The X Server provides identifying information for resources (e.g., window IDs for windows), and multiple X Clients can access the stored data with the identifying information. (*Id.*)

"Windows" in the X Window System are hierarchically organized according to parent-child relationships. (Ex. M, 1.5; Ex. N, 2.1.3, 2.2, and 2.4.) The root window of the hierarchy covers a whole display and has one or more child windows. (*Id.*) A child window is also called a sub-window. (*Id.*) Each window is identified by a window ID. (*Id.*)

As for communication, an X Client sends "requests" to the X Server (e.g., to request information from the X Server or request that the X Server draw something). (Ex. M, 1.4; Ex. N, 1.3.) The X Server may send "replies" in response. The X Server also sends "events" to X Clients. (Ex. M, 1.6; Ex. N, 1.3.)

2. Events

The Janssen posting uses the language "handles all events and refresh on the sub-window." This language must be understood in the context of "events" in the X Window System.

The X Server sends "events" to an X Client when the X Server receives keystroke or mouse input directed to a window of the X Client. (Ex. M, 5.1 and 5.3; Ex. N, 2.5.1.) Other events relate to changes to the display or messages used for inter-client communication. (*Id.*) A third category of events are change notifications received by an X Client, for example, when the position, size, or border of a window of the X Client has changed. (Ex. M, 5.3.7.)

Typically, the X Server directs user input events to the window having user input focus. (Ex. M, 5.1; Ex. N, 2.5.2.) An X Client can change which windows receive which events through various mechanisms, and an X Client can change the windows for which it receives events. (Ex. M, 5.2; Ex. N, 2.5.2.) For example, for a particular window, an X Client can select which types of events are to be delivered to the X Client and which are to be ignored. (*Id.*)

For most event types, a particular event is sent to more than one X Client if each of the X Clients has selected the appropriate event type on the window in question. (*Id.*) Each of the X Clients has its own "event mask" for such a window. (*Id.*) If it has the window ID of the window, an X Client selects the event types for events it should receive and calls the function XSelectInput(), passing the window ID and the event mask. (*Id.*) The X Server routes events to interested X Client(s) according to the event mask(s). (*Id.*)

3. Communication Between X Clients

The Janssen posting describes interactions between the browser and subprogram ("pass the window ID" and "the understanding that the program is to handle ... but the browser gets to handle ..."). This language must be understood in the context of communication in the X Window System.

In the X Window System, mechanisms for inter-client communication include environment variables and properties, copy and paste using selections, buffering of data with X Server, command line options, and events. (Ex. M, 11.)

An X Client can use an environment variable to pass data to another X Client. For example, Exhibit O shows a file entitled "gs.interface" from the archive file "Ghostview-

1.5.tar.gz" for Ghostview 1.5 (*see* Ex. L). The "gs.interface file" describes Ghostview using an environment variable to *pass the window ID* of a window to Ghostscript. (Ex. O.)

> When the GHOSTVIEW environment variable is set, ghostscript draws on an existing drawable rather than creating its own window. Ghostscript can be directed to draw on either a window or a pixmap.
>
> Drawing on a Window
>
> ***The GHOSTVIEW environment variable contains the window id of the target window. The window id is an integer.*** Ghostscript will use the attributes of the window to obtain the width, height, colormap, screen, and visual of the window. The remainder of the information is gotten from the GHOSTVIEW property on that window.
>
> Drawing on a Pixmap
>
> The GHOSTVIEW environment variable contains a window id and a pixmap id. They are integers separated by white space. Ghostscript will use the attributes of the window to obtain the colormap, screen, and visual to use. The width and height will be obtained from the pixmap. The remainder of the information, is gotten from the GHOSTVIEW property on the window. In this case, the property is deleted when read.
>
> The GHOSTVIEW environment variable
>
> parameters: window-id [pixmap-id]
>
> scanf format: "%d %d"
>
> explanation of parameters:
>
> window-id: tells ghostscript where to
> - read the GHOSTVIEW property
> - send events
> If pixmap-id is not present,
> ***ghostscript will draw on this window.***
>
> pixmap-id: If present, tells ghostscript that a pixmap will be used as the final destination for drawing. The window will not be touched for drawing purposes.

(Ex. O, page 1, emphasis added.) In this way, by passing a window ID, Ghostview 1.5 told Ghostscript in which window to draw the image from a postscript file.

The file "gs.interface" describes additional communications between Ghostscript and Ghostview – events sent between Ghostscript and Ghostview.

> Ghostscript sends events to the window where it read the GHOSTVIEW property. These events are of type ClientMessage. The message_type is set to either PAGE or DONE. The first long data value gives the window to be used to send replies to ghostscript. The second long data value gives the primary drawable. If rendering to a pixmap, it is the primary drawable. If rendering to a window, the backing pixmap is the primary drawable. If no backing pixmap is employed, then the window is the primary drawable. This field is necessary to distinguish multiple ghostscripts rendering to separate pixmaps where the GHOSTVIEW property was placed on the same window.
>
> The PAGE message indicates that a "page" has completed. Ghostscript will wait until it receives a ClientMessage whose message_type is NEXT before continuing.
>
> The DONE message indicates that ghostscript has finished processing.

(Ex. O, page 2.) This is one example of communication between X Clients using ClientMessage events. More generally, an X Client can send ClientMessage events to another X Client using calls to XSendEvent() and can handle ClientMessage events received from the other X Client. (Ex. M, 11.3.)

For another inter-client communication mechanism, the X Server stores selections of data for copy and paste operations between X Clients. (Ex. M, 11.4.)

D. The Patent Applicants' Visible Embryo Project Software

Some of the dependent claims concern a distributed application architecture that two of the patent applicants (Doyle and Ang) described in a prior art article. The article, entitled "Processing Cross-sectional Image Data for Reconstruction of Human Developmental Anatomy from Museum Specimens," was published in a "SIGBIO" newsletter in early 1993. (Ex. P.) The

Doyle-Ang SIGBIO article qualifies as prior art against the '906 patent as a printed publication under 35 U.S.C. § 102(b).

The Doyle-Ang SIGBIO article describes embryo visualization software developed as part of the "Visible Embryo Project" referenced in the '906 patent. (*See* Ex. P, pages 5-6.) The software operated on the X Window System and was distributed between two computers. This distributed architecture is of particular relevance to claims 4, 5, 9, and 10 of the '906 patent, according to which "additional instructions for controlling said controllable application reside on said network server."

Not coincidentally, the "application client 210" and "application server 220" in the '906 patent are software for the Visible Embryo Project. ('906 patent, column 10, lines 16-46.) "Versions and descriptions of software embodying the present invention are generally available as hyperlinked data objects from the Visible Embryo Project's World Wide Web document at the URL address 'HTTP://visembryo.ucsf.edu/'."

The Doyle-Ang SIGBIO article indicates that the Visible Embryo Project software had "already been successfully tested on workstations from Silicon Graphics, Sun, and IBM." (Ex. P, page 6.) It also reports that the embryo visualization software had been publicly demonstrated at SIGGRAPH '92, a conference in Chicago in 1992. (Ex. P, pages 5-6.) Nevertheless, in the '906 patent the patent applicants characterized the embryo visualization software as being "presently under development" ('906 patent, column 10, lines 18-19), and they failed to cite the Doyle-Ang SIGBIO article during initial prosecution or during the '831 reexamination. Even if certain aspects of the embryo visualization software were still "under development" in October 1994 as claimed, that does not change the prior art status or materiality of the Doyle-Ang SIGBIO article or the software that was "successfully tested" and demonstrated in 1992.

V. EXPLANATION OF THE PERTINENCY AND MANNER OF APPLYING CITED PRIOR ART

Reexamination of claims 1-10 of the '906 patent is requested for the following reasons:

- Claims 1 and 6 of the '906 patent are anticipated by Janssen (as Janssen would have been understood in the context of the general knowledge in the art evidenced by Raggett II, Mosaic, and X Window System art).
- Claims 1 and 6 of the '906 patent are obvious over APA Mosaic in view of Raggett II and Janssen.
- Claims 2, 3, 7, and 8 of the '906 patent are obvious over Janssen (or alternatively APA Mosaic in view of Raggett II and Janssen) in view of Young (Ex. M) or Ghostview 1.5 (Ex. L and O).
- Claims 4, 5, 9, and 10 of the '906 patent are obvious over Janssen (or alternatively APA Mosaic in view of Raggett II and Janssen) in view of Young (or Ghostview 1.5) and the Doyle-Ang SIGBIO article (Ex. P).

The patent owner did not present to the Examiner or ask the Examiner to consider any of the Janssen, Young, Ghostview 1.5 and Doyle-Ang SIGBIO prior art, either during initial prosecution of the '906 patent or during the '831 re-examination. These references are without doubt highly relevant as they disclose the very claim requirements the Examiner found lacking in the limited prior art previously considered by the Office in the '831 re-examination.

A. The Janssen Posting Anticipates Claims 1 and 6

Claims 1 and 6 of the '906 patent recite a prior art method and product whereby a browser handles a foreign-format data object identified in a special text code and having associated type information by automatically invoking an external application to provide immediate interaction and display of that object. ('906 patent, Title.) Specifically, claims 1 and 6 each recite the limitation, "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." In the Examiner's Statement of Reasons for Patentability and/or Confirmation, dated September 27, 2005, the Office interpreted this claim language as follows:

> Significantly, the instant claimed 'interactive processing' of the '906 patent begins at the moment the browser application parses an 'embed text format' detected within the hypermedia document. The web browser invokes the claimed 'executable application' immediately after an 'embed' tag is parsed and before the hypermedia document is completely displayed in the browser-controlled window. The invoked 'executable application' enables the claimed 'interactive processing.'
>
> Instant '906 independent claims 1 and 6 therefore require an operative coupling between the claimed 'executable application' and the claimed 'interactive processing' such that the claimed 'interactive processing' must be enabled by an 'executable application' that meets five explicitly claimed requirements:
>
> 1. The executable application must be external to the first distributed multimedia document.
> 2. The executable application must be automatically invoked by the browser application when the 'embed text format' is parsed by the browser application.
> 3. The executable application must execute on the client workstation.
> 4. The executable application must display the object within the display area created at the first location within the portion of the first distributed hypermedia document being displayed in the first browser-controlled window.
> 5. The executable application must enable interactive processing of the object within the display area created at the first location within the portion of the first distributed hypermedia document being displayed in the first browser-controlled window.
>
> ...
>
> As discussed supra, a proper construction of the claimed 'interactive processing' necessarily requires some capability of ongoing real-time manipulation and control by the user that is applied to the object displayed within the first browser-controlled window. It is axiomatic that an executable application that terminates is incapable of providing the type of 'interactive processing' required by instant '906 independent claims 1 and 6.

(Pages 9-10, emphases in original.)

Janssen's expansion of Raggett II meets all limitations of claims 1 and 6, and it satisfies all requirements set forth in the above-quoted Examiner's Statement of Reasons for Patentability and/or Confirmation. More specifically, a person of skill in the art reading Janssen in its proper X Window System context (i.e., Raggett II, Mosaic, available interactive helper applications, and the X Window System), would have discerned each and every claim limitation of claims 1 and 6, as demonstrated in the claim chart below.

Janssen in particular discloses each of the five claim requirements identified by the Examiner in the above-quoted action:

1. The executable application must be external to the first distributed multimedia document: Janssen's executable "sub-program" is external to the HTML document. (Ex. B.) Raggett II (on which Janssen expands) describes the separate program as being external to the HTML document. (Ex. C.) In Raggett II, the browser identifies the format of the foreign-format data from a MIME type attribute in an EMBED tag parsed from the HTML document. (*Id.*) The MIME content type has been bound to a separate program, "e.g., via X resources or a config file." This description refers to a well-known MIME type mechanism by which a program relates a MIME type for a data format to an available application on the computer capable of processing that format of data. (*See* Ex. F-H.) Indeed, this is the exact same MIME type binding mechanism used with the EMBED tag described in the '906 patent at column 15, lines 9-16.

2. The executable application must be automatically invoked by the browser application when the 'embed text format' is parsed by the browser application: Janssen also incorporated this automatic invocation feature from Raggett II. Specifically, the separate

program is automatically invoked by the browser when it parses the EMBED tag. (Ex. C.) In Raggett II, the EMBED tag includes, for example, a URL that references foreign-format data in a separate file. (*Id.*) Upon parsing the EMBED tag, the Janssen browser interprets the MIME type specified in the EMBED tag, creates a sub-window where the foreign-format data will be displayed, and invokes the sub-program appropriate to the format, resulting in inline display of the foreign-format data in the sub-window of the browser window. (Ex. B and C.)

3. The executable application must execute on the client workstation: The Janssen sub-program executed, at least in part, on the client workstation. This was standard operating procedure at the time; the interactive Mosaic helper applications ran, predominately, on the client workstation on which the browser ran. (*See* Exhibits I, J, and L.)

4. The executable application must display the object within the display area created at the first location within the portion of the first distributed hypermedia document being displayed in the first browser-controlled window: Janssen describes this as well -- the foreign-format data is displayed within the "inset" area within the HTML document displayed in the Janssen browser's window. (Ex. B.) The Janssen browser creates the X sub-window over the "area where the inset is to be displayed." This sub-window is "refreshed" by the separate program to render the foreign-format data. (*Id.*)

5. The executable application must enable interactive processing of the object within the display area created at the first location within the portion of the first distributed hypermedia document being displayed in the first browser-controlled window: As explained above, adding this interactivity was one way in which Janssen expanded upon Raggett II. Janssen expressly disclosed that the sub-program continued to run and handled "all

events" in the sub-window. The plain meaning of "all events" is *every* event, including user input events, not just "some" events.

More specifically, the Janssen browser enables interactive processing of the embedded data object within the sub-window within the browser's HTML document display window. Janssen states, "the browser [passes] 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." As previously discussed, this statement would be understood in the context of the X Window System to refer to well known mechanisms of a first program passing a window ID to a second program, allowing the second program to select events (including mouse clicks, keystrokes, etc.) to receive for the window and handle such events (e.g., by setting an event mask and registering an event handler) and refresh or update the display of that window accordingly. (*See* Exs. M, N, and O.) The Mosaic helper applications Xv and Ghostview responded to user input in their window(s) (including the main display window) by changing how content was viewed or changing the characteristics of the content. (*See* Exs. I, J, and L.) The Janssen browser thus enables, through the separate program, ongoing real-time manipulation and control by the user of the embedded content in the sub-window. Because the separate program remains active in order to handle "all events and refresh" on the sub-window as described by Janssen, it does not terminate.

As summarized and shown in the following claim chart, the Janssen posting includes all recited claim elements of claims 1 and 6. Moreover, as just discussed, the Janssen posting meets all requirements of the Examiner's interpretation of the above-discussed "interactive processing" clause of claims 1 and 6. Accordingly, reexamination of claims 1 and 6 based on Janssen is appropriate.

Claim 1 of the '906 Patent	Prior Art
A method for running an application program in a computer network environment, comprising:	Janssen would have been understood in the context of the general knowledge in the art evidenced by the Raggett II, APA Mosaic, and X-Window System art. Janssen describes a "subprogram which understands the inset format." (Ex. B.) The "subprogram" runs in a computer network environment. A Mosaic helper application is one such "subprogram" and is used as a viewer for the "inset format." (APA Mosaic. *See, e.g.*, '906 patent, column 3, lines 13-16.)
providing at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment;	Janssen describes a browser. (Ex. B.) The browser executes on a client workstation and interacts with a network server coupled to the network environment, which is a distributed hypermedia environment. (APA Mosaic. *See, e.g.*, '906 patent, column 5, lines 34-36, and Response received May 12, 2004, page 3 ("When the browser is launched on a client workstation...").)
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	Janssen describes a browser. (Ex. B.) The browser: • executes on a client workstation; • parses a hypermedia document to identify text formats included in said distributed hypermedia document; • responds to predetermined text formats to initiate processing specified by said text formats; and • causes display, on the client workstation, of at least a portion of the hypermedia document received over the network from the network server, where the portion of the hypermedia document is displayed within a browser-controlled window on the client workstation. (APA Mosaic. *See, e.g.*, '906 patent, column 5, lines 28-38, and Response received May 12, 2004, page 3 ("The browser then retrieves a selected HTML published source document from a network server utilizing a uniform resource locator (URL) that locates the HTML document on the network..." and "The browser application then parses the local copy of the HTML document, renders the temporary local copy of the HTML document in to a Web page, and displays the rendered Web page in a browser-controlled window.").)

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	Janssen is a reply to Raggett II and describes processing an EMBED tag included in a hypermedia document. (Ex. B.) The EMBED tag can have a URL that references "foreign data in a separate file." (Raggett II, Ex. C. "Well both of these will be possible with the HTML+ DTD, by using the capability to embed foreign formats inline in the HTML+ source, e.g. <H2>A example of an equation</H2> <EMBED TYPE="text/eqn">zeta (s) ~=~ sum from k=1 to inf k sup –s~~~ (Re s > 1) </EMBED>" "p.s. you can also put the foreign data in a separate file referenced by a URL.")
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	Janssen describes a "MIME content type." (Ex. B.) The "MIME content type" is indicated in the EMBED tag, and the browser utilizes the MIME content type to identify and locate an executable application external to the first distributed hypermedia document. (Raggett II, Ex. C. "The browser identifies the format of the embedded data from the 'type' attribute, specified as a MIME content type.")
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.	Janssen describes a browser that automatically invokes a "subprogram which understands the inset format" to execute on the client workstation, and the "subprogram" causes display of the foreign data in the "inset format." (Ex. B. "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." *See also* Raggett II, Ex. C. "Browsers can then be upgraded to display new formats without changing their code at all. All you would need is a way of binding the MIME content type to the function name for that format, e.g., via X resources or a config file.") In this way, Janssen describes the browser enabling interactive processing of the foreign data within a display area created at the location within the portion of the hypermedia document being displayed in the browser-controlled window. (*Id.*)

B. APA Mosaic, Raggett II, and Janssen Render Obvious Claims 1 and 6

Janssen in combination with Raggett II and APA Mosaic renders the subject matter of claims 1 and 6 of the '906 patent obvious. Raggett II provides the suggestion to modify existing Mosaic browser technology inasmuch as such modifications are described in Raggett II. Janssen similarly expands upon the Raggett II/Mosaic combination.

More particularly, APA Mosaic teaches a hypermedia browser that meets all limitations of claims 1 and 6, except the limitations, "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" 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."

Raggett II describes modifying the Mosaic browser to provide the capability to "embed foreign formats inline" in the hypermedia document, using an EMBED tag that has a "type attribute" specifying a "MIME content type" of the foreign format. (Ex. C.) Raggett II further describes binding the content type specified in this embed tag to functions implemented as separate programs for displaying the foreign format. Raggett II states, "Browsers can then be upgraded to display new formats without changing their code at all." (*Id.*) With this statement, Raggett II illustrates one motivation to modify the Mosaic browser to support the EMBED tag and bind content types to appropriate separate programs. (*Id.*) Raggett II also describes the

EMBED tag specifying the location of foreign-format data external to the HTML document. (Ex. C. "[Y]ou can also put the foreign data in a separate file referenced by a URL.")

The Janssen posting describes further modifying the Raggett II/Mosaic combination to have the browser create an "X sub-window" where "the inset is to be displayed." (Ex. B.) The browser then passes the window ID of this sub-window to the "subprogram" that handles the foreign format, "with the understanding that that program is to *handle all events* and refresh on the sub-window." (*Id.*) Well-known mechanisms existed in the X Window System to pass a sub-window ID to a separate program and enable interactive processing of the content in the sub-window by the separate program, which handled all events, such as a user's keyboard and mouse inputs, directed to the sub-window. (*See, e.g.*, Ex. M, 5.2; Ex. N, 2.5.2.) In fact, the Ghostview 1.5 software discussed above utilized such mechanisms. (Ex. O.)

Janssen does not directly state why this mechanism is his "favorite way to handle this." (Ex. B.) However, Janssen does state that passing the window ID allows the separate program "to handle all events and refresh on the sub-window." This would be readily understood by those familiar with X Window System programming (as evidenced by, e.g., Exhibits M, N, and O) as a mechanism that enables interactive processing of the foreign format content in the sub-window by the separate program responding to keyboard/mouse input events. Accordingly, apart from the fact that Janssen directly and explicitly modifies the Raggett II/Mosaic browser combination, Janssen provides a motivation for this further modification of the Raggett II/Mosaic browser – using proven techniques to enable interactive processing by the separate program.

As already fully discussed in the previous section, the combination of Janssen, Raggett II and APA Mosaic meets all requirements of the claim language as interpreted by the

Office. Accordingly, reexamination of claims 1 and 6 based on the combination of Janssen, Raggett II and APA Mosaic is appropriate.

C. Janssen and Young/Ghostview Render Obvious Claims 2, 3, 7 and 8

Claims 2, 3, 7 and 8 of the '906 patent are dependent claims, which add further limitations relating to the browser controlling the separate program via inter-process communication. Claim 2 recites the limitation of "interactively controlling said controllable application on said client workstation via inter-process communications between said browser and said controllable application." Claim 3 recites the further limitation, "wherein the communications to interactively control said controllable application continue to be exchanged between said controllable application and the browser even after the controllable application program has been launched." Claims 7 and 8 recite nearly identical limitations.

In the X Window System, there are various mechanisms available for inter-process communications between client programs, including environment variables ("atoms and properties"), client message events, and the X selection mechanism. (Ex. M, 11.1 to 11.4, page 280; Ex. O.) With client message events, a client program can send an event message to another client program by calling the XSendEvent() function of the X Server. (Ex. M, 11.3, page 294, "This feature can be used to forward events from one application to another, or to create and send new events"; Ex. O.)

Young describes a program called "xtalk" that allows users on two different machines to communicate with each other by text. (Ex. M, 11.3, page 296.) The xtalk program on one user's machine sends/receives client message events to/from the xtalk program on the other user's machine, for example, to make connection requests, including "disconnect" and

"accept" notifications, and to send/receive keyboard events that affect display on the other user's machine. (Ex. M, 11.3, page 297.)

Ghostview 1.5 provides an example of inter-process communication in the context of a viewer application (namely, Ghostview) and separate rendering program (namely, Ghostscript). (Ex. L, page 5.) The Ghostview and Ghostscript programs communicate using an environment variable to pass a window ID, as discussed above. (Ex. O.) Ghostview and Ghostscript also communicate with each other on a continuing basis by sending client message events, such as "PAGE" (Ghostscript indicating to Ghostview that it completed drawing a page), "NEXT" (Ghostview indicating to Ghostscript to draw a next page) and "DONE" (Ghostscript indicating to Ghostscript that it has drawn up to end of the file). (Ex. O, page 1.)

Xtalk and Ghostview/Ghostscript show programs using inter-process communication to control display in a window or sub-window by another program. Xtalk illustrates using client message events for one program to forward keyboard events that affect display in a separate program's window. Ghostview sends client message events to control paged display by Ghostscript in a window whose ID was passed by Ghostview.

Janssen teaches a browser system that enables interactive processing of inset foreign-format data using a separate program. Upon parsing an EMBED tag, the Janssen browser creates an X sub-window over an area of the HTML document where the foreign-format data is to be displayed. The Janssen browser passes a window ID for this sub-window to the separate program, which then handles "all events and refresh on the sub-window" while the browser handles "configuration and window movement." (Ex. B.) If the browser causes a change to the position, size, or border of the sub-window, this results in a change notification being sent to the separate program, which the separate program handles. (Ex. M, 5.3.7.) Aside

from this type of event, Young's teaching (or alternatively that of Ghostview 1.5) to use client message events for inter-process communication to control display in a sub-window by another program would motivate like use of inter-process communication by the Janssen browser to control display or exchange status information. For example, the Janssen browser would directly communicate to the subprogram window configuration and movement changes (such as window re-sizing, scrolling or paging of the HTML document display) that affect the separate program's inset data display.

Note the similarities to inter-process messages sent between the browser and external program described in the '906 patent. In the table shown at column 12, lines 15-17 of the '906 patent, the set of predefined messages includes the messages: "server update done," "server ready," "server exiting," "area shown," "area hidden," and "area destroyed." Both the MEAPI inter-process communications and Ghostview's message events control paged display to effect the "understanding" expressed in the Janssen posting, i.e., that "the browser gets to handle configuration and window movement" while the subprogram "is to handle all events and refresh on the sub-window." (Ex. B.)

The Janssen browser as modified by the teaching of Young or Ghostview 1.5 meets the added claim limitations of claims 2, 3, 7 and 8. These claims therefore would have been obvious over Janssen in view of either reference.

Claim 2 of the '906 Patent	Prior Art
The method of claim 1, wherein said executable application is a controllable application and further comprising the step of: interactively controlling said controllable application via inter-process communications between said browser and said controllable	Young describes various inter-process communication mechanisms in the X Window System, including environment variables, client message events and X selection. (Ex. M, 11, pages 280-332.) More particularly, Young describes a program sending client message events to control another application, such as one user's xtalk program sending "connection requests," "disconnect" and "accept" notifications, as

application.	well as forwarding keyboard input events, to a subwindow of another user's xtalk program. (Ex. M, 11.3, page 297.) The Ghostview 1.5 documentation describes Ghostview interactively controlling Ghostscript, including by passing a window ID via an environment variable and sending client message events to control paged display. (Ex. S.)

Claim 3 of the '906 Patent	Prior Art
The method of claim 2, wherein the communications to interactively control said controllable application continue to be exchanged between said controllable application and the browser even after the controllable application has been launched.	Young describes one program controlling another program (e.g., two xtalk programs) via inter-process communication mechanisms, including environment variables, client message events and X selection, which occurs after launch of the other program. (Ex. M, 11.1-11.4.) The Ghostview 1.5 documentation describes Ghostview interactively controlling Ghostscript, including by passing a window ID via an environment variable and sending client message events to control page display, which occur after Ghostview launches Ghostscript. (Ex. O.)

D. Janssen, Young/Ghostview, and Doyle-Ang Render Obvious Claims 4, 5, 9 and 10

Claims 4, 5, 9 and 10 of the '906 patent are dependent claims, which add further limitations relating to the separate program executing on both the client workstation and network server as a distributed application. Claim 4 recites limitations of "issuing... commands to the network server," "executing, on the network server," "sending information from said network server... and processing said information at the client workstation to interactively control said controllable application." Claim 5 recites the further limitation, "wherein the communications to interactively control said controllable application continue to be exchanged between said

controllable application and the browser even after the controllable application program has been launched." Claims 9 and 10 recite nearly identical limitations.

The Doyle-Ang SIGBIO article describes software developed as part of the "Visible Embryo Project." (Ex. P.) The article reads, in part:

> Software tools were developed to allow the interactive three-dimensional visualization of the embryo reconstruction in real time. Figure 3 shows the display of the application as it appeared at the SIGGRAPH '92 conference in Chicago (Doyle, et al., 1992). The left of the screen shows a surface-based model of the embryo's exterior. This model was built from data which was derived, through three-dimensional interpolation, from the original embryo dataset. Two-hundred volume slices of the embryo (stored as texture maps) can be interactively displayed at this lower resolution while the model is rotated freely in three dimensions. A cutting-plane can be seen to intersect the surface-based model. This cutting plane can be interactively controlled to intersect with the embryo model at any arbitrary angle and position. To the right of the screen, one can see a window that displays a high-resolution image of the oblique section through the embryo as indicated by the interactive cutting plane. *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.
>
> Current efforts are being directed towards the development of a very portable tool for viewing arbitrary oblique slices through such data. This program allows interactive display of orthogonal and oblique slices through volumetric data without using any machine-specific functions. The application is written in pure ANSI-standard C and *uses the X Window (Motif) toolkit for its interface. It has already been successfully tested on workstations from Silicon Graphics, Sun, and IBM.*

(Ex. P, pages 5-6.)

According to the article, the embryo visualization software is distributed between two computers separated by a network – the "interface panel" and "3-D surface model" run on a workstation while an "oblique section image" is computed on a supercomputer (network server).

The operation of such a distributed software application necessarily involves issuing of commands to the network server and sending of response information to the workstation. Although the Doyle-Ang SIGBIO article presents little technical detail about communication across the network, certain features are inherent to communication between a workstation and network server in distributed software applications. These inherent features include: issuing of commands from the workstation to the network server, execution of instructions on the network server, sending of information from the network server back to the workstation, and processing of the information at the workstation.

In the article, the authors indicate that this split of the computational load between supercomputer and workstation was made to "maintain the quick response needed for effective real-time interaction." This teaching to split the computational load between a workstation and other computer so as to maintain real-time interactivity would have motivated distributing the load of computationally intensive subprograms for the Janssen (in combination with Young or Ghostview) browser when required for real-time interactivity.

The combination Janssen and Young (or Ghostview) as modified by the teaching of the Doyle-Ang SIGBIO article meets the added claim limitations of claims 4, 5, 9 and 10. These claims therefore would have been obvious.

Claim 4 of the '906 Patent	Prior Art
The method of claim 3, wherein additional instructions for controlling said controllable application reside on said network server, wherein said step of interactively controlling said controllable application includes the following sub-steps:	The Doyle-Ang SIGBIO article describes instructions controlling the three-dimensional embryo visualization application residing on the supercomputer (network server). (Ex. P, page 6: "Computation of the high-resolution oblique section image displayed in the right window took place on the Convex supercomputer.")
issuing, from the client workstation,	The Doyle-Ang SIGBIO article describes the interface

one or more commands to the network server;	panel running on the workstation and communicating with the network server. (Ex. P, page 6: "Both of these operations [interface panel and sectional computation] occurred simultaneously, communicating through a high-speed fiber optic network.")
executing, on the network server, one or more instructions in response to said commands;	The Doyle-Ang SIGBIO article describes the oblique sectional image computation being performed on the network server under interactive control of the interface panel on the workstation. (Ex. P, page 6: "This cutting plane can be interactively controlled to intersect with the embryo model at any arbitrary angle and position. To the right of the screen, one can see a window that displays a high-resolution image of the oblique section through the embryo as indicated by the interactive cutting plane. ... Computation of the high-resolution oblique section image displayed in the right window took place on the Convex supercomputer.")
sending information from said network server to said client workstation in response to said executed instructions; and	The Doyle-Ang SIGBIO article describes the oblique sectional image computed at the network server being communicated to the workstation for display. (Ex. P, page 7: "This program allows interactive display of orthogonal and oblique slices through volumetric data without using any machine-specific functions"; Figure 3)
processing said information at the client workstation to interactively control said controllable application.	The Doyle-Ang SIGBIO article describes the interface panel running on the workstation interactively controlling the display of the oblique sectional image data. (Ex. P, page 6: "This cutting plane can be interactively controlled to intersect with the embryo model at any arbitrary angle and position. To the right of the screen, one can see a window that displays a high-resolution image of the oblique section through the embryo as indicated by the interactive cutting plane.")

Claim 5 of the '906 Patent	**Prior Art**
The method of claim 4, wherein said additional instructions for controlling said controllable application reside on said client workstation.	The Doyle-Ang SIGBIO article describes the interface panel and 3-D surface model program as residing on the workstation. (Ex. P, page 6: "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.")

VI. CONCLUSION

For the above reasons, reexamination of claims 1-10 of the '906 patent is hereby respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

By _____
Kyle B. Rinehart
Registration No. 47,027

By _____
Stephen A. Wight
Registration No. 37,759

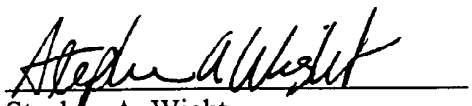
One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 595-5300
Facsimile: (503) 595-5301

CERTIFICATE OF SERVICE

I certify that a true and correct copy of the foregoing document (with exhibits attached) was served, as indicated below on The Regents of the University of California, on December 22, 2005.

The Regents of the University of California
c/o: Law Office of Charles E. Krueger
P.O. Box 5607
Walnut Creek, CA 94596-1607

Via First Class Mail



Stephen A. Wight