Exhibit J

Ex. J: Summary of evidence supporting Defendants' proposed constructions

Reprinted below are excerpts from the intrinsic and extrinsic evidence that support Defendants' proposed constructions. Any exhibits referenced below are attached to Defendants' claim construction brief being filed today, with the exception of documents from the prosecution histories, which are being filed separately today in the following packets:

- "906 PH": Prosecution history of U.S. Patent No. 5,838,906
- "359 PH": Abandoned Application No. 09/075,359
- "831 PH": First reexamination of the '906 patent (No. 90/006,831)
- "563 PH": Interference involving the '906 patent (No. 105,563)
- "858 PH": Second reexamination of the '906 patent (No. 90/007,858)
- "985 PH": Prosecution history of U.S. Patent No. 7,989,985

TABLE OF CONTENTS

A. "executable application"

Claim Term(s)	Defendants' Proposed Construction	Eolas's Proposed Construction
executable application	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

1. Defendants' intrinsic evidence

a. Claims

In the following chart, the term(s) for construction appear in all the claims marked with an "x." The letter "m" indicates a method claim, and "¶6" indicates a claim that the Defendants contend includes limitations subject to § 112, ¶ 6.

	'906 patent						'985 patent								
	m 1	m 4	m 5	¶6 6	¶6 9	¶6 10	m 1	¶6 16	m ¶6 20	m ¶6 24	¶6 28	m ¶6 32	m 36	m ¶6 40	m 44
executable application	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

b. Specification (all cites to '906 patent)

Title: Distributed hypermedia method [and system] for automatically invoking ***external application*** providing interaction and display of embedded objects within a hypermedia document

8:56–:57 & fig. 5 (Detailed Description of a Preferred Embodiment): FIG. 5 is an illustration of an embodiment of ***the invention*** using a client computer, server computer and a network.

8:66–:67 (Detailed Description of a Preferred Embodiment): Client computer 200 includes ***processes***, such as browser client 208 and ***application client 210***.

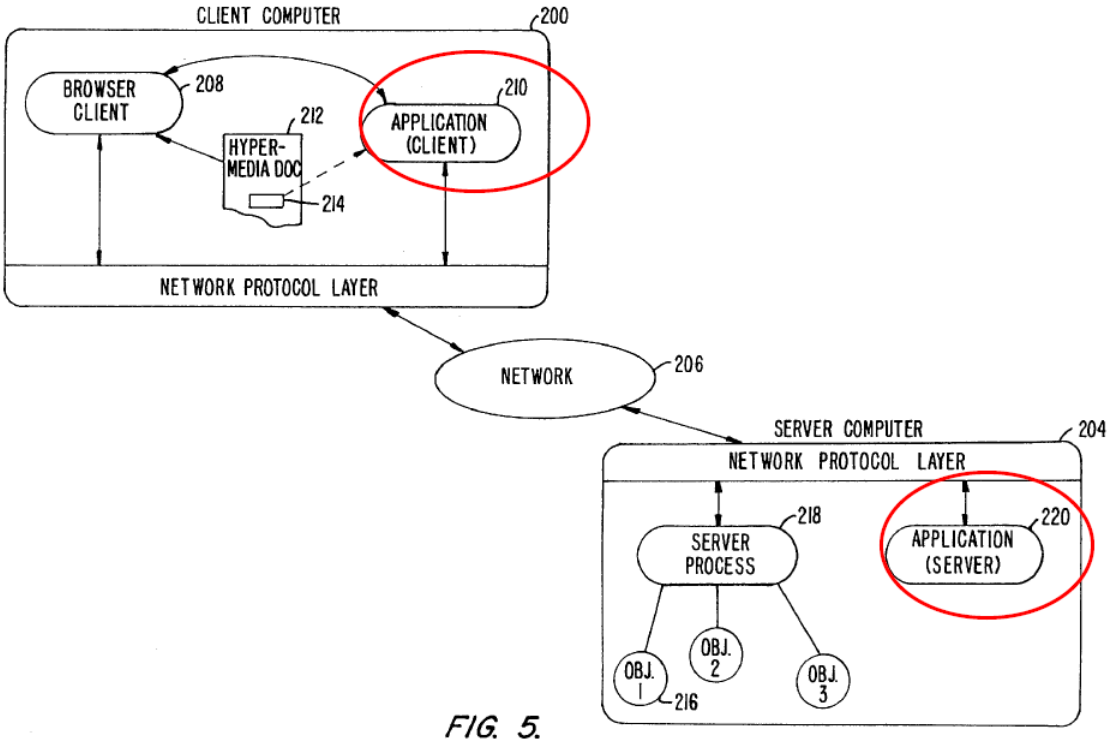


FIG. 5.

9:27–:40 & fig. 5 (Detailed Description of a Preferred Embodiment): In FIG. 5, hypermedia document 212 includes an embedded program link at 214. Embedded program link 214 identifies ***application client 212 [sic: 210]*** as an application to invoke. In this present example, ***the application, namely, application client 210***, resides on the same computer as the browser client 208 that the user is executing to view the hypermedia document. Embedded program link 214 may include additional information, such as parameters, that tell ***application client 210*** how to proceed. For example, embedded program link 214 may include a specification as to a data object that ***application client 210*** is to retrieve and process.

9:41–:45 (Detailed Description of a Preferred Embodiment): When browser client 208 encounters embedded program link 214, it invokes ***application client 210*** (optionally, with parameters or other information) and ***application client 210 executes*** instructions to perform processing in accordance with ***the present invention***.

9:66–10:16 (Detailed Description of a Preferred Embodiment): After ***application client 210*** receives the multidimensional data object 216, ***application client 210 executes instructions*** to display the multidimensional embryo data on the display screen to a user of the client computer 200. The user is then able to interactively operate controls to recompute different views for the image data. ***In a preferred embodiment, a control window is displayed*** within, or adjacent to, a window generated by browser client 208 that contains a display of hypermedia document 212. An example of such display is discussed below in connection with FIG. 9. Thus, the user is able to interactively manipulate a multidimensional image object by means of ***the present invention***. In order to make ***application client 210*** integral with displays created by browser client 208, both the browser client and the ***application client*** must be in communication with each other, as shown by the arrow connecting the two within client computer 200. ***The manner of communication is through an application program interface (API), discussed below.***

10:17–:27 (Detailed Description of a Preferred Embodiment): Browser client 208 is a process, such as NCSA Mosaic, Cello, etc. ***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.

11:17–:39 & fig. 6 (Detailed Description of a Preferred Embodiment): FIG. 6 shows yet another embodiment of ***the present invention***. FIG. 6 is similar to FIG. 5, except that additional computers 222 and 224 are illustrated. Each additional computer includes a ***process*** labeled ***"Application (Distributed)."*** The ***distributed application*** performs a portion of the task that an ***application***, such as ***application server 220*** or ***application client 210***, perform. In the present example, tasks such as volume rendering may be broken up and easily performed among two or more computers. These computers can be remote from each other on network 206. Thus, several computers, such as server computer 204 and additional computers 222 and 224 can all work together to perform the task of computing a new viewpoint and frame buffer for the embryo for the new orientation of the embryo image in the present example. The coordination of the ***distributed processing*** can be performed at client computer 200 by ***application client 210***, at server computer 204 by ***application server 220***, or by any of the ***distributed applications executing*** on additional computers, such as 222 and 224. In a preferred embodiment, distributed processing is coordinated by a ***program called "VIS" represented by application client 210*** in FIG. 6.

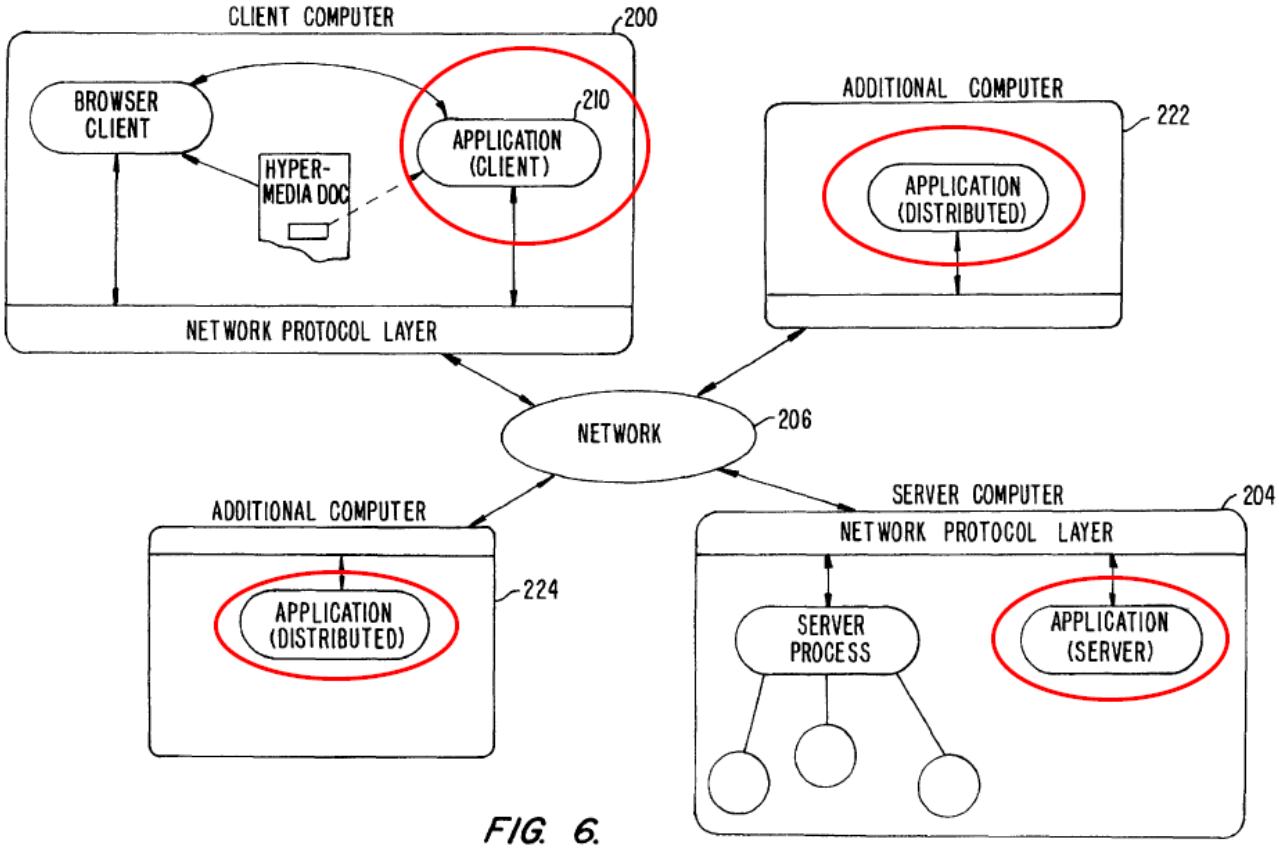


FIG. 6.

12:9–:27 (Detailed Description of a Preferred Embodiment): The various ***processes*** in the system of ***the present invention*** communicate through a custom API called Mosaic/***External Application*** Program Interface MEAPI. The MEAPI set of predefined messages includes those shown in Table I.

TABLE I

Message Function	Message Name
Messages from server to client:	
1. Server Update Done	XtNrefreshNotify
2. Server Ready	XtNpanelStartNotify
3. Server Exiting	XtNpanelExitNotify
Messages from client to server:	
4. Area Shown	XtNmapNotify
5. Area Hidden	XtNunmapNotify
6. Area Destroyed	XtNexitNotify

12:50–:53 (Detailed Description of a Preferred Embodiment): Next, a discussion of the software processes that perform parsing of a hypermedia document and launching of an ***application program*** is provided in connection with Table II and FIGS. 7A, 7B, 8A and 8B.

12:54–:65 (Detailed Description of a Preferred Embodiment): Table II, below, shows an example of an HTML tag format used by ***the present invention*** to embed a link to an ***application program*** within a hypermedia document.

TABLE II

```
<EMBED
    TYPE = "type"
    HREF = "href"
    WIDTH = width
    HEIGHT = height
>
```

13:2–:18 (Detailed Description of a Preferred Embodiment): 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. In the case where TYPE is "application/x-vis" this means that the object at the URL address is a three dimensional image object since the ***program "x-vis"*** is a data visualization tool designed to operate on three dimensional image objects. However, any manner of ***application program*** may be specified by the TYPE element so that other types of ***applications***, such as a ***spreadsheet program***, ***database program***, ***word processor***, etc. may be used with ***the present invention***. Accordingly, the object reference by the HREF element would be, respectively, a spreadsheet object, database object, word processor document object, etc.

14:64–:67 & fig. 8A (Detailed Description of a Preferred Embodiment): FIG. 8A is a flowchart for routine HTMLwidget. HTMLwidget creates display data structures and launches an ***external application program*** to handle the data object specified by the URL in the EMBED tag.

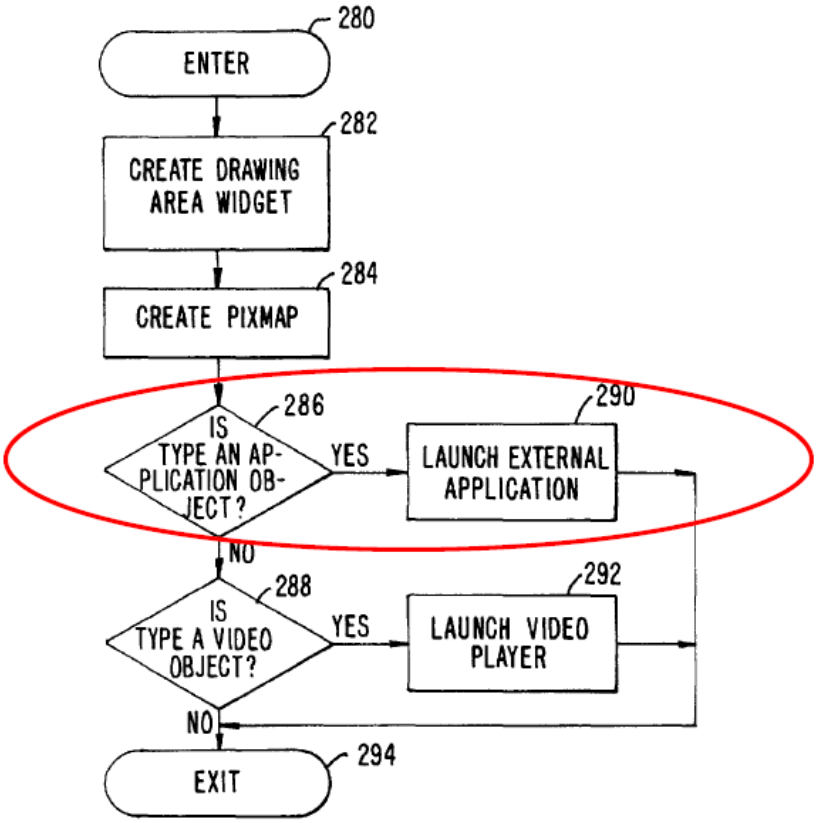


FIG. 8A.

15:9–:21 (Detailed Description of a Preferred Embodiment): At step 286 a check is made as to whether the type attribute of the object, i.e., the value for the TYPE element of the EMBED tag, is an ***application***. If so, step 290 is executed to launch a predetermined ***application***. In a preferred embodiment an ***application*** is launched according to a user-defined list of application type/***application*** pairs. The list is defined as a user-configurable XResource as described in "Xlib Programming Manual." An alternative embodiment could use the MIME database as the source of the list of application type/***application*** pairs. The routine "vis_start_***external_application***" in file HTMLformat.c is invoked to match the application type and to identify the ***application*** to launch.

15:22–:38 (Detailed Description of a Preferred Embodiment): 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 HTMLformat. The ***external application*** is also passed information about the ID of the pixmap, the data URL and dimensions. Codes for communication such as popping-up/iconifying, start notification, quit notification and refresh notification with ***external applications*** and DrawingArea refreshing are also added. Examples of such codes are (1) "setup/start" in vis_register_client and vis_get panel_window in HTMLwidgets.c; (2) "handle messages from ***external applications***" in vis_handle panel_msg in HTMLwidgets.c; (3) "send messages to ***external applications***" in vis_send_msg in HTMLwidgets.c; (4) "terminate ***external applications***" in vis_exit in HTMLwidgets.c which calls vis_send_msg to send a quit message; and (5) "respond to refresh msgs" in vis_redraw in HTMLwidgets.c.

15:58–16:8 (Detailed Description of a Preferred Embodiment): ***The present invention*** allows a user to have interactive control over application objects such as three dimensional image objects and video objects. In a preferred embodiment, controls are provided on the ***external applications'*** user interface. In the case of a ***VIS/panel application***, a ***process, "panel"*** creates a

graphical user interface (GUI) thru which the user interacts with the data. The ***application program, VIS,*** can be ***executing*** locally with the user's computer or remotely on a server, or on one or more different computers, on the network. The ***application program*** updates pixmap data and transfers the pixmap data (frame image data) to a buffer to which the browser has access. The browser only needs to respond to the refresh request to copy the contents from the updated pixmap to the DrawingArea. The Panel process sends messages as "Msg" sending performed by routines such as vis_send_msg and vis_handle panel_msg to send events (mousemove, keypress, etc.) to the ***external application***.

16:8–:28 & fig. 9 (Detailed Description of a Preferred Embodiment): FIG. 9 is a screen display of ***the invention*** showing an interactive application object (in this case a three dimensional image object) in a window within a browser window. In FIG. 9, the browser is NCSA Mosaic version 2.4. The ***processes VIS***, ***Panel*** and VRServer work as discussed above. FIG. 9 shows screen display 356 Mosaic window 350 containing image window 352 and a portion of ***a panel window 354***. Note that image window 352 is within Mosaic window 350 while ***panel window 354*** is external to Mosaic window 350. Another possibility is to have ***panel window 354*** within Mosaic window 350. By using the controls in ***panel window 354*** the user is able to manipulate the image within image window 352 in real time do perform such operations as scaling, rotation, translation, color map selection, etc. In FIG. 9, two Mosaic windows are being used to show two different views of an embryo image. One of the views is rotated by six degrees from the other view so that a stereoscopic effect can be achieved when viewing the images. ***Communication between Panel and VIS is via "Tooltalk" described in, e.g., "Tooltalk 1.1.1 Reference Manual," from SunSoft***.

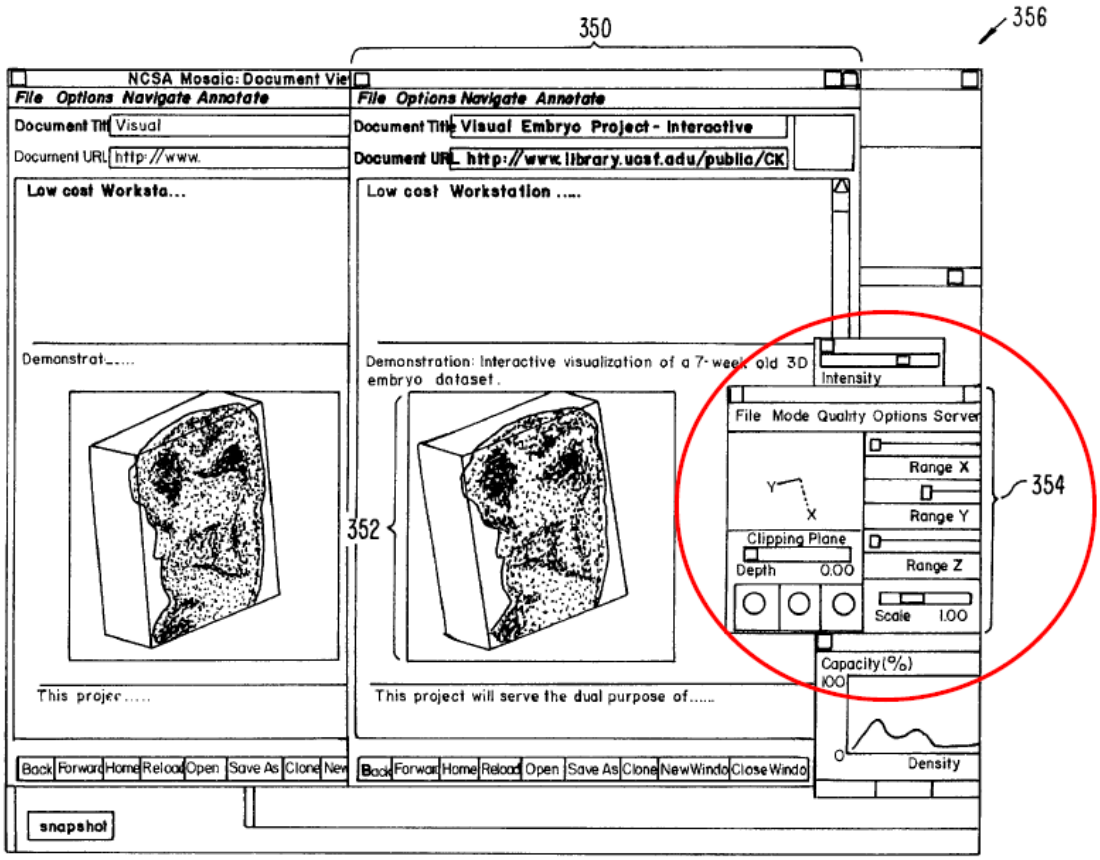


FIG. 9.

16:28–:46 & fig. 10 (Detailed Description of a Preferred Embodiment): FIG. 10 is an illustration of the ***processes VIS***, ***Panel*** and VRServer discussed above. 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. The ***Panel process communicates with the VIS process through a communications protocol (ToolTalk, in the preferred embodiment)*** to exchange visualization command messages and image data. The image data is computed by one or more copies of a process called VRServer that may be executing on remote computers on the network. VRServer processes respond to requests such as rendering requests to generate image segments. The image segments are sent to ***VIS*** and combined into a pixmap, or frame image, by ***VIS***. The frame image is then transferred to the Mosaic screen via communications between ***VIS***, ***Panel*** and Mosaic. A further description of the data transfer may be found in the paper "Integrated Control of Distributed Volume Visualization Through the World-Wide-Web," referenced above.

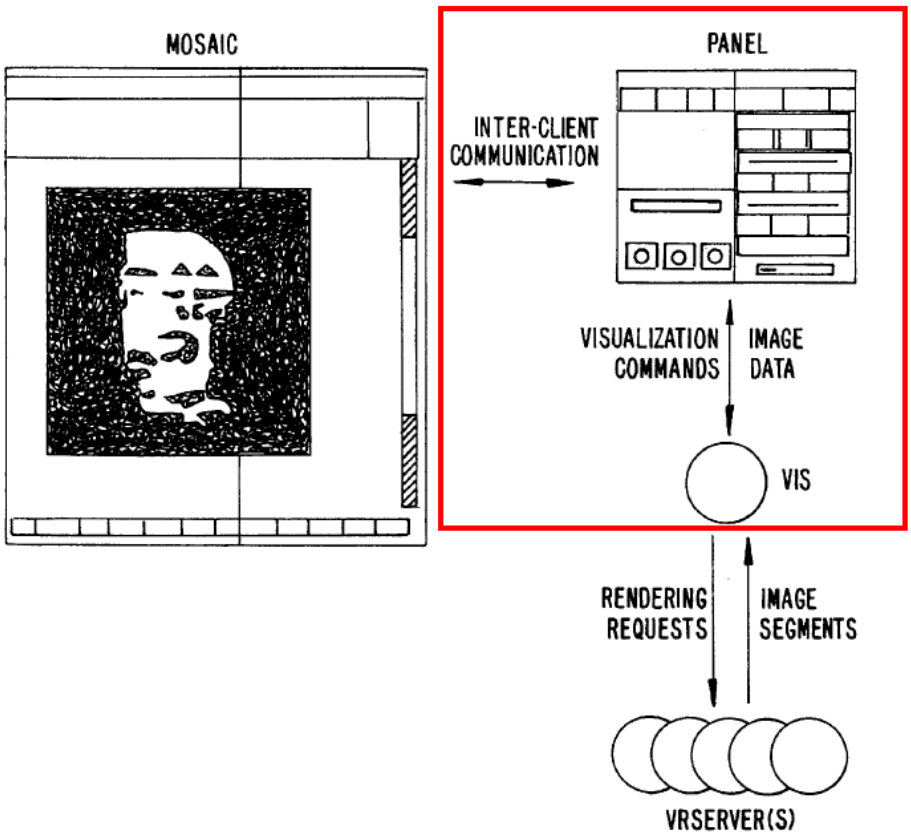


FIG. 10.

c. <u>Prosecution history</u>

i. <u>'906 prosecution history (08/324,443)</u>

Amendment A, at 17 (Aug. 8, 1996) (906 PH Ex. 03 at PH_001_0000783895):

> The operation of Ness scripts is illustrated in the "Extended Birthday Card" example at pages 30 and 31. Note that the object named is: **extend** "visible cake". Both the executable script and the object to be manipulated are within the document.

In view of the above, it is believed that the claims are not obvious over the disclosed prior art in view of Hansen. There 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.

The system of Ness provides for interaction with an object embedded in a document by executing code embedded in the document. However, there is no teaching or suggestion of the claimed system of utilizing a browser to invoke an external application identified by an original document, being displayed by a browser within a first window, to display and process an external object within the first window.

Amendment B, at 15 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784043):

The Examiner relies upon Hansen's teaching that, in a programmer's source code editor, a programmer should have the power to organize various fragments of the program for perusal by a reader, in order to aid the reader in comprehending the program [p256 col.1 1st paragraph]. The Examiner then states: "Hence, it would have been obvious for one of ordinary skill in the art to provide external application to display and process the object within the browser-controlled window because it would have improved the system by reducing the display and aiding the reader comprehension of the hypermedia document."

However, there is no suggestion in Hansen that an executable application external to the programming editor environment should be displayed and interactively processed within a document window. There is no discussion of external application programs at all. The fact that Hansen teaches that it is good to graphically organize the sub-elements of a document for better comprehension would not suggest to the person of skill in the art to combine parts of one reference, Khoyi's object data processing system, with another reference in order to meet Applicants' novel claimed combination.

Amendment B, at 16 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784044):

Turning first to modifying Mosaic, to combine these references as proposed would have required novel and unobvious inventive steps. One must first consider that Mosaic is an application program which operates on any one of three operating systems: UNIX, Windows, and the Mac OS. Much of the current commercial success of the World Wide Web is due to this cross-platform compatibility of Web browsers. The system taught by Khoyi, on the other hand, is *a fully-independent and proprietary operating system*. As is stated in section 1.5 of Khoyi, "The operating system of the present invention differs from the traditional operating system in that, firstly, the actual functions and services performed by the operating system are reduced to the minimum ... functions and services which would normally be performed by an operating system, together with many functions and operations which would normally be performed by the applications programs themselves, are performed by libraries of routines [pack routines]. Examples of services and functions performed by pack routines include, but are not limited to, input/output operations, graphics/text and display operations, file access and management operations, and mathematical operations."

Amendment B at 17, 18–19 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784045):

Therefore, adding the functionality of Khoyi to Mosaic would have been impossible at the time of the Applicants' invention without the creation of new novel and nonobvious technology. Even if such a combination had been possible and operable at the time of the Applicants' invention, Mosaic would have had to be significantly modified in a number of additional complex and nonobvious ways to achieve the combination.

* * * * *

Mosaic would also have to be modified to allow it to be caused to re-render the document window in response to any change in the object imposed by the external program. Such re-rendering would require synchronization messages to be continuously exchanged between the browser application and the external program. This would involve the creation of some kind of message event loop that would wait for re-rendering messages to come from Khoyi's external program.

Similarly Khoyi would have to be modified to synchronize with Mosaic so that changes to the data would cause to external program to send a message to Mosaic to cause it to re-render its display. *Of course, even after doing all of the*

above, the external applications still could not be interacted with from within the Mosaic document, as required by the claimed invention, since Khoyi must launch any external application into a separate window before the reader can interactively control it.

Amendment B at 24 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784052):

Neither Mosaic, nor Khoyi, nor Hansen shows an executable application which is external to a document being displayed and interactively processed within that document's display window, nor do they show such an application where said executable application is interactively controlled on said client workstation by interprocess communications between the external application and the browser. This feature produces surprising and unexpected results over the prior art, since it allows the reader to perform all necessary interactive functions with external applications without directing his or her attention away from the hypermedia document. Additional surprising and unexpected results are yielded by the fact that the hypermedia browser application can have its functionality extended without making any changes to the hypermedia browser's object code. Further, surprising and unexpected results come from the ability of the document author to design interactive hypermedia document content that displays a similar look and feel to the reader, regardless of what the underlying operating system or computer platform the browser program is being executed upon.

Amendment B at 25 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784053):

None of the cited references show this feature. This feature produces the additional unexpected and surprising results over the prior art of allowing the browser application and the external application to precisely coordinate their activity, such as caching of the external application and shutting down its execution when no longer needed, entirely under the control of the browser application, in a manner that is transparent to the user. This drastically clarifies and simplifies the user's use of the hypermedia document and its related embedded applications.

Applicants' Response, at 12–14 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784142-44):

During Applicant's interview with the Examiner on November 6, 1997, the Examiner asserted that since OLE shows an object handler which is dynamic-link library (DLL) code that can automatically be invoked at document rendering time, and since this object handler DLL code may be considered to be part of the server application, then it would have been obvious to enhance Mosaic by providing such a DLL object handler that would be automatically invoked at document rendering time to provide display and interactive processing of the object within a window in the hypermedia document. Applicants respectfully assert that such reasoning is incorrect.

As Koppolu states, "the invoking of a server application can be relatively slow when the server application executes as a separate process from the container application. In certain situations, this slowness may be particularly undesirable, such as, for example, if the user wants to print a compound document that includes many containee objects." Such a problem is solved in the Koppolu system by providing special code that is loaded when the word processor application is launched, where that code provides a subset of the functionality of the full server application. An example would be code which allows printing of embedded or linked spreadsheet data, without having to start up the spreadsheet application itself. This code is called an **"object handler."**

Koppolu does not give much detail about the functioning of object handlers, which are shown to be implemented as dynamic link libraries (DLLs). Full details on object handler implementation and functionality can be found in Brockschmidt.

Brockschmidt comments on object handlers (Chapter 11, section: "Why Use a Handler?"): "There are two main reasons for implementing an object handler to work with your local server: speed and portability. First, an object handler can generally satisfy most requests a container might make on an object such as drawing an object on a specific device or making a copy of the object in another IStorage. Object handlers may also be capable of reloading a linked file and providing an updated presentation to the container. **Object handlers do not, however, provide any sort of editing facilities** for the object itself."

If the entire server application is implemented as an in-process DLL, this is called an "in-process server." The above statement by Brockschmidt shows that the use of the term "object handler" relates specifically to a limited set of object-related facilities that can be automatically invoked by the container application at document rendering time, but which **do not include** capabilities for interactively processing object data. Interactive processing of object data can only be accomplished through interactive invoking of either an in-process server or a local server (standalone executable). In-process servers do allow editing capabilities for object native data, but these editing capabilities are invoked **only after the containee object has been interactively activated by the user,** as described below. This is further supported by the teaching of Koppolu that user interaction with containee objects is provided by OLE only after interactive activation of the containee object server by the user (Col. 7, lines 56-66).

Brockschmidt goes on to describe the problems associated with use of both object handlers and in-process servers. These include 1) limited interoperability, even across different versions of OLE and different versions of Intel processors, and 2) the lack of message loops in DLLs, drastically limiting use of interactive capabilities such as keyboard accelerators. According to Brockschmidt, "The other technical issue of an in-process server specifically (but not a handler) is that since there is nothing that can ever run stand-alone (like a local server EXE can) **there is no possibility to provide linked objects.**" Brockschmidt explains this by pointing out that in-process DLLs cannot access files external to the compound document file. This limits the use of in-process servers to working with embedded (encapsulated) data stored ***within*** the container document file, rather than linked ***external*** data files.

To repeat, Brockschmidt clearly states that "an object handler can generally satisfy most requests a container might make on an object such as drawing an object on a specific device or making a copy of the object in another IStorage. Object handlers may also be capable of reloading a linked file and providing an updated presentation to the container. **Object handlers do not, however, provide any sort of editing facilities** for the object itself." Brockschmidt goes on to emphasize: "When an end-user opens the document in which the object lives and the container application loads the object, it transitions to the loaded state where it may be seen and printed but not edited or otherwise manipulated in any way. **Only when the object is activated does it transition to the running state where the user may perform any number of actions on that object, such as playing or editing the data.**"

Applicants' Response, at 20-21 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784150):

Comparing OLE binary data formats to Mosaic's ASCII text tag mechanism, from the point of view of parsing, would be similar to comparing machine-code programming to the use of a higher level programming language. Similar to OLE's document data files, machine code programs are stored in a binary data structure format that is specifically tailored to the computer processor architecture. They are especially efficient, since they do not require a parser for execution. tag-parsing mechanism, however, is more like using a higher-level programming language. What is given up by Mosaic in terms of run-time performance is recouped through ease of document development, simplification of browser design, and cross-platform compatibility for document viewing.

Notice of Allowability, at 2-3 (Mar. 30, 1998) (906 PH Ex. 17 at PH_001_0000784168):

The claims are allowable over the prior art of record because the prior art does not teach nor reasonably suggest the claimed combination of a browser, while parsing a hypermedia document in a distributed hypermedia environment, automatically invokes an external executable application associated with an embedded object to provide interactive processing and to display the object within an area of the hypermedia document's display window.

The examiner agrees that the claimed external executable application is not a code library extension nor object handler (e.g. windows dll and OLE) as pointed out in applicant's argument (paper #19 pages 12-14).

ii. First reexam (90/006,831)

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 8 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785916):

In the case of the Raggett I "*ismap*" attribute, Raggett explicitly discloses:

> "The *ismap* attribute causes the browser to send mouse clicks on the figure, back to the server using the selected coordinate scheme" [see Raggett I, page 13, 1st sentence under "Active areas"].

As is clearly indicated by Raggett I, it is the browser application that responds to the mouse click that occurs over an active region identified by a coordinate scheme superimposed over a static graphical image. Thus, in the case of Raggett I and active map areas in general (e.g., using the "ismap" attribute and "<figt " tag), it is the browser application that provides the interactivity.

In contrast, the instant '906 claims explicitly require the "interactive processing" to be enabled by an "executable application" that is a separate application from the browser application.

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 54–56 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785962-64):

III. VIOLA SCRIPTS (OR CORRESPONDING BYTE-CODE FORMS) DO NOT ANTICIPATE NOR FAIRLY SUGGEST THE EXTERNAL "EXECUTABLE APPLICATION" AS CLAIMED IN THE '906 PATENT.

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 consistent with the specification, where such interpretation is also consistent with the interpretation that those skilled in the art would reach. In re Hyatt, 211 F.3d 1367, 1372, 54 USPQ2d 1664, 1667 (Fed. Cir. 2000); In re Cortright, 165 F.3d 1353, 1359, 49 USPQ2d 1464, 1468 (Fed. Cir. 1999).

While expert witnesses and dictionaries (considered as extrinsic evidence) may differ regarding the proper construction of the instant claimed "executable application", the Central Processing Unit (i.e., CPU or microprocessor) found in every computer system has only a single, precisely defined interpretation as to what constitutes an "executable application." When the CPU initiates a "fetch and execute" cycle, the program counter is loaded with the address of the next executable instruction. To be "executable" the contents of the memory location pointed to by the program counter must contain an instruction in binary form that is a member of the native instruction set of the microprocessor (i.e., a binary machine language instruction). The binary representation of the precise portion of the machine language instruction that determines what kind of action the computer should take (e.g., add, jump, load, store) is referred to as an operation code (i.e., OP code). From the perspective of the CPU, if a recognizable machine language instruction (i.e., a native CPU instruction) is not found within the memory location pointed to by the program counter, the computer will crash.

The Viola system uses "C-like" Viola scripts that must be INTERPRETED by the browser and then TRANSLATED or CONVERTED into binary native executable machine code that can be understood by the CPU. Alternately, the Viola script is precompiled to intermediate byte-code form and the byte-code is interpreted (i.e., translated) into binary native executable machine code at runtime. This extra step of translation results in an unavoidable performance penalty, as interpreted applications run much slower than compiled native binary executable applications.

Accordingly, the "C-like" Viola scripts (or corresponding byte-code representations) are not "executable applications" from the perspective of the CPU, which is the only perspective that really matters at runtime. A conventional CPU is only capable of processing binary machine language instructions from its own native instruction set.

Without an intermediate translation step performed by an interpreter component of the Viola browser, a Viola script (or corresponding byte-code representation) cannot be processed as an executable application by the CPU.

Significantly, the instant '906 specification is silent regarding the use of applications that rely upon scripts that must be interpreted before they can be executed. The instant '906 specification is silent with respect to interpreting code prior to execution. The instant '906 specification is silent with respect to the use of byte-code intermediate forms.

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 60 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785968):

V. **EVEN ASSUMING, ARGUENDO, THAT "INTERPRETING A SCRIPT" (OR CORRESPONDING BYTE-CODE REPRESENTATION) MAY BE BROADLY CONSIDERED AS EQUIVALENT TO "EXECUTING AN APPLICATION", SUCH INTEPRETATION MERGES THE BROWSER AND THE "EXECUTABLE APPLICATION" INTO ONE PROGRAM THAT FAILS TO TEACH EVERY ELEMENT OF THE '906 PATENT CLAIMS.**

Assuming *arguendo* that one adopts the alternate broader modern construction where "interpreting a script" (or interpreted the corresponding byte-code representation) may be considered as equivalent to "executing an application," then the Viola script arguably becomes an integral component of the Viola browser that parses, interprets (i.e. translates), and executes each line of the script (or corresponding byte-code). 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 [see claims 1 and 6].

iii. Second reexam (90/007,858)

Office Action, at 24, 29, 31–32 (July 30, 2007) (858 PH Ex. 03 at PH_001_0000786969-77):

22. **Claims 1-3 and 6-8** are rejected under 35 U.S.C. 102(e) as being anticipated by Cohen *et al.* (U.S. Patent Number 5,367,621, hereafter "Cohen"), when viewed with "Introducing NCSA Mosaic", written by the Software Development Group, National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, December 1993, being the Defendant's Trial Exhibit Number 226 (hereafter referred to as "NCSA Mosaic").

* * * * *

Regarding ***claim 6***, Cohen discloses

* * * * *

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 [*see Fig. 1a, whereby the external executable application is specified as "DATA = 'graph.exe\GOCA FORMAT C'", seen in Fig. 1a, wherein the graph.exe program is external to the 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 [*see steps 410-426 in Fig. 6 and steps 562-578 in Fig. 7c; also see col. 10, lines 33-60, wherein "In step 412, the link tags are located in the softcopy book text. In particular, the link tags 164, 168, and 172 in the book text of Fig. 1b are located. Then in step 414, a determination is made as to whether any link tags have a link description with the AUTOLAUNCH parameter equaling "yes" in the corresponding link descriptor tag....In step 416 of Fig. 6, if an AUTOLAUNCH parameter is equal to "yes", then the program gets the DATA string from the link description. Reference to Fig. 1a will show that the link description tag 150 has the DATA = 'graph.exe \ GOCA Format C'.... With reference to the graphic object type link descriptor 150 of Fig. 1a, the string 'graph.exe \GOCA Format C' is output by the softcopy book READ program 400 to begin execution of the specified I/O handler program, namely graph.exe, whose flow diagram handler program is shown in Fig. 7c."; also see col. 13, lines 52-67, wherein "In step 566, the data string is parsed to identify if the graphic software support specified by the author in the link descriptor 150, is present in this workstation."*].

Final Rejection, at 4, 6–7 (Apr. 18, 2008) (858 PH Ex. 08 at PH_001_0000787214, 16–17):

Response to Arguments

6. Patent Owner's arguments filed 10/1/07, with respect to the Cohen reference, have been fully considered, but they are not persuasive.

7. First, regarding much of the Patent Owner's arguments, which beginning on page 1, the Patent Owner states that the claim construction set forth in the Markman ruling in the related litigation, which was affirmed by the U.S. Court of Appeals for the Federal Circuit, is utilized in the subsequent remarks. For instance, the Patent Owner argues on page 10 that Cohen fails to expressly teach of the feature of "interactive processing", whereby the Patent Owner interprets the limitation as being the processing of the user utilizing a mouse or keyboard or similar device, to change the structure or presentation of an object. However, the examiner notes that this is not the proper standard for claim construction during examination before the Office, as recognized by the Courts.

* * * * *

11. If the Patent Owner wishes the claims to be narrowed based upon the Court's construction of the claim terms, then the claims must be amended accordingly. The case law above makes clear that claims during examination are interpreted broadly not narrowly. To incorporate the limitations into the claims would countermand the case law prohibiting reading of limitations from the specification into the claims during examination. The same standard applies to issued patents under reexamination because, the statutory presumption of validity, 35 U.S.C. 282, has no application in reexamination (*In re Etter*, 756 F.2d 852, 225 USPQ 1 (Fed. Cir. 1985)).

12. With this, in response to the Patent Owner's arguments on page 10, which argue that Cohen fails to expressly teach of the feature that "enable[s] interactive processing of said object", whereby, as noted above, the Patent Owner defines "interactive processing" as being a processing of the user utilizing the mouse or keyboard or similar device, that changes the structure or presentation of the object, thus being an interactive process. However, the current claim language does not specify this. Further, there is no requirement that the "interactive processing" be a process performed by the "user".

13. Further, on pages 10 and 11 of the Patent Owner's arguments, the Patent Owner additionally provides sections in the specification of the '906 Patent that describe the "interactive processing". However, the examiner notes that it is improper to import features found in the specification into the claim language. In this regard, MPEP 2111.01 [R-5] states under the heading "II. IT IS IMPROPER TO IMPORT CLAIM LIMITATIONS FROM THE SPECIFCATION":

> "Though understanding the claim language may be aided by explanations contained in the written description, **it is important not to import into a claim limitations that are not part of the claim**. For example, a particular embodiment appearing in the written description may not be read into a claim when the claim language is broader than the embodiment." *Superguide Corp. v. DirecTV Enterprises, Inc.*, 358 F.3d 870, 875, 69 USPQ2d 1865, 1868 (Fed. Cir. 2004). [Emphasis added.]

14. Currently, claim 1 states "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 …". With this, the reference of Cohen states in column 9, lines 41-49, that "The profile 300 includes the hardware types for a particular I/O function, characteristics for each hardware type, and the software drivers, which **enable** the application programs and I/O handler programs to **interact** with the particular I/O hardware or software." [Emphasis added]. Thus, Cohen recognizes that the application programs perform an interactive process with the particular I/O hardware or software, and are enabled by the profile 300. There is no limitation in the current claim language that particularly requires that the process of interactive processing includes the function that the user, by using a mouse or keyboard or similar input device, can change the structure or presentation of the object, as argued. If the Patent Owner wishes that this function be considered, the Patent Owner must add the particular language to the claim.

Amendment, at 2, 5–6 (June 17, 2008) (858 PH Ex. 10 at PH_001_0000787265-66):

AMENDMENT TO THE CLAIMS:

Please amend the following claims.

* * * * *

6. (Amended) A computer program product for use in a system having at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment, the computer program product comprising:

a computer usable medium having computer readable program code physically embodied therein, said computer program product further comprising:

computer readable program code for causing said client workstation to execute a browser application to parse a first distributed hypermedia document to identify text formats included in said distributed hypermedia document and to respond to predetermined text formats to initiate processes specified by said text formats;

computer readable program code for causing said client workstation to utilize 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 an end-user to directly interact with [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.

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 60 (Sept. 10, 2008) (858 PH Ex. 11 at PH_001_0000787398-400):

STATEMENT OF REASONS FOR PATENTABILITY AND/OR CONFIRMATION

The following is an examiner's statement of reasons for patentability and/or confirmation of the claims found patentable in this reexamination proceeding:

Claims 1-14 are deemed as patentable, as amended.

With the amendment dated 6/23/08, ***claims 1, 4, 5, 6, 9, and 10*** are independent.

With respect to independent ***claims 1 and 6***, in the examiner's opinion, based on the prior art of record, it would not have been obvious to have the system, as claimed, include the features of an embed text format being parsed by the browser to automatically invoke the executable application to execute on the client workstation in order to display the object and enable an end-user to directly interact with the object within a display area created at the first location within the portion of the hypermedia document within the browser controlled window. The examiner notes that the closest prior art, Cohen (U.S. Patent Number 5,367,621), utilizes the IBM BookManager system, whereby Cohen teaches of the AUTOLAUNCH function, which automatically launches an object, whereby the system can automatically invoke multimedia objects, such as "photographic quality graphics, motion video, or sound", as read in col. 2, lines 50-66.

However, 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 window after the object is automatically invoked. Cohen shows that the graphic 190', as seen in Fig. 4b is automatically invoked. However, there is no indication that an end-user can directly interact with this graphic. Further, the specification of Cohen discusses inserting an audio object "eleph_sound.Audio 1 – Elephant's trumpet" and a movie object "eleph_movie.Motion Picture of African Elephant family", as seen in Fig. 1b. But these examples are not automatically invoked using the AUTOLAUNCH function, and if they would be set to AUTOLAUNCH, there is no indication in Cohen that would provide the function allowing the end-user to directly interact with the automatically invoked object.

As noted in the specification of the '906 Patent in col. 7, lines 12-15 "Also, the user is able to rotate, scale and otherwise reposition the viewpoint with respect to these images without exiting the hypermedia browser software." There is no indication in Cohen that the BookManager READ program allows the end-user to perform this direct interaction of the object once the multimedia is launched automatically. Further, the examiner can find no other teaching in the prior art of record that would motivate one of ordinary skill in the art to modify the Cohen teachings so as to allow the end-user to directly interact with the automatically invoked object. Therefore, because of this feature that was added in the amendment dated 6/23/08, the invention defined in claims 1 and 6 is rendered as patentable.

iv. '985 prosecution history (10/217,955)

Notice of Allowability, at 2 (Mar. 20, 2009) (985 PH Ex. 15 at PH_001_0000784733):

The following is an examiner's statement of reasons for allowance: the claims are allowable as the claims contain the subject matter deemed allowable in both Re exam 90/006,831 and Re exam 90/007,838 for the same reasons as set forth in the NIRC of the two Re exams.

2. Defendants' extrinsic evidence

a. Dictionaries

Barron's Dictionary of Computer Terms 119 (2d ed.1989) ("execute") [Ex. Q at PA-0000333370]:

> **EXECUTE** To execute an instruction is to do what the instruction says to do. A computer alternates between a fetch cycle, when it locates the next instruction, and an execute cycle, when it carries the instruction out. (See **computer design.**)

Barron's Dictionary of Computer Terms 202 (2d ed.1989) ("module") [Ex. Q at PA-0000333376]:

> **MODULE** A module is a part of a larger system. For example, a Lunar Module is a part of the Apollo rocket system. A module in a computer program is a part of the program that is written and tested separately and then is combined with other modules to form the complete program. (See **top-down programming.**)

21st Centry Dictionary of Computer Terms 13 (1994) ("application program") [Ex. X at PA-0000333433]:

> **application program** Computer programs designed to enable users to perform specific job functions. Word processing, accounting, and engineering programs are examples of application programs.

21st Centry Dictionary of Computer Terms 130 (1994) ("executable file") [Ex. X at PA-0000333436]:

> **executable file** A file that contains program code that cannot be understood by humans but can be directly executed by the computer.

Microsoft Press Computer Dictionary 23–24 (2d ed. 1994) ("application") [Ex. Y at PA-0000333408] – [PA-0000333409]:

application A computer program designed to help people perform a certain type of work. An application thus differs from an operating system (which runs a computer), a utility (which performs maintenance or general-purpose chores), and a language (with which computer programs are created). Depending on the work for which it was designed, an application can manipulate text, numbers, graphics, or a combination of these elements. Some application packages offer considerable computing power by focusing on a single task, such as word processing; others, called integrated software, offer somewhat less power but include several applications, such as a word processor, a spreadsheet, and a database program.

Microsoft Press Computer Dictionary 90 (2d ed. 1994) ("computer program") [Ex. Y at PA-0000333412]:

computer program A set of instructions in some computer language, intended to be executed on a computer to perform a useful task. The term usually implies a self-contained entity, as opposed to a routine or a library. *Compare* library, routine; *see also* computer language.

Microsoft Press Computer Dictionary 137–38 (2d ed. 1994) ("dynamic link library") [Ex. Y at PA-0000333414–15]:

dynamic-link library A feature of the Microsoft Windows family of operating systems and the OS/2 operating system that allows executable routines—generally serving a specific function or set of functions—to be stored separately as files with DLL extensions and to be loaded only when

needed by the program that calls them. A dynamic-link library has several advantages. First, because a dynamic-link library is loaded only when it is needed, it does not consume any memory until it is used. Second, because a dynamic-link library is a separate file, a programmer can make corrections or improvements to only that module without affecting the operation of the calling program or any other dynamic-link library. Finally, because a dynamic-link library often contains related functions—for example, routines for creating animation on a video display—a programmer can use the same dynamic-link library with other programs.

Microsoft Press Computer Dictionary 153 (2d ed. 1994) ("executable program") [Ex. Y at PA-0000333416]:

executable program A computer program that is ready to run. The term usually refers to a compiled program that has been translated into machine code in a format that can be loaded into memory and run; however, for interpreted languages it can simply refer to source code in the proper format. Applications such as word-processing programs are executable programs. The user does not have to alter the program in any way before being able to run it. *See also* code, compiler, computer program, interpreter, source code.

Microsoft Press Computer Dictionary 236 (2d ed. 1994) ("library") [Ex. Y at PA-0000333420]:

library In programming, a collection of routines stored in a file. Each set of instructions in a library has a name, and each performs a different, often very specific, task. For example, the *printf()* function is part of the Standard C library and displays characters on the screen. Such sets of instructions simplify work and prevent duplication of effort each time a particular task needs to be carried out. A programmer can identify a library to a program, refer to library routines in the program, and have the program carry out the appropriate tasks without having to write (or rewrite) the instructions themselves each time they are needed. Libraries can include standard routines for a particular programming language, or they can contain customized routines written by the programmer.

Also, as in its traditional sense, any collection of information; sometimes used to refer to software or data files.

Microsoft Press Computer Dictionary 319 (2d ed. 1994) ("program") [Ex. Y at PA-0000333425]:

program Synonymous with *software;* a sequence of instructions that can be executed by a computer. The term can refer to the original source code or to the executable (machine language) version. The term *program* implies a degree of completeness; that is, a source code program comprises all statements and files necessary for complete interpretation or compilation, and an executable program can be loaded into a given environment and executed independently of other programs. *See also* program creation, routine, statement.

b. Testimony

Doyle direct, Trial Tr., *Eolas Techs Inc. v. Microsoft Corp.*, No. 99-C-626 (N.D. Ill. 2003), at 303:15–306:14 (July 9, 2003) [Ex. GG (EOLASTX-E-0000000644)]:

Q What type of software were you using when you arrived at the University of California at the time that you were developing this online medical library?

A Well, the project had been working with AT&T's Bell Labs that had a software program called Write Pages that was a proprietary kind of hypermedia browser that could allow someone to browse through something that looked like a series of journal covers. You could click on the journal cover, it would pull up that issue of the journal. You could click on the table of contents of the journal and it could pull up an article, and you could browse through and view it. But it was able to work with a self-contained database of information.

Q Was it a web browser in the sense of the Mosaic browser?

A No, not at all.

Q What limitations, if any, did you uncover as a result of trying to develop an online medical library for use by physicians around the world?

A Well, the biggest limitation struck me right away was that if we wanted to do anything new with this we were limited, severely limited, because ***if we wanted to add a new data type***, for example, a new kind of image that might be able to be used in one of these articles, we'd have to go and request to the programmers at Bell Labs that they add a new kind of image format, and then ***they would have to rewrite a new version of the software that put all of the code necessary, all the -- you know, the program instructions necessary to render this new kind of data to allow the user to work with this new kind of data***.

And we realized that if this -- if we were going to work on something that would be more generally useful that there would be no end to the number of kinds of data that we'd want the system to be able to handle, and ***it was just an unworkable situation*** to think that we could use this thing to do new, innovative kinds of research.

Q So if you wanted to add a new type of medical image or support so that that Write Pages software could display a medical image, would you actually have to then go back to Bell Labs and ask them to rewrite the browser?

A That's correct. They'd have to add this new capability of the browser. They would then send us back a browser that would be bigger than the one that they sent us before because they added this new software to it. And then if we wanted to do anything else new with it, then we'd have to send it back to them, ask them to rewrite it and add more stuff to it, and we'd get what we call ***browser bloat***. The ***browser would just continue to grow and grow and grow, and eventually you'd have, you know, just an enormous application***.

Q Did you undertake to solve some of the problems or limitations in the Bell Lab software that you were using as part of your research for this online medical library?

A Yes, we did. As soon as I started at the university in California and we started talking about these new kinds of projects, one of the things, for example, I wanted to be able to do was to create a new kind of way to display an article where, for instance, if it's for a radiology

journal, you know, radiologists are the kind of doctors who look at X-rays and MRIs, I wanted to be able to allow the scientist or the doctor to actually be able to see the actual data and interact with it rather than seeing, you know, the author's one preferred view of that data in a still image. And we had seen the Mosaic web browser at Illinois, and we knew that it was freely available for academic researchers to use, and so that source code was available, and we looked at it, and we thought, well, we can use this system and start building on this base to add new functionality and create an entirely new kind of web browser.

Q Did you continue to develop this idea then?

A Yes, we did. We were thinking about a project that was -- we were considering working on relating to brain research, and so we thought we'd use this as a reason to start considering this, and so we started looking at the idea of coming up with a way to create the capabilities that eventually came -- became possible ***in the '906 invention***, the ability to allow pages to embed interactive programs in them where ***you don't have to add the actual executable code to the browser or to the document itself***.

Doyle cross, Trial Tr., *Eolas Techs Inc. v. Microsoft Corp.*, No. 99-C-626 (N.D. Ill. 2003), at 477:20-478:4 (July 10, 2003) [Ex. GG (EOLASTX-E-0000000644)]:

Q. . . . [H]ow many lines of code did you add to the Mosaic browser code?

A. I don't recall exactly.

Q. Does 305 sound about right?

A. Could be.

Q. Does 100,000 sound about right for the total Mosaic code?

A. Could be.

Q. So everything you did, you did with 305 lines of code that were written and added to Mosaic?

A. Sounds like it could be on the browser side.

Doyle redirect, Trial Tr., *Eolas Techs Inc. v. Microsoft Corp.*, No. 99-C-626 (N.D. Ill. 2003), at 537:1–:10 (July 10, 2003) [Ex. GG (EOLASTX-E-0000000644)]:

Q. One last question, Dr. Doyle. Do you remember when Mr. Pritikin was talking to you about the number of lines with code Mosaic versus the lines of your invention?

A. Yes.

Q. Can your invention be weighed in terms of lines with code?

A. No. In fact, that was part of the point. ***We were trying to prevent what's called browser blow [sic: bloat]***. In the software industry the more functionality you can get out of the ***fewer lines of code means you have a more elegant solution***.

Martin direct, Trial Tr., *Eolas Techs Inc. v. Microsoft Corp.*, No. 99-C-626 (N.D. Ill. 2003), at 20:18–21:13[1] (July 10, 2003) [Ex. GG (EOLASTX-E-0000000644)]:

Q What did you decide to do at that point?

A Succinctly, what we decided to do was to go and look at all the different applications that had been already written like VIS to handle other data types and try and figure out a way to take their output and put it back into the browser so that a user would see one composite, if you will, view of the page without having to make the browser any bigger.

Q And what were the -- were there -- ***were there advantages to continuing to make the browser bigger***?

A ***None -- no technical ones.***

Q Do programmers judge the merits of their programming by the number of lines of code they add to a program?

A Well, there's always a certain machismo, if you will, speaking as a -- since most of the programmers that I grew up with were male, but -- in terms of how much code you could write, but for the sophisticated programmer it really counts more ***for elegance, minimization, the principle of the smallest and simplest answer is the right one***. So we wanted to find ways of doing things without having to either, A, write all the code in the world or, B, break a lot of stuff by reimplementing things.

[1] In other versions of this transcript the testimony appears on pages 574:18–575:13.

B. "automatically invoke" (in various contexts)

Claim Term(s)	Defendants' Proposed Construction	Eolas's Proposed Construction
automatically [invoking / invoke] [the / said] executable application	the executable application is launched to permit a user to interact with the object without any intervening activation of the object by the user	automatically calling or activating the executable application
executable application is automatically invoked by the browser		executable application is automatically called or activated by the browser

1. Defendants' intrinsic evidence

a. Claims

In the following chart, the term(s) for construction appear in all the claims marked with an "x." The letter "m" indicates a method claim, and "¶6" indicates a claim that the Defendants contend includes limitations subject to § 112, ¶ 6.

	'906 patent						'985 patent								
	m 1	m 4	m 5	¶6 6	¶6 9	¶6 10	m 1	¶6 16	m ¶6 20	m ¶6 24	¶6 28	m ¶6 32	m 36	m ¶6 40	m 44
automatically [invoking / invoke] [the / said] executable application	x	x	x	x	x	x	x	x	x				x	x	
executable application is automatically invoked by the browser										x	x	x			x

b. Specification (all cites to '906 patent)

Title: Distributed hypermedia method [and system] for ***automatically invoking*** external application providing interaction and display of embedded objects within a hypermedia document

3:13–:16 (Background of the Invention): [In the prior art] Many viewers exist that handle various file formats such as ".TIF," ".GIF," formats. When a browser program ***invokes*** a viewer program, the viewer is ***launched*** as a separate process.

6:35–:39 (Background of the Invention): [In the prior art] Users are limited to traditional hypertext and hypermedia forms of selecting linked data objects for retrieval and ***launching*** viewers or other forms of external software to have the data objects presented in a comprehensible way.

7:1–:4 (Summary of the Invention): Interprocess communication between the hypermedia browser and the embedded application program is ongoing after the program object has been ***launched***.

9:28–:31 (Detailed Description of a Preferred Embodiment): In FIG. 5, hypermedia document 212 includes an embedded program link at 214. Embedded program link 214 identifies application client 212 [sic: 210] as an application to ***invoke***.

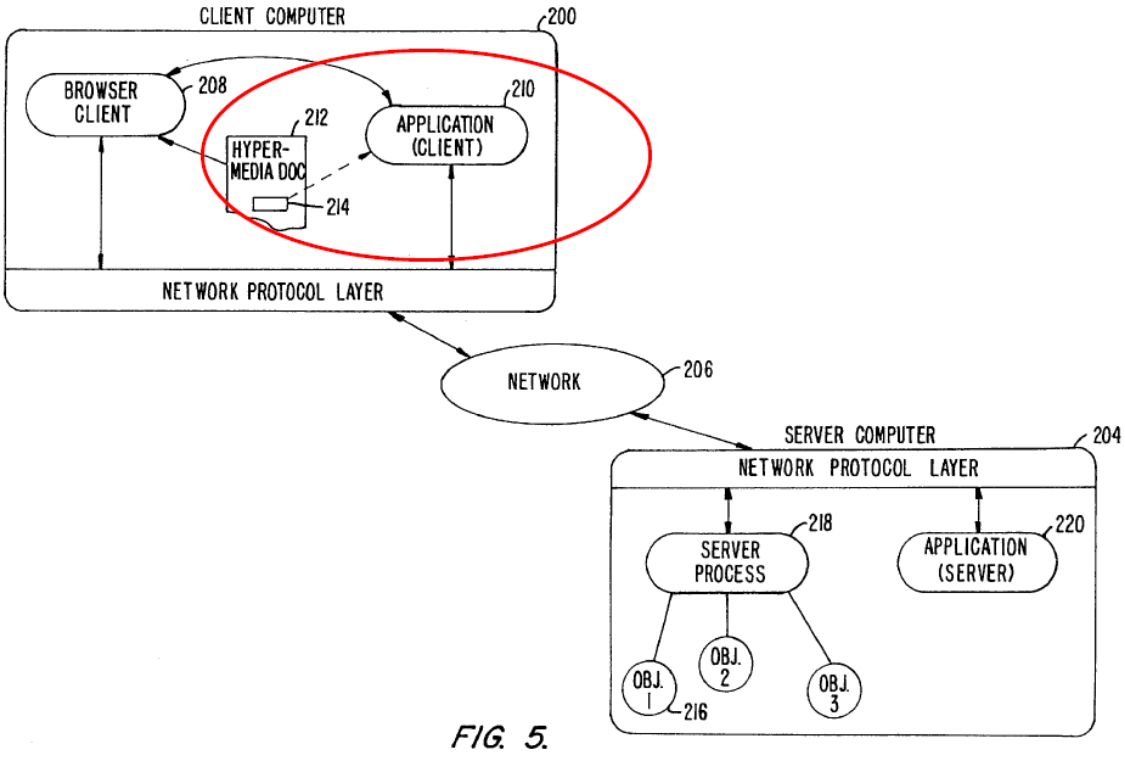


FIG. 5.

9:41–:45 (Detailed Description of a Preferred Embodiment): When browser client 208 encounters embedded program link 214, it ***invokes*** application client 210 (optionally, with parameters or other information) and application client 210 executes instructions to perform processing in accordance with ***the present invention***.

12:50–:53 (Detailed Description of a Preferred Embodiment): Next, a discussion of the software processes that perform parsing of a hypermedia document and ***launching*** of an application program is provided in connection with Table II and FIGS. 7A, 7B, 8A and 8B.

12:66–13:5 (Detailed Description of a Preferred Embodiment): As shown in Table II, the EMBED tag includes TYPE, HREF, WIDTH and HEIGHT elements. The 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.

13:19–:31 (Detailed Description of a Preferred Embodiment): TYPE values such as "video/mpeg", "image/gif", "video/x-sgi-movie", etc. describe the type of data that HREF specifies. This is useful where an external application program, such as a video player, needs to know what format the data is in, or where the browser client needs to determine which application to ***launch*** based on the data format. Thus, the TYPE value can specify either an application program or a data type.

14:64–:67 (Detailed Description of a Preferred Embodiment): FIG. 8A is a flowchart for routine HTMLwidget. HTMLwidget creates display data structures and ***launches*** an external application program to handle the data object specified by the URL in the EMBED tag.

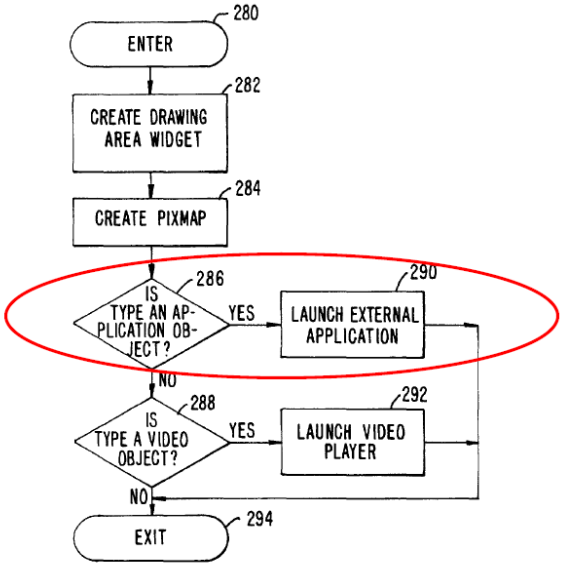


FIG. 8A.

15:9–:12 & fig. 8A (Detailed Description of a Preferred Embodiment): At step 286 a check is made as to whether the type attribute of the object, i.e., the value for the TYPE element of the EMBED tag, is an application. If so, step 290 is executed to ***launch*** a predetermined application.

15:49–:52 (Detailed Description of a Preferred Embodiment): FIG. 8B is a flowchart for routine HTML. Routine HTML takes care of "shutting down" the objects, data areas, etc. that were set up to ***launch*** the external application and display the data object.

c. Prosecution history

i. '906 prosecution history (08/324,443)

Amendment B, at 2, 4 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784032): To overcome the Mosaic prior art, the applicant amended the independent claims to require "***automatically invoke*** said executable application."

Amendment B, at 11 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784039):

A. Mosaic does not disclose the recited embed text format that is parsed by the browser to initiate processing to automatically invoke an executable application external to the hypermedia document.

As described above, in Mosaic the URL is an address to an object. A URL anchor in Mosaic is not an embed text format that is parsed by the browser to initiate invocation of an executable application external to the document. Rather, when the anchor is activated, by the user interactively selecting the anchor, the browser retrieves the object and, if the object is another hypermedia document, replaces the first document with the second document. If the object has a file name associated with a helper application the application is launched and the object is viewed and/or edited in a separate window controlled by the helper application.

Accordingly, the external application is not automatically invoked as a result of the browser parsing the hypermedia document text, as required by the claim, but rather it is invoked by an interactive command given by the user, namely interactively selecting the URL anchor.

Amendment B, at 12 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784040):

As set forth in the attached Doyle declaration, the claimed invention "lifted the glass" of the browser display to allow interactive control of document elements while being displayed in the browser controlled window. The Applicants' claimed invention allowed these elements to become "active" or "live" without requiring external programs to be first launched by the user's interactive commands.

Declaration of Michael D. Doyle, ¶¶ 4–5 (Oct. 28, 1997) (accompanying Applicants' Response (Dec. 23, 1997)) (906 PH Ex. 15 at PH_001_0000784130):

In addition, when the browser parses an <EMBED> tag in the document, the browser automatically launches the external application specifying the location of the visual object to render and identify the shared image buffer. The format and operation of an EMBED tag for 3D image data is described at paragraph 3.1.

5. As stated in ATTACHMENT B, starting at the bottom of page 2, interface and control software had been developed that allows the embedding of a visualization application within a Mosaic document. As is apparent from the photographs, the object is displayed and processed within the browser-controlled window. The visualization application is external to the hypermedia document displayed by the browser. Automatic launching of the external application when an HTML document is opened by the browser is depicted in the video.

Applicants' Response, at 2 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784132): (emphasis in original):

When an embed text format is parsed by the browser, the executable application is **automatically invoked** as a result of the parsing to execute on the client workstation.

Applicants' Response, at 3 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784133):

Turning to the first part of the argument, there is no disclosure or suggestion in Mosaic or Koppolu of automatically invoking an external application when an embed text format is parsed. Each of those references require user input, specifically clicking with a mouse pointer, to activate external applications to allow display and interaction with an external object.

Applicants' Response, at 7 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784137):

Mosaic launches helper applications in response to a user's interactive command, in a separate window to view certain types of file types. As described in the specification, the mechanism for specifying and locating a linked object is an HTML anchor "element" that includes an object address in the format of Uniform Resource Locator (URL). (Application at pg. 3, line 30).

Many viewers exist that handle various file formats such as ".TIF," ".GIF," etc. When a user commands the browser program to invoke a viewer program, typically by clicking on an anchor with a mouse, the viewer is launched as a separate process.

Applicants' Response, at 8 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784138):

Koppolu's OLE system provides a method for interacting with a contained object within the window environment of a container application of a container document. When a user interactively selects a bitmapped image of the contained object, the method integrates a plurality of the server resources with the displayed container window environment. When the user then interactively activates the previously-selected object image, OLE invokes a server application to process the original data referenced by the contained object image. Since OLE was designed for integration of very large programs, a facility is provided whereby the server application can conserve space on the computer display by integrating the server application's menu and GUI system with that of the container application.

Applicants' Response, at 10–11 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784140 – 141):

In this example, the word processor is described as a "container application" and the spreadsheet application is termed a "server application." It is important to note here that the container document does not automatically launch the server application at document-rendering time, but rather, the **user must issue two separate interactive commands** after the time of document rendering in order to cause the container application to invoke the server application.

Applicants' Response, at 13 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784143):

If the entire server application is implemented as an in-process DLL, this is called an "in-process server." The above statement by Brockschmidt shows that the use of the term "object handler" relates specifically to a limited set of object-related facilities that can be automatically invoked by the container application at document rendering time, but which **do not include** capabilities for interactively processing object data. Interactive processing of object data can only be accomplished through interactive invoking of either an in-process server or a local server (standalone executable). In-process servers do allow editing capabilities for object native data, but these editing capabilities are invoked **only after the containee object has been interactively activated by the user**, as described below. This is further supported by the teaching of Koppolu that user interaction with containee objects is provided by OLE only after interactive activation of the containee object server by the user (Col. 7, lines 56-66).

Applicants' Response, at 14 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784144):

> In view of the above, it is clear that the Koppolu (OLE) reference does not disclose or suggest the missing features. In OLE, when a compound document is opened, static pictures of included objects are rendered in presentation format. Invoking of an external application requires a user-activated selection of the object. The object handlers provide no interactive control of a displayed object.

Applicants' Response, at 22 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784152):

> Since OLE was designed to integrate a group of already-popular large programs, such as Microsoft Excel and Microsoft Word, OLE provided the advance that allowed these applications to be interactively invoked to change the GUI environment that surrounded the compound document so that the user would not have to move his or her attention to an external windowing environment in order to edit the object. Since these external applications were intended to be invoked only for editing purposes, and since that invocation inevitably resulted in a modification of the container application's GUI, it made sense that OLE containee server applications could ***not*** be automatically invoked to allow interactive processing of object data. In fact OLE forced the user to make not one but ***two*** interactive commands prior to server invocation, thereby reducing the possibility that one of these large external applications would be inadvertently invoked.

Applicants' Response, at 24-25 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784154):

> Koppolu also teaches away from the use of OLE for networked data distribution. There is no provision, suggestion, or motivation in Koppolu to provide for automatic invocation of a server application to allow interactive processing of object data when a container document is viewed. Furthermore, there is no suggestion in either Mosaic or Koppolu of modifying Mosaic so that an external application, by analogy to Koppolu the server application, is automatically invoked at the time of Web document rendering to display and enable interactive processing of the object within an embedded window within the Web document.
>
> * * * * *

Additionally, because the claimed embed text formats in the document cause the browser to automatically invoke the external application, the hypermedia ***document*** itself, and by implication the author of that document, ***directly*** control the extension of the functionality of the browser. As a consequence of the features of the claimed invention, the ***document***, rather than the browser, **becomes the application;** that is, the document together with its embedded program objects, exposes all the functionality that the user needs to interact with and process the entire content of the compound hypermedia document.

Notice of Allowability, at 2–3 (Mar. 30, 1998) (906 PH Ex. 17 at PH_001_0000784168-69):

The claims are allowable over the prior art of record because the prior art does not teach nor reasonably suggest the claimed combination of a browser, while parsing a hypermedia document in a distributed hypermedia environment, automatically invokes an external executable application associated with an embedded object to provide interactive processing and to display the object within an area of the hypermedia document's display window.

ii. Abandoned application (09/075,359)

Amendment C at 5, 6–7 (Nov. 29, 2001) (359 PH Ex. 06 at PH_001_0000787827-29):

However, as will be described below, the two actions of real-time updating and automatic invoking are completely different.

As set forth above, in Risberg the program itself is configured by a user to establish a connection to a service. The service, which is external to the program, automatically forwards updates to the program for display in an active document. Thus, in response to the user selecting the data, such as IBM stock quotes, the service automatically sends updates of the latest quotes. Accordingly, there is no "invoking" by the Risberg program to cause real-time updates, in fact, the program passively receives the updates provided by the subscription service and displays the latest data.

In contrast, in the claimed system there is no user configuration of the browser program to invoke the external program code. The browser program parses a hypermedia document and automatically invokes the external computer code when an embed text format is parsed. The user of the browser program takes no action in the invoking of the external program code.

* * * * *

In Mosaic, viewer programs may be invoked by the browser in response to user selection of a link to a file format that cannot be displayed by the browser. There is no teaching in Mosaic of automatic invoking.

In Risberg, updates to information being displayed are provided automatically from an external source, e.g., the Dow Jones server. However, the information which is provided in response to user selection during set up of the active object. For example, if a user creates a quote object he selects a Market Type attribute, such as equity, option, or future, and a Symbol attribute which selects the specific symbol, i.e., stock, to be used for the quote. The user also creates scripts, using the scripting language provided by the program, to create macros to perform often used functions.

iii. First reexam (90/006,831)

Examiner Interview Request, at 3 (Apr. 22, 2004) (831 PH Ex. 05 at PH_001_0000785313):

2. The statement in Raggett I that sophisticated browsers could link to an external editing application teaches away from the claimed element of automatically invoking an executable application in order to display the object and to enable in-place interaction.

Interview with Examiner Andrew Caldwell (April 27, 2004) (831 PH Ex. 06 at PH_001_0000785332):

- "linked to"
  - Means hyperlinked
    - Therefore the editor is **not automatically invoked**

Declaration of Edward W. Felten, ¶ 48 (May 7, 2004) (accompanying Applicants' Response (May 11, 2004)) (831 PH Ex. 10 at PH_001_0000785445):

The reference to "linking" to an "external" program refers to the use of a hyperlink or button that the user can click to launch a separate program, as is done with helper applications. (Having the browser automatically invoke an editor wouldn't make sense anyway, since only the page's author would be in a position to edit a copy of the page that anybody else would see, and it wouldn't make sense to invoke an editor automatically when ordinary users had no reason to want to invoke it.)

Applicant's Response, at 3 (May 11, 2004) (831 PH Ex. 07 at PH_001_0000785361):

A. The Claimed Invention.

The invention, as recited for example in claims 1 and 6, is for use in a system having at least one client workstation and one network server coupled to a network environment.

The claims recite a browser application, executed on the client workstation, that parses a hypermedia document to identify text formats in the document and responds to predetermined text formats to initiate processing specified by the text formats.

The browser displays a portion of a first distributed hypermedia document, received over the network from the network server, in a browser-controlled window. The hypermedia document includes an embed text format, located at a first location in the hypermedia document, that specifies the location of at least a portion of an object external to the hypermedia document. The object has associated type information utilized by the browser to identify and locate an executable application external to the hypermedia document.

When an embed text format is parsed by the browser, the executable application is automatically invoked, as a result of the parsing, to execute on the client workstation.

Applicant's Response, at 12 (Oct. 12, 2004) (831 PH Ex. 16 at PH_001_0000785814):

Toye states that when an object or file is selected by the user the system will automatically invoke the application for display in a NoteMail page. Further, Toye teaches that the application launching functionality is similar to opening a file using Macintosh Finder. [Toye at page 40, first full paragraph]. Thus, Toye teaches that automatic invoking is a result of user selection, not parsing as required by claims 1 and 6 of the '906 patent, and that the result of the user's interactive selection is similar to opening a file using Macintosh Finder, where the application launched processes the file in its own window. [Felten II, at paragraph 36].

Applicant's Response, at 13-14 (Oct. 12, 2004) (831 PH Ex. 16 at PH_001_0000785815-16):

This combination would not show automatic invocation of the editor program when the hypermedia document is parsed or enable interactive processing within a portion of the first hypermedia document being displayed in the browser window, as required by claims 1 and 6 of the '906 patent. Instead, the external editor application of Toye would be invoked only if the

user took the additional manual action of selecting the static image by clicking on it, causing interactive processing to be enabled in an external window when the external application was restarted. [Felten II, at paragraphs 48-50].

Felten II Declaration, ¶¶ 33–34 (Oct. 6, 2004) (accompanying Applicants' Response (Oct. 12, 2004)) (831 PH Ex. 13 at PH_001_0000785582):

33. Toye teaches that NoteMail interacts with an external program by first displaying a static snapshot of the external content. If the user clicks on that static snapshot, the external editor application is restarted in a separate window.

* * * * *

(Toye at p. 40, col. 2, first full paragraph) It is clear from this discussion that before the data can be edited, the user must select the displayed data with the mouse and the application must be restarted. Since the user must take specific action to select the data before editing is enabled, the editor is not "automatically invoke[d] ... in order to display said object and enable interactive processing" as required by the '906 claims.

Interview with Examiner St. John Courtenay III, at 26 (Aug. 18, 2005) (831 PH Ex. 17 at PH_001_0000785892):

Toye: No Automatic Invocation for Interactive Control

- Although Toye uses the language "automatically invoked," Toye teaches that this action occurs **only as a consequence of the user's active selection**
- Therefore, **Toye does not teach automatic invocation** of an external application to display an object and enable interactive processing of that object within a display area created within a hypermedia document

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 8–9 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785916-17):

> The instant claimed '906 "executable application" that provides the claimed "interactive processing" is invoked not in response to a user event detected by the browser (as in the case of Raggett I, *supra)*, but rather in response to the browser application parsing an "embed text format" (i.e., an "EMBED" tag, see col. 12, line 60, '906 patent) that is detected within the hypermedia document when the hypermedia document is first loaded by the browser.
>
> 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."

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 11–12 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785919-20):

> The Examiner concurs that automatic invoking, as taught by Toye, is the result of manual user selection with a mouse of a "static snapshot" image that automatically launches the "appropriate application" to edit the data object. This approach appears to be similar to the method employed by conventional file manager programs that implement file type association to invoke the appropriate application when the user clicks on the filename or file icon.

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 17 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785925):

> MediaMosaic does enable interactive control and manipulation of objects embedded in what arguably may be construed to be a "browser-controlled window," BUT ONLY AFTER USER INTERVENTION, such as by making a selection with a mouse.

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 19 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785927):

In contrast, Toye teaches that external data is displayed as a "static snapshot" (i.e., representing a data object) within a NoteMail page that must be selected by a mouse to launch an editor application in a separate window" [see Felten II, at paragraph 47]. Thus, Toye clearly requires user intervention to enable interactive processing.

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 28–29 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785936-37):

Furthermore, Toye teaches that "automatic invoking" of the "appropriate application" is performed by selection, and not by parsing. Toye teaches that notebook data is displayed as a data object or filename that must be selected by a mouse to launch an appropriate application in a separate window" [see Toye page 40, 2nd column, paragraph 2; see also page 36, 2nd column, last paragraph, i.e., " ... ability to construct hyper-documents containing bitmaps, video, and audio"; see also Felten II, at paragraph 47].

Significantly, Toye appears to merely disclose a conventional system for invoking appropriate applications by standard prior art file association techniques, such as invoking the appropriate application based upon the file extension (e.g., when the user clicks and selects a *.doc filename or corresponding file icon and this user action automatically invokes the appropriate word processor). See also Toye: "The functionality is similar to

opening a file using the Macintosh Finder and automatically invoking the appropriate application for processing that file" [p. 40, 2nd column, 2nd paragraph].

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 35 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785943):

The manual selection step required by Toye defeats the purpose of the use of an EMBED tag that is parsed to invoke an executable application, thus teaching away from the hypothetical four-way combination of Mosaic (APA), Berners-Lee, Raggett I and II.

In contrast, the instant '906 claims require the browser (and not the user) to invoke the "executable application" that in turn executes on the client workstation to enable the claimed "interactive processing."

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 37 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785943):

In contrast, the instant '906 claims require the browser (not the user) to invoke the "executable application" that in turn executes on the client workstation to enable the claimed "interactive processing."

iv. Second reexam (90/007,858)

Applicant's Response, at 4 (Sept. 27, 2007) (858 PH Ex. 05 at PH_001_0000787033):

A. The Claimed Invention.

The invention, as recited for example in claims 6, is for use in a system having at least one client workstation and one network server coupled to a network environment.

The claims recite a browser application, executed on the client workstation, that parses a hypermedia document to identify text formats in the document and responds to predetermined text formats to initiate processing specified by the text formats.

The browser displays a portion of a first distributed hypermedia document, received over the network from the network server, in a browser-controlled window. The hypermedia document includes an embed text format, located at a first location in the hypermedia document, that specifies the location of at least a portion of an object external to the hypermedia document. The object has associated type information utilized by the browser to identify and locate an executable application external to the hypermedia document.

When an embed text format is parsed by the browser, the executable application is automatically invoked, as a result of the parsing, to execute on the client workstation.

Declaration of Edward W. Felten, ¶ 25 (858 PH Ex. 06 at PH_001_0000787055):

25. 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. The use of an embed text format was a significant improvement over the prior art Mosaic browser, as it allowed the browser to recognize immediately that an embedded object was present and special processing was needed.

Declaration of Edward W. Felten, ¶¶ 28–30 (858 PH Ex. 06 at PH_001_0000787055-56):

28. Claim 6 includes the limitation "wherein said embed text format is parsed by said browser to automatically invoke said executable application" ('906 Patent at 18:23-25), which requires that the executable application be invoked automatically, that is, without requiring any action such as a mouse click from the user.

29. In Mosaic, as stated above, a helper application was launched via an ordinary hyperlink. All hyperlinks required a mouse click by the user to be activated. To launch a helper application, the user had to make such a mouse click. Therefore, the launching of helper applications was not automatic.

30. Automatic invocation was an important improvement in the claimed '906 technology over the prior art Mosaic browser. Automatic invocation allowed the object to appear immediately when the user visited the enclosing web page, thus helping to make the object appear to be an integral part of the web page.

v. '985 prosecution history (10/217,955)

Applicants' Supplemental Amendment after Non-Final Rejection, at 16-17 (April 11, 2008) (985 PH Ex. 11 at PH_001_0000784583-84):

Claim 19 of the present application recites "automatically invoke the executable application, in response to the identifying of the embed text format, to execute on the client workstation in order to display the object and enable interactive processing of the object while the object is being displayed within a display area created at the first location within the portion of the hypermedia document being displayed in the browser-controlled window".

As determined in Part I of the Confirmation, neither the four-way combination of the patent owner's admitted prior art (APA), Berners-Lee, Raggett I and Raggett II nor Toye, singularly or in combination, fairly teach or suggest automatically invoking an external application to enable interactive processing of an object being displayed within the display area.

The four-way combination was determined to teach displaying a static object. Toye was found to display a "static snapshot" of external content where interactive processing was not automatically invoked but required manual selection by the user.

Accordingly, since the recited automatically invoking to enable interactive processing is not taught or suggested by the cited references a *prima facie* case of obviousness has not been established.

Independent claims 4, 23, 27, 31, 35, 39, 43 and 47 recite similar limitations as claim 19 and are thus allowable for the same reasons. The remaining claims are dependent claims which are allowable for the same reasons as the claims on which they depend.

Notice of Allowability, at 2 (Mar. 20, 2009) (985 PH Ex. 15 at PH_001_0000784733):

The following is an examiner's statement of reasons for allowance: the claims are allowable as the claims contain the subject matter deemed allowable in both Re exam 90/006,831 and Re exam 90/007,838 for the same reasons as set forth in the NIRC of the two Re exams.

d. Cited prior art

Object linking and Embedding OLE 2.01 Design Specification (Sep. 27, 1993). cited by other ., at 70-71(*see, e.g.*, Ex. N at PH_001_0000008561-63, 600-01):

2.5. States and Visualizations of Objects

With editing in place, OLE 2 enhances the conceptual structure of a compound document by exposing in a single window the entire hierarchy of containment. In OLE 1 there were two kinds of objects to be considered: the container and the embedded objects it contained. That those objects themselves might also contain embedded objects was of no consequence, since this fact was not relevant until they were opened into their own windows. The container and the open object were always the same, and it was not necessary to think of embedded objects as containers themselves. In OLE 2 this situation changes: the open object becomes the root of a hierarchy of embedded objects, *each* of which may be a container in its own right:

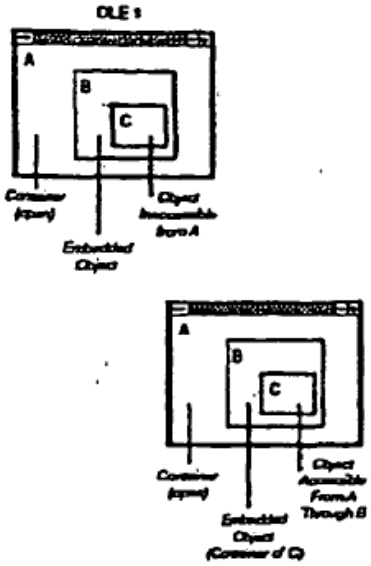


Figure 20. OLE 1 and OLE 2 object hierarchies.

As the user navigates through this hierarchy, OLE objects assume different states and appearances. An OLE object may be *inactive*, *selected*, *active*, or *open*.

2.5.1. Inactive

An object is said to be inactive when it is neither active nor part of a selection. It is displayed in its *presentation* form which is (usually) conveyed through its cached meta-file description. It may be desirable to know whether an object is embedded or linked at a glance without having to interact with it. Container applications should provide a "show objects" option that places a single pixel wide black *solid* border around the extent of an embedded object and a *dotted* border (Figure 22) around linked objects. If the container application cannot guarantee that a linked object is up to date with its source (because an automatic update was unsuccessful or the link is manual), the dotted border should appear in the Windows "disabled text" color (typically gray), suggesting the link is likely out of date. Only the container document's first level objects should be bordered. For example, if in Figure 7 "show objects" were chosen from the outer-level container (Microsoft Word), then only the worksheet object would be bordered, not the graph object nested inside of it. These borders should be clearly distinct from the visualizations for the other states, and are useful to distinguish embedded from linked objects.

An inactive object may be selected by single clicking anywhere within its extent, or it may be double-clicked which will perform its primary verb. The state diagram in Figure 27 shows that double-clicking will activate or open the object for editing, the usual primary verbs.

U.S. Compact Disc vs. LP Sales ($)

	1983	1987	1991
CD's	6,345K	18,652K	32,657K
LP's	31,539K	25,571K	17,429K
Total	37,883K	45,223K	50,086K

Figure 21. Unmodified presentation

2.5.2. Selected

The embedded object is selected when it is either single clicked or included in a multiple selection. It is selected (and deselected) and rendered according to the normal highlight rules of its container (see Figure 3), and responds to appropriate commands as any selected object of the container would. When the object is the singly selected, object-specific operations may be performed on it; the container retrieves these verbs from the system registry. When the object is selected the container may supply handles (for resizing, etc.) which affect the object as a unit with respect to the container. It is recommended that resizing an OLE object while it is selected results in a scaling operation, since there is no mechanism by which the container can communicate a cropping area that would be honored by the object when it is active.

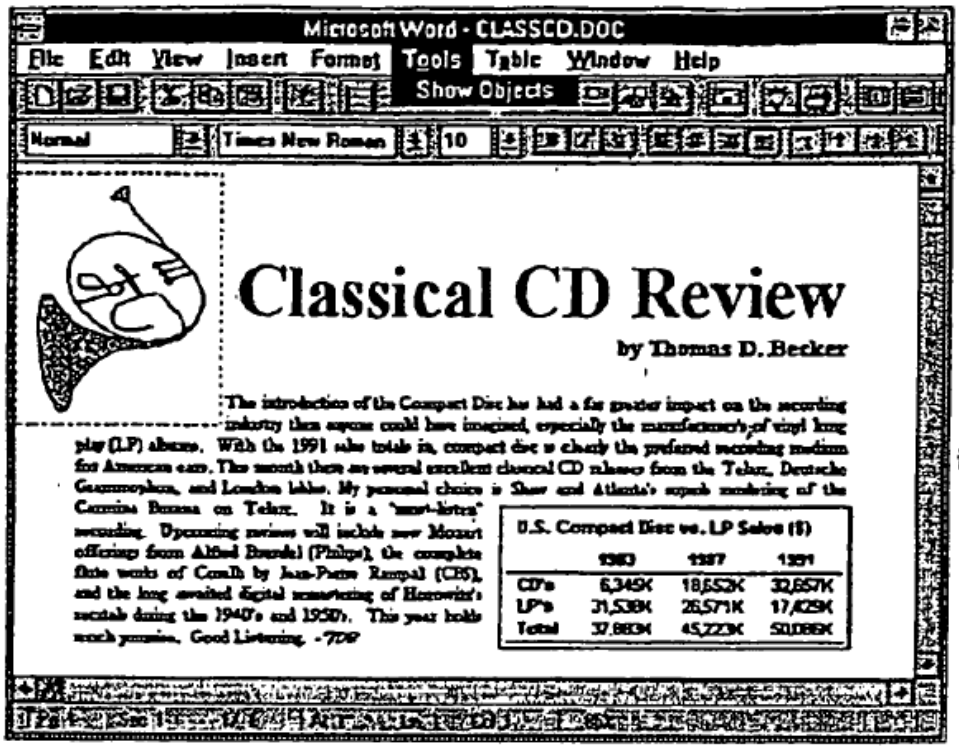


Figure 22. The Show Objects option.

When an object is singly selected, any of its registered verbs may be applied. "Edit" and "Open" will activate and open the object respectively, but other verbs (e.g. "Play") might perform and then leave the object selected. Any number of single clicks will simply reselect the object, while clicking outside will deselect the object. Verbs whose appropriateness depends on the state of the object should ideally enable and disable appropriately. For example, a media object which has Play and Rewind as verbs should disable Rewind when the object is already at the beginning. Similarly if an open object has two verbs Edit (for in-place editing) and Open (for opened editing), Edit should be disabled since the object cannot directly achieve the in-place active state without first closing.

2.5.3. Active

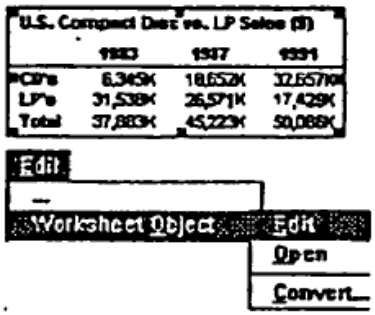


Figure 23. Selected OLE object.

OLE 2 objects may enter an *active* state in which the user may interact with the object's contents in-place, reusing its container document's window for its application's menus and interface controls. The user can make an object active either by performing its appropriate verb ("Edit"), double-clicking it (since for many objects the primary verb will be "Edit"), or selecting the object and pressing Enter. If Enter already has reserved meaning within the container, then Alt + Enter is recommended. When an object becomes active its application's menus and interface controls are grafted into the document's window and apply over the extent of the active object. Frame adornments appear outside the extent of the object, and thus may cover neighboring material in the document temporarily. Row/column headers (as pictured below), handles, or scrollbars are examples of frame adornments an object may wish to present. Scrollbars would allow the scrolling of a large spreadsheet within the object's viewport for example. The object and its frame adornments are surrounded by a black diagonal hatch border as an indication of the active state and to suggest the area of focus. The hatch is always black; it does not change color as focus changes between windows. There is only one object activated at a time; there is no attempt to activate all objects that use the same application as a set.[3] The hatch pattern is comprised of right-ascending diagonal lines as

[3] If for instance all Paintbrush Bitmaps were activated together, commands such as "Select All" or "Clear All" are ambiguous. The user would not know which object(s) would be affected by such document-scoped commands.

illustrated below. The object takes on the appearance which is best suited for its own editing: frame adornments may appear, table gridlines, handles, and other editing aids. The hatch border is considered to be part of the object's territory so it is the object's cursor that appears when the mouse hovers above the border. Clicking in the hatch pattern (and not on resize handles) should be re-interpreted by the object as clicking just inside the edge of the border. The hatch area is effectively a click slop zone that prevents inadvertent deactivations and makes it easier to select the contents of the object which lies right along its edge. The examples below show the border around an active worksheet (notice the border surrounds frame adornments).

Should the container may set at a view-scale (zoom ratio) which the object cannot match in order to perform in-place activation, the object should instead open into a separate window; if the object does not support an open mode, then it should not respond to the verb but issue an appropriate error message (in a dialog) indicating why.

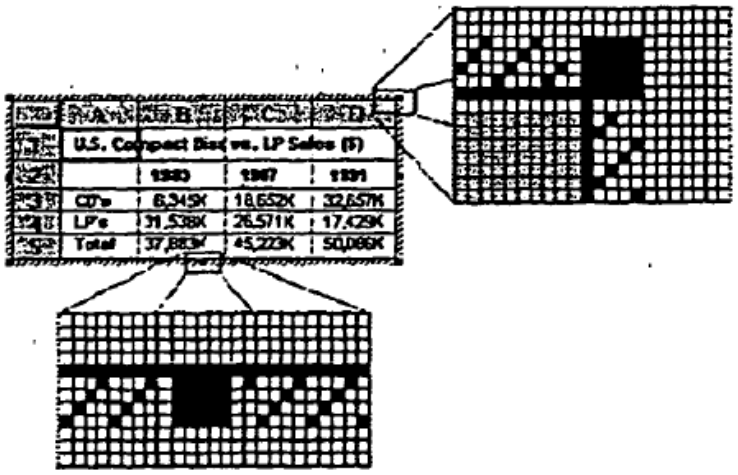


Figure 24. Hatch border around in-place activated objects.

Note that at any given instant there is at most one object that is active in-place per container. A single click in the container area, or a double click on a new OLE object (which may be nested in the currently active object) deactivates the current object and gives the focus to the new object.

An active object may be deactivated by clicking outside its extent in the container document or by pressing the Escape key. If an object uses the Escape key at all times, it is recommended that Shift+Escape is used to deactivate, after which it becomes the selected element of its container.

Edits made to an active object immediately and automatically become apart of the container document, just like edits to native data. Consequently, there is no "update changes?" prompt when an in-place active object deactivates. Of course, changes to the entire document, embedded or otherwise, can be abandoned by declining to save the file to disk. As we shall later see, in-place active objects participate in the Undo stack of the window in which they are activated.

Those objects which support resizing while in-place active should include square resize handles within the active hatch pattern. The solid black handles should be of the same width as the hatch pattern and have a single white pixel separation from the diagonal lines. It is recommended that in-place resizing exposes more or less of the object's content (adding or removing rows/columns in the case of this worksheet). In-place resizing should be seen as adjusting the viewport rather than scaling the object's appearance. Certain objects however may default to in-place scaling if cropping is not meaningful.

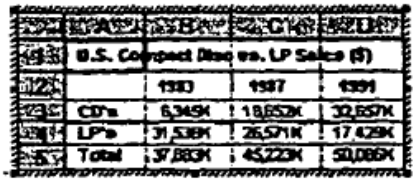


Figure 25. In-place resizing.

* * * * *

3.2.4. States of Embedded Objects

Figure 67 illustrates the main states of objects as they relate to the execution of servers when the objects are instantiated by containers, etc. In addition to the object states, there will be relevant states of server applications (e.g., dormant: invisible, not object, but server task is running so that getting into the Editing state is quick).

There is no requirement that all objects in a container be loaded at once; the container may choose to optimize the loading process. Similarly, containers may defer making the object running, in order to conserve memory, or may pre-activate server tasks for fast startup.

Passive: Object is on disk.

Loaded: Object structure is in client memory. The object may use the cached picture for rendering. The native object data will not normally be in memory.

Running: The server .EXE (or handler) for the object is running, with the object data available to it. The object can do full negotiation and rendering, etc. Clients who have links to this object may bind to it or receive notification of availability.

From the server's perspective, it is sometimes (though less often) useful to distinguish variations in the running state:

Ready: The server or handler has created UI resources, but does not have the focus. The server's tools and menu may be visible but inactive if the focus is in a different window, or invisible if the focus is elsewhere in the container's window hierarchy.

Active: The server or handler has the menus and is getting input.

Open: The server or handler has opened the object in a pop-up window, which may look like its normal frame window.

Executing: The server is asynchronously executing some command invoked by the client.

Note that there is no direct correspondence between the UI states described earlier and the states illustrated above. A UI Inactive object, for example, could be Passive, Loaded, Running, Ready, or Executing. The states where the two classifications coincide are indicated by identical names: objects that are *Active* or *Open* in the UI sense, are also Active or Open in the sense defined here.

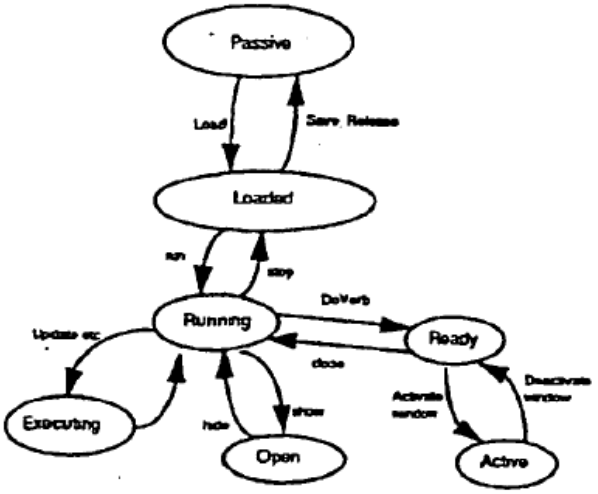
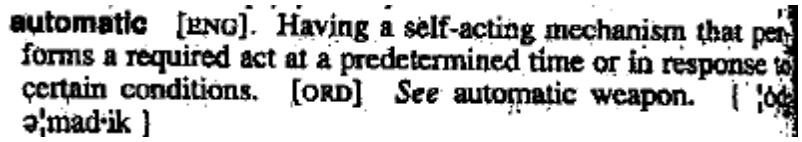


Figure 67. Sketch of object states transitions

2. Defendants' extrinsic evidence

a. Dictionaries

McGraw-Hill Dictionary of Scientific and Technical Terms 158 (5th ed. 1994) ("automatic") [Ex. U at PA-0000333403]:



automatic [ENG]. Having a self-acting mechanism that performs a required act at a predetermined time or in response to certain conditions. [ORD] *See* automatic weapon. [ˌȯd·əˈmad·ik]

b. Testimony

Doyle cross, Trial Tr., *Eolas Techs Inc. v. Microsoft Corp.*, No. 99-C-626 (N.D. Ill. 2003), at 459:12–460:1 (July 10, 2003) [Ex. GG (EOLASTX-E-0000000644)]:

> Q. The claims require that the executable application be ***automatically invoked***, isn't that right?
>
> A. That's correct.
>
> Q. And what does "invoke" mean in the world of computers?
>
> A. Well, as I mentioned on direct, when the browser sees the embed tag, it invokes the application without the user having to do anything.
>
> Q. So what that means is that the executable application starts up ***without a mouse click***, right?
>
> A. That's correct.
>
> Q. When the Web page is displayed.
>
> A. Correct.
>
> Q. So if a ***mouse click*** were required first, it would be outside the scope of this patent.
>
> A. Correct.

Michael Doyle Dep., *Eolas Techs Inc. v. Microsoft Corp.*, No. 99-C-626 (N.D. Ill. February 28-March 1, 2000), at 109:10–110:10 [Ex. FF (EOLASTX-E-0000000180)]:

> Q. So the word "***automatically***" in claim 6, you would define as happening in response to the loading of the Web page?
>
> A. I would use it in the sense that it's used in the invention where it happens ***as a result of the parser identifying the embed text format*** and going through the other operations that show the elements of the invention.

Q. I'm still not clear that that was an answer to the question. Let me make it simple. What does the word "***automatically***" mean as used in claim 6 at column 18, line 24?

MS. CONLIN: Objection, asked and answered.

THE WITNESS: Again, it shows that the browser renders or ***automatically invokes*** the executable application in response to the elements that are described above in claim 6.

BY MR. PETERSEN:

Q. In response to the parsing of the Web document?

A. The ***parsing of the Web*** document, the ***browser using type information to identify and locate*** executable application and the browser -- and the text format's "***parsed by said brower to automatically invoke*** said executable application."

Michael Doyle Dep., *Eolas Techs Inc. v. Microsoft Corp.*, No. 99-C-626 (N.D. Ill. February 28-March 1, 2000), at 345:12–346:10 [Ex. FF (EOLASTX-E-0000000181)]:

Q Take a look, if you would, again at Exhibit 78, the attachment to your invention disclosure form, please. And I'd like you to read the first paragraph of the first page of that document, and then I'll ask you some questions.

A Yes, I see that paragraph.

Q Is that an accurate description of ***your invention*** at the time that the invention disclosure was signed in April of 1994?

A I wouldn't say it was a completely accurate description, no.

Q What's wrong with it?

MS. CONLIN: Objection, misstates his testimony.

THE WITNESS: There is a sentence that says "when a user browsing the WWW selects such a link," and so on. That does not appear to be accurate.

BY MR. PETERSEN:

Q Why not?

A Because ***a user didn't have to do any selection of links for a Web page to cause the execution of the external application***.

Michael Doyle Dep., *Eolas Techs Inc. v. Microsoft Corp.*, No. 99-C-626 (N.D. Ill. February 28-March 1, 2000), at 570:21–571:9 [Ex. FF (EOLASTX-E-0000000182)]:

Q. What does it mean to ***automatically invoke*** an executable application?

A. To invoke ***without requiring user interaction***.

Q. When it says that the embed text format is parsed by the browser to ***automatically invoke*** the external application -- executable application, does that mean that the browser invokes the application?

A. Well, as shown in the specification, the external application is invoked ***as a result of the parsing*** of the embed text format as described in its entirety in the claim.

Cheong Ang Dep., *Eolas Techs Inc. v. Microsoft Corp.*, No. C-99-0212 (N.D. Ill. January 21-22, 2000), at 232:25–233:9 [Ex. EE (EOLASTX-E-0000000177)]:

Q. At line 24 of Column 18 --

A. Okay.

Q. -- there's the phrase "to ***automatically invoke***," I'm going to ask you what's your understanding of that is, but if you'd first read whatever portion of the claim around that that you feel is necessary and let me know when you have.

A. ***"Automatically invoke" means "invoke without human intervention or user intervention."***

David Martin Dep., *Eolas Techs Inc. v. Microsoft Corp.*, No. C-99-0212 (N.D. Ill. January 20-21, 2000), at 193:9–194:1 [Ex. DD (EOLASTX-E-0000000174)]:

Q. Mr. Martin, looking again at Claim 6 of the patent which is Exhibit 15 in front of you, in Column 18, about line 24 it says that the "embed text format is parsed by said browser to automatically invoke the application," do you see that?

A. Yes.

Q. What does it mean to "***automatically invoke***" in that sentence?

A. Generically it means to respond to the information contained in the hypermedia document, to start -- if I can refer back -- to start the program code for the application external to the first hypermedia document.

Q. But the word "***automatically***," what does that add to it?

A. ***It means that it's done in the course of parsing the hypermedia document***.

C. "text formats" and "embed text format"

Claim Term(s)	Defendants' Proposed Construction	Eolas's Proposed Construction
text formats	tags or symbols that specify document formatting	text that initiates processing
embed text format	[see proposed constructions below for "embed text format . . . first location"]	text format for embedding an object

1. Defendants' intrinsic evidence

a. Claims

In the following chart, the term(s) for construction appear in all the claims marked with an "x." The letter "m" indicates a method claim, and "¶6" indicates a claim that the Defendants contend includes limitations subject to § 112, ¶ 6.

	'906 patent						'985 patent								
	m 1	m 4	m 5	¶6 6	¶6 9	¶6 10	m 1	¶6 16	m ¶6 20	m ¶6 24	¶6 28	m ¶6 32	m 36	m ¶6 40	m 44
text format	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
embed text format	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

b. Specification (all cites to '906 patent)

1:53–:60 & 2:23–:28 (Background of the invention): Other Internet standards are the HyperText Transmission Protocol ("HTTP") that allows hypertext documents to be exchanged freely among any computers connected to the Internet and ***HyperText Markup Language ("HTML") that defines the way in which hypertext documents designate links to information***. See, e.g., Berners-Lee, T. J., "The world-wide web," Computer Networks and ISDN Systems 25 (1992). … ***A hypermedia document is similar to a hypertext document***, except that the user is able 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.

2:43–:48 (Background of the invention): ***The mechanism for specifying and locating a linked object*** such as hypermedia document 14 ***is an HTML "element" that includes an object address in the format of a Uniform Resource Locator (URL)***.

5:24–:38 (Background of the invention): The Internet is said to provide an "open distributed hypermedia system." It is an "open" system since Internet 100 implements a standard protocol that each of the connecting computer systems, 106, 130, 120, 132 and 134 must implement (TCP/IP). ***It is a "hypermedia" system because it is able to handle hypermedia documents*** as described above ***via standards such as*** the HTTP and ***HTML hypertext transmission and mark up standards***, respectively.

9:24–:40 (Detailed Description of a Preferred Embodiment): "Once hypermedia document 212 has been loaded into client computer 200, browser client 208 parses hypermedia document 212. In parsing hypermedia document 212, browser client 208 ***detects links to data objects as discussed***

above in the Background of the Invention section. In FIG. 5, ***hypermedia document 212 includes an embedded program 30 link at 214. Embedded program link 214 identifies application client 212 as an application to invoke***. In this present example, the application, namely, application client 210, resides on the same computer as the browser client 208 that the user is executing to view the hypermedia document. Embedded program link 214 may include additional information, such as parameters, that tell application client 210 how to proceed. For example, ***embedded program link 214 may include a specification as to a data object that application client 210 is to retrieve and process***."

9:50–:58 (Detailed description of a preferred embodiment): This means that application client 210 can make requests over network 206 for data objects, such as multidimensional image objects. For example, application client 210 may request an object, such as object 1 at 216, located in server computer 204. Application client 210 may make the request by any suitable means. Assuming network 206 is the Internet, such a request would typically be made by using HTTP in response to a ***HTML-style link definition for embedded program link 214***.

12:54-13:36 (Detailed Description of a Preferred Embodiment):

Table II, below, shows an example of an HTML tag format used by the present invention to embed a link to an application program within a hypermedia document.

TABLE II

```
<EMBED
    TYPE = "type"
    HREF = "href"
    WIDTH = width
    HEIGHT = height
>
```

As shown in Table II, the EMBED tag includes TYPE, HREF, WIDTH and HEIGHT elements. The 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. In the case where TYPE is "application/x-vis" this means that the object at the URL address is a three dimensional image object since the program "x-vis" is a data visualization tool designed to operate on three dimensional image objects. However, any manner of application program may be specified by the ***TYPE element so that other types of applications, such as a spreadsheet program, database program, word processor, etc. may be used with the present invention. Accordingly, the object reference by the HREF element would be, respectively, a spreadsheet object, database object, word processor document object, etc***.

On the other hand, TYPE values such as "video/mpeg", "image/gif', "video/x-sgi-movie", etc. describe the type of data that HREF specifies. This is useful where an external application program, such as a video player, needs to know what format the data is in, or where the browser client needs to determine which application to launch based on the data format. Thus, the TYPE value can specify either an application program or a data type. Other TYPE values are possible. HREF specifies a URL address as discussed above for a data object. Where TYPE is "application/x-vis" the URL address specifies a multi-dimensional image object. Where TYPE is "video/mpeg" the URL address specifies a video object.

WIDTH and HEIGHT elements specify the width and height dimensions, respectively, of a Distributed Hypermedia Object Embedding (DHOE) window to display an external application object such as the three dimensional image object or video object discussed above.

13:37–:50 & fig. 7A (Detailed description of a preferred embodiment): FIG. 7A is a flowchart describing some of the functionality within the HTMLparse.c file of routines. ***The routines in HTMLparse.c perform the task of parsing a hypermedia document and detecting the EMBED tag***. In a preferred embodiment, the enhancements to include the ***EMBED tag*** are made to an HTML library included in public domain NCSA Mosaic version 2.4. Note that much of the source code in is pre-existing NCSA Mosaic code. Only those portions of the source code that relate to the new functionality discussed in this specification should be considered as part of the invention. The new functionality is identifiable as being set off from the main body of source code by conditional compilation macros such as "#ifdef . . . #endif" as will be readily apparent to one of skill in the art.

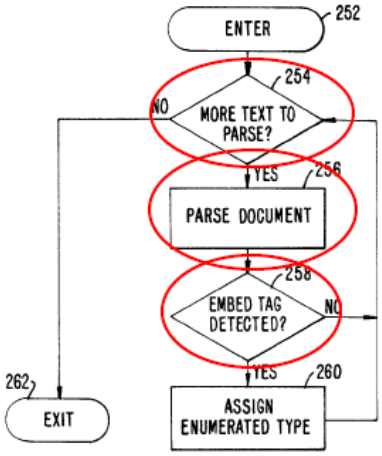


FIG. 7A.

14:18-23 (Detailed Description of a Preferred Embodiment): Steps 254, 256 and 258 represent a loop where the document is parsed or scanned ***for HTML tags or other symbols***. While the file HTMLparse.c includes routines to handle all possible tags and symbols that may be encountered, FIG. 7A, for simplicity, only illustrates the handling of EMBED tags.

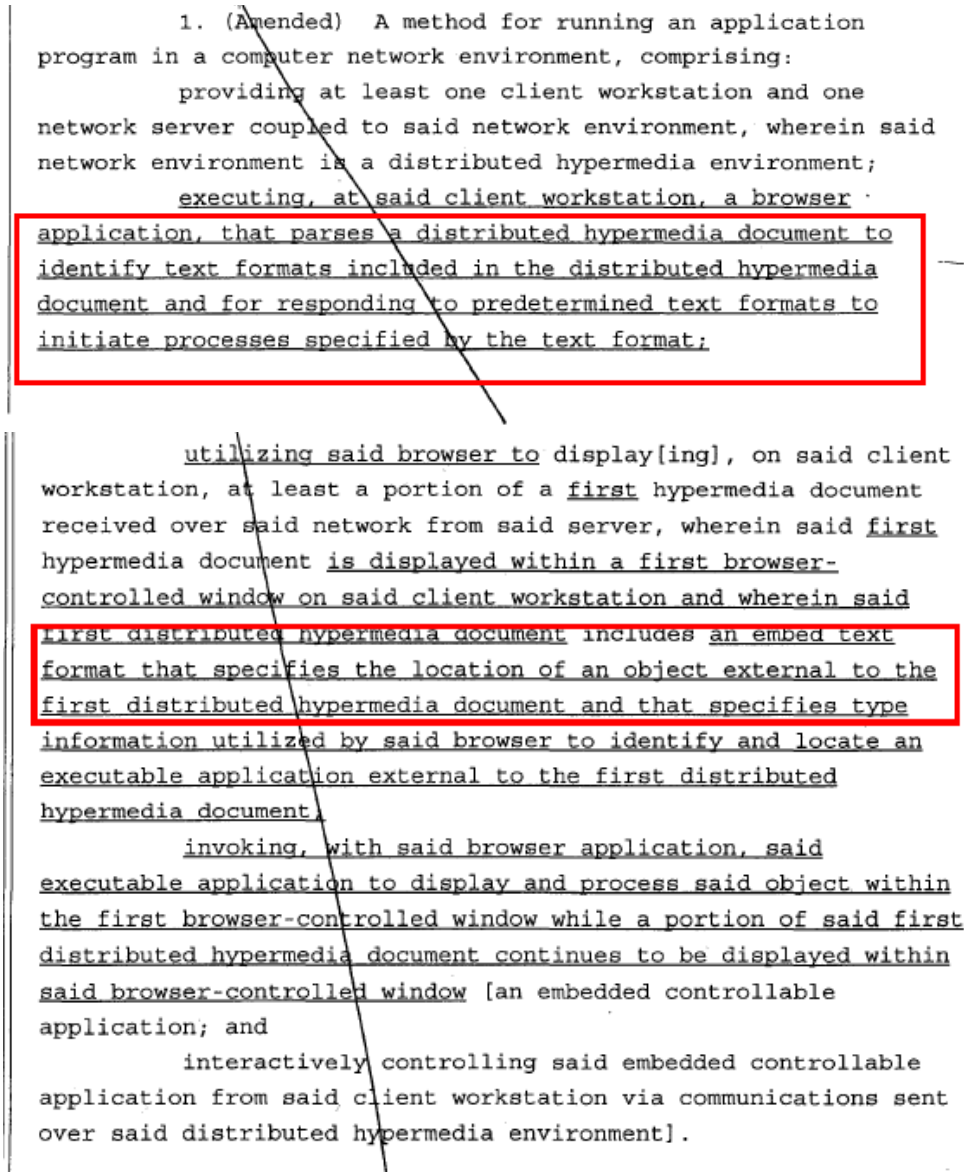
14:24-26 (Detailed Description of a Preferred Embodiment): Assuming there is more text to parse, execution proceeds to step 256 where routines in HTMLparse.c obtain the ***next item (e.g., word, tag or symbol)*** from the document.

14:64–:67 (Detailed description of a preferred embodiment): FIG. 8A is a flowchart for routine HTMLwidget. HTMLwidget creates display data structures and launches an external application program to handle ***the data object specified by the URL in the EMBED tag***.

c. Prosecution history

i. '906 prosecution history (08/324,443)

Applicants' Response, at 1-2 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783879-80):



1. (Amended) 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 distributed hypermedia document to identify text formats included in the distributed hypermedia document and for responding to predetermined text formats to initiate processes specified by the text format;

utilizing said browser to display[ing], on said client workstation, at least a portion of a first hypermedia document received over said network from said server, wherein said first hypermedia document is displayed within a first browser-controlled window on said client workstation and wherein said first distributed hypermedia document includes an embed text format that specifies the location of an object external to the first distributed hypermedia document and that specifies type information utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document,

invoking, with said browser application, said executable application to display and process said object within the first browser-controlled window while a portion of said first distributed hypermedia document continues to be displayed within said browser-controlled window [an embedded controllable application; and

interactively controlling said embedded controllable application from said client workstation via communications sent over said distributed hypermedia environment].

Applicants' Response, at 13–14 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783885-86): "[In t]he present invention, as defined for example in amended claim 1, . . . ***[t]he distributed hypermedia document includes an embed text format*** that specifies type information utilized by the browser to identify and locate an executable application external to the distributed hypermedia document"

Applicants' Response, at 16 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783894):

The claimed combination is fundamentally different from the Mercury Project. In the claimed combination, the external object and executable object are embedded by reference in the HTML document and the object is displayed and processed within the same window where a portion of the original document is displayed. In the Mercury Project information is passed back to the server and a new document is generated and displayed. There is no display and processing the external object within the window in which a portion of the original document is displayed.

Applicants' Response, at 16-17 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783894-95):

The Hansen reference discloses embedding an executable script in a document. The Ness script is a sequence of attribute specifications, i.e., declarations of global variables, global functions, and extend blocks. An extend block associates a set of contained attributes with some named object and has the following syntax:

extend <name>

<attributes>

end extend

where <name> must be a string constant giving the name of the associated object. (Page 25). One attribute is an event specification such a mouse click.

The operation of Ness scripts is illustrated in the "Extended Birthday Card" example at pages 30 and 31. Note that the object named is: extend "visible cake". ***Both the executable script and the object to be manipulated are within the document.*** . . .

There 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.

The system of Ness provides for interaction with an object embedded in a document by executing code embedded in the document. However, there is no teaching or suggestion of the claimed system of utilizing a browser to invoke an external application identified by an original document, being displayed by a browser within a first window, to display and process an external object within the first window.

Applicants' Response, at 17 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783895):

In view of the above, it is believed that the claims are not obvious over the disclosed prior art in view of Hansen. There 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.

Applicants' Response, at 4 (Dec. 29, 1997) (906 PH Ex. 16 at PH_001_0000784134): "The different functions and purposes of Mosaic and Koppolu (OLE) are reflected in the different document structure. In Mosaic, since HTML documents are designed to be platform independent, the document structure is simple ASCII text. ***A browser parses a received document to identify HTML tags which specify various aspects of the document's appearance and links to other documents***. In Koppolu, a container application creates a complex file structure which is utilized to render a document. There is no text parsing in Koppolu to render the compound document."

Applicants' Response, at 2 (Jan. 8, 1997) (906 PH Ex. 06 at PH_001_0000783957):

In claim 1, the distributed hypermedia document includes an embed text format that specifies the location of an object external to the distributed hypermedia document and that specifies type information utilized by the browser to identify and locate an executable application external to the distributed hypermedia document. The browser invokes the executable

Office Action, at 3 (Jan. 24, 1997) (906 PH Ex. 07 at PH_001_0000783999):

Wynne did not specifically disclose the document having text formats. It is not clear whether the HyperNet's hypermedia documents disclosed by Wynne uses text formats. However it is well known in the art at the time of the invention to form hypermedia documents using text formats (e.g. SGML, HTML, etc.). Hansen teaches to use text because it is machine independent so the result is more portable [p.257 4th paragraph]. Hence, one of ordinary skill in the art would have been motivated to use text formats to form hypermedia document.

Examiner Interview Summary Record, at 1 (Feb. 26, 1997) (906 PH Ex. 09 at PH_001_0000784011):

Description of the general nature of what was agreed to if an agreement was reached, or any other comments:

1) HyperNet is a compiled system, 2)Tag in document to activate external program (delayed binding),

3) display and process by the external application within Browser's controlled window.

Applicant's argument is persuasive to overcome the Hypernet ref. The claims are distinguished over the prior art of record.

Applicants' Response, at 18 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784046):

Next, some sort of data interchange interface would have to be constructed between Mosaic and the Khoyi virtual machine to allow Khoyi's "packs" to create data structures that could be transferred to Mosaic, and for messages created by Mosaic to be transferred to Khoyi.

Mosaic would then have to be modified to allow data object components of the document to be "linked" to Khoyi's applications which can process the data. These links would be defined by an external link table, and the linking relationship would not be affected by the text of the document.

These links would be distinguished from the HTML anchor links defined in the hypermedia document, which would require incorporating two incompatible linking systems to be maintained by the system. Mosaic teaches that a major advantage of the HTML document format is that all links should be defined by the document text. This teaches away from combining the two systems in the proposed way, since the result would be awkward, overly complex and difficult to maintain.

Amendment B, at 19 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784047):

The two linking systems could not be combined into one, due to the architecture of Khoyi's object system. Since Khoyi does not represent links within the document itself, but rather uses a link table which is external to the document, some sort of mechanism would have to be created to allow such links to be fully defined by the document text itself. This modification would render Khoyi's object system inoperable, since Khoyi's entire application architecture depends upon the link tables being the source of all link definitions, and being accessible to all of the various programs that may have a need to operate on a given data object.

Furthermore, since a document in the Khoyi system does not allow the document author to explicitly define or control the definition of the link's internal details, the document itself cannot specify such details as the precise location of a data file on a remote network disk drive. The Web, on the other hand,

Declaration of Michael D. Doyle, at 10 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784071):

identifier, which is referenced in a document, for example, as a "link marker." The actual definition of the link referenced by any particular link marker is located in an operating system data structure called a "link table." The document itself does not allow the document author to explicitly define or control the definition of the link's internal details, such as the precise location of a data file on a disk drive. The Web, on the other hand, employs a uniform resourse locator (URL) construct to manage both link definition and object localization on networked systems, from within the Web document, under the precise control of the Web document author. It appears that the URL mechanism would be incompatible with the linking mechanism requirements imposed by the Khoyi operating system. Since the HTML-based mechanism for linking and object management is one of the major requirements for a successful Web browser, such an incompatibility would likely render the resulting system useless for its intended purpose.

Office Action, at 4 (Aug. 25, 1997) (906 PH Ex. 12 at PH_001_0000784094):

HTML is a text tag structure document encoding. It is apparent the prior art as modified would have had a text tag for indicating links to an in-place interactive object.

Declaration of Michael D. Doyle, at 2-3 (Oct. 29, 1997) (accompanying Applicants' Remarks (Oct. 31, 1997)) (906 PH Ex. 15 at PH_001_0000784129):

2. The subject matter claimed in the above patent application was reduced to practice in this country prior to April 15, 1994, the filing date of the parent of the Koppolu reference cited by the examiner.

3. The reduction to practice of the claimed invention is evidenced by ATTACHMENTS A and B. ATTACHMENT A is a copy of a paper entitled "Integrated Control of Distributed Volume Visualization Through the World-Wide-Web", by Ang, Martin, and Doyle. This paper was submitted for publication prior to April 15, 1994. 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.

4. As stated in ATTACHMENT A, at page 5, paragraph 3.2, Mosaic (the browser) interprets the HTML <EMBED> tag included in a document to create a drawing area widget in a document presentation and creates a shared window system buffer to receive visualization results. In addition, when the browser parses an <EMBED> tag in the document, the browser automatically launches the external application specifying the location of the visual object to render and identify the shared image buffer. The format and operation of an EMBED tag for 3D image data is described at paragraph 3.1.

5. As stated in ATTACHMENT B, starting at the bottom of page 2, interface and control software had been developed that allows the embedding of a visualization application within a Mosaic document. As is apparent from the photographs, the object is displayed and processed within the browser-controlled window. The visualization application is external to the hypermedia document displayed by the browser. Automatic launching of the external application when an HTML document is opened by the browser is depicted in the video.

Declaration of Michael D. Doyle, at Attachment A, pp. 4-5 (Oct. 29, 1997) (accompanying Applicants' Remarks (Oct. 31, 1997)) (906 PH Ex. 13 at PH_001_0000784108–9):

We have enhanced the Mosaic W3 browser to support both a three-dimensional data object and communication with VIS as a cooperating application (figure 2). Mosaic provides the user with the ability to locate and browse information available from a wide variety of sources including FTP, WAIS, and Gopher. HTTP servers respond to requests from clients, e.g. Mosaic, and transfer hypertext documents. Those documents may contain text and images as intrinsic elements and may also contain external links to any arbitrary data object (e.g. audio, video, etc...). Mosaic may also communicate with other Internet servers, e.g FTP, either directly – translating request results into HTML on demand – or via a gateway that provides translation services. As a W3 client, Mosaic communicates with the server(s) of interest in response to user actions (e.g. selecting a hyperlink), initiating a connection and requesting the document specified by the URL. The server delivers the file specified in the URL, which may be a HTML document or a variety of multimedia data files (for example, images, audio files, and MPEG movies) and Mosaic uses the predefined SGML DTD for HTML to parse and present the information. Data types not directly supported b Mosaic are displayed via user-specifiable external applications and we have extended that paradigm to both include three-dimensional volume data as well as to integrate the external application more completely with Mosaic.

3.1 Mosaic 3D Image support

We have extended the HTML DTD to support three dimensional data via the introduction of a new SGML element: IMG3D. This element provides information to the presentation system (i.e. Mosaic) about the content that is referenced in the document. The IMG3D element is defined in the HTML DTD as follows:

```
<!ELEMENT    IMG3D EMPTY>
<!ATTLIST    IMG3D VOLUME CDATA #REQUIRED
             WIDTH NUMBER #REQUIRED
             HEIGHT NUMBER #REQUIRED
             IMAGE CDATA #IMPLIED>
```

which is translated as "SGML document instance element tag IMG3D containing no content; three required attributes: volume data set, window width and height; and an optional image". In a HTML document, a 3D image element would be represented as:

```
<IMG3D       VOLUME="http://www.library.ucsf.edu/.../data/Embryo.hdf"
             WIDTH=400
             HEIGHT=400
             IMAGE="http://www.library.ucsf.edu/.../images/Embryo.gif">
```

which may be interpreted as "create a 3D visualization window of width 400 pixels, height 400 pixels, and visualize the data embryo.hdf located at the HTTP server site www.library.ucsf.edu".

Applicants' Response, at 17 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784147):

In order to insure cross-platform uniformity of document appearance, the document was defined through the use of ASCII text, where specific text formats, otherwise known as "tags," would be used within the document text to specify various aspects of the document's appearance and linkages to other documents or related data. Each browser, therefore, incorporated a parser which would distinguish the formatting tags from the document's narrative text, classify those tags into pre-defined categories, break each tag into its basic components, and then invoke appropriate browser subroutines to respond appropriately to the meanings of the tag components. Although the browser subroutines were built from machine-specific native code, this text tag mechanism allowed the design of a variety of browsers for various computing platforms that could respond in similar ways to similar types of text tags, and therefore result in similar-appearing documents on dissimilar computers. A binary document data format was avoided in order to promote cross-platform compatibility, due to the variation in binary data handling methodologies on various different operating systems, and to simplify the requirements for document creation tools. All that a Web document author needs in order to create a Web document file is a simple ASCII text editor, which is a pre-existing application in all commonly-found operating system packages.

ii. First reexam (90/006,831)

Applicants' Response, at 3 (May 11, 2004) (831 PH Ex. 07 at PH_001_0000785361): "The claims recite a browser application, executed on the client workstation, that parse's a hypermedia document to ***identify text formats in the document and responds to predetermined text formats to initiate processing specified by the text formats***."

Applicants' Response, at 3 (May 11, 2004) (831 PH Ex. 07 at PH_001_0000785361):

> "The ***hypermedia document includes an embed text format***, located at a first location in the hypermedia document, that specifies the location of at least a portion of an object external to the hypermedia document. The object has associated type information utilized by the browser to identify and locate an executable application external to the hypermedia document.
>
> When an embed text format is parsed by the browser, the executable application is automatically invoked, as a result of the parsing, to execute on the client workstation.
>
> When the automatically invoked application executes on the client workstation, ***the object is displayed within a display window created within the portion of the hypermedia document being displayed*** and interactive processing of said object is enabled."

Declaration of Edward W. Felten, at ¶¶ 18–24 (May 7, 2004) (accompanying Applicants' Response (May 11, 2004)) (831 PH Ex. 10 at PH_001_0000785440-41):

> **B. The '906 Patent**
>
> 18. The claims of the '906 Patent describe a technology that allows web page authors to include, within the boundaries of a web page, interactive objects. This is done (briefly stated) by including in the web page's HTML text an embed text format, that provides information about where to get the object's data, along with information to identify and locate an executable application that will be invoked on the client computer to display the data and to provide interactivity with it, and by providing a web browser that knows how to parse the HTML to extract the embed text format, how to use type information to identify and locate the executable application, how to invoke the executable application, to execute on the client computer, and how to interface to the executable application so as to allow the user to interact with it within the boundaries of the browser window.

C. Prior Art Browsers

19. The Office Action cites the applicants' admitted prior art. I have reviewed all prior art references referenced in the '906 Patent's file history. It appears that the Office Action's discussion of this prior art focuses on the Mosaic browser, which was the most advanced prior art browser.

20. Mosaic, and other prior art browsers, executed on a client computer, and operated by downloading copies of web pages (and other files, such as embedded static images) over a network from web servers. After downloading a copy of a file, Mosaic would sometimes keep a copy of that file in a local cache, on the user's client computer. Caching allowed the file to be referenced more quickly if it was needed again later.

21. After downloading a file, Mosaic would parse that file (i.e., analyze its structure) to determine how the file should be displayed on the screen. Mosaic would then paint the contents of the file into a browser window.

22. When Mosaic, or another prior art browser, was used to view web pages, several steps stood between the author of the web page and the user who was viewing it. First, the file would be copied, at least once and perhaps more times, while in transit between the web server and the user's browser. Second, the file would be written in one format (typically, HTML) but displayed in another form, by rendering the HTML into a visual representation that would actually be presented to the user.

23. Because these steps stood between the author and the user, there was no realistic way for the user to edit the web page on the client workstation. The user did not have access to the version of the page that was distributed – that version lived on the server, and it wouldn't make sense to let an arbitrary user edit the contents of somebody else's web page.

24. In addition, because web pages were written in one format (HTML) and viewed in another (visual representation), it did not make sense to talk about editing and viewing a document in the same window. Web page authors would typically work with two separate windows open, one (a browser) to see what the visual representation looked like, and another (an external editor) to actually modify the page's HTML representation. An author would fiddle with the HTML, then click the save button in the editor and the refresh button in the browser to see what the visual representation of the page looked like, then fiddle with the HTML some more, and so on until he was satisfied with the page's appearance.

Applicants' Response, at 4 (Oct. 12, 2004) (831 PH Ex. 16 at PH_001_0000785806): "The browser application then parses the local copy of the HTML document, ***renders the temporary local copy of the HTML document into a Web page, and displays the rendered Web page in a browser-controlled window***. [Felten I, at paragraph 21]. During the rendering step…"

Applicants' Response, at 13 (Oct. 12, 2004) (831 PH Ex. 16 at PH_001_0000785815): "***'Format' data is stored separate from the text portions of the document [Felten II, at paragraph 31]. There is no teaching in NoteMail of using text formats, within the document text, intended to initiate processes specified by those text formats.***"

Applicants' Response, at 18 (Oct. 12, 2004) (831 PH Ex. 16 at PH_001_0000785820): Another important principle of the Web model taught by the Mosaic, Berners-Lee, Raggett I and II combination is that of referential integrity. In the Web model, the HTML document author can ***specify the specific locations***, contained in "hypertext links," from which the browser will retrieve new HTML documents when users click upon those links. ***These links are easily specified through embed text formats in the document text***. … .

Applicants' Response, at 20 (Oct. 12, 2004) (831 PH Ex. 16 at PH_001_0000785822)):

> A distributed hypermedia system "is a "distributed" system because data objects that are imbedded within a document may be located on many of the computer systems connected to the Internet." ['906 at col. 5, lines 25-38].
>
> The use of the HTML allows the Internet to be an open system where a standard protocol is implemented by each computer connected to the internet. The structure of the document is defined by the author utilizing particular sets of characters that have a universal meaning.

Applicants' Response, at 21 (Oct. 12, 2004)) (831 PH Ex. 16 at PH_001_0000785823): … Thus, Toye is ***not a hypermedia system because, in the admitted prior art,*** Berners-Lee, and Raggett I and II combination, ***links are defined by the author as text formats in the hypermedia document and resolved by the browser application***. The Mosaic, Berners-Lee, Raggett I and II combination teaches the use of a hypermedia document that is a text document where some characters within the text are interpreted as ***mark-up tags specified by the HTML standard***. The ***mark-up "tags" give structure to the document***.

Declaration of Edward W. Felten, at ¶ 18 (May 7, 2004) (accompanying Applicants' Response (May 11, 2004)) (831 PH Ex. 10 at PH_001_0000785440): "The claims of the '906 Patent describe a technology that allows web page authors to include, within the boundaries of a web page, interactive objects. ***This is done (briefly stated) by including in the web page's HTML text an embed text format, that provides information about where to get the object's data, along with information to identify and locate an executable application that will be invoked on the client computer to display the data and to provide interactivity with it, and by providing a web browser that knows how to parse the HTML to extract the embed text format***, how to use type information to identify and locate the executable application, how to invoke the executable application, to execute on the client computer, and how to interface to the executable application so as to allow the user to interact with it within the boundaries of the browser window."

Declaration of Edward W. Felten, at ¶¶ 37–46 (May 7, 2004) (accompanying Applicants' Response (May 11, 2004)) (831 PH Ex. 10 at PH_001_0000785442-44):

"Raggett I proposed a slight extension of [building support for displaying additional formats into the browser itself], in which, rather than receiving an image, the browser receives information in some foreign format, and then uses an external program to render that information into an image, which the browser displays within the web page. . . .

This extension is described in the following paragraph, which is also cited in the Office Action:

> The EMBED tag provides a simple form of object level embedding. This is very convenient for mathematical equations and simple drawings. It allows authors to continue to use familiar standards, such as TeX and eqn. Images and complex drawings are better specified using the FIG or IMG elements. The type attribute specifies a registered MIME content type and is used by the browser to identify the appropriate shared library or external filter to use to render the ,embedded data, e.g. by returning a pixmap. It should be possible to add support for new formats without having to change the browser's code, e.g. through using a common calling mechanism and name binding scheme. Sophisticated browsers can link to external editors for creating or revising embedded data. Arbitrary 8-bitdata is allowed, but &, < . and> must be replaced by their SGML entity definitions. For example <embed type="application/eqn">2 pi int sin (omega t) dt</embed> gives [image of equation appears here].
>
> (Raggett I at p. 6)

This paragraph teaches a method for displaying new types of static information within a Web page. The teaching of the use of static information is evident for several reasons.

First, the use of static information is consistent with the teaching of the remainder of Raggett I and with the teaching of Berners-Lee that preceded it.

Second, Raggett I motivates its proposed embed tag by referring to two types of data that one might want to display: "mathematical equations and simple drawings". These are types of data that one would want to display statically.

Third, Raggett I says that Raggett's proposed embed tag "allows authors to continue to use familiar standards, such as TeX and eqn." (italics in ' original). These are well-known formats for describing the display of static data. TeX is used to specify the typesetting of textual documents; it is still widely used to format scientific publications. Eqn is used to specify the typesetting of mathematical equations. The TeX format is conventionally used with a program called "tex" or "latex" that produces as output a static document. The eqn format is conventionally used with a program called "eqn" that produces as output a static image or description of an equation. (For information on TeX, see Donald E. Knuth, The TeXbook, Addison-Wesley, '1986

For information on eqn, see Brian W. Kernighan and Lorinda L. Cherry, "A System for Typesetting Mathematics," Communications a/the ACM 18:3, March 1975; attached as Exhibit 8.)

Fourth, Raggett I refers to the invocation of a "shared library or external filter to render the embedded data, e.g. by returning a pixmap". This passage uses several terms of art (in the art of computer science) in ways that teach non-interactivity. "Filter" is a term of art that refers to a type of non-interactive program that translates data from one format to another. "Render" as used by Raggett I is a term of art that refers to the generation of a static image that is to be displayed. "Pixmap" as used by Raggett I is a term of art for a data structure describing an image. "Return"

is a term of art that refers to the information 'produced by a program when that program terminates. A program that has returned something cannot do' anything else; for example it cannot provide interactive processing. The use of these four terms of art further teaches the use of static images.

Fifth, the only specific example of the use of Raggett's proposed embed tag that is given in Raggett I involves the ,use of a non-interactive filter which renders static data and then returns. The example depicts the use of the "eqn" program to translate the description of an equation into a static image.

Sixth, the discussion of the FIG and ISMAP features in Raggett I is inconsistent with the proposition that Raggett's proposed embed tag allowed interaction with an embedded object. In Raggett I, an instance of Raggett's proposed embed tag can be placed within a FIG element:

> Instead of the *src* attribute, you can include an EMBED element immediately following the <fig> tag. This is useful for simple graphs, etc. defined in an external format. (Raggett I at p. 12, emphasis in original)

When the FIG element is used in conjunction with the ISMAP parameter (as described in the "Active areas" section of Raggett I, p. 13), the FIG element's display area becomes an image map: any mouse clicks made by the user within the visual depiction of the embedded data will be interpreted by the browser as pertaining to the image-map feature, and will therefore be intercepted by the browser and sent by the browser to the web server. This section of Raggett I teaches that the browser may intercept mouse clicks within the depiction of the embedded data, thereby contradicting the proposition that the embedded data itself can react to mouse clicks. . . .

[I]f one of ordinary skill in tlw art (at the time) were asked to implement the Raggett feature, he would do so by to starting with the existing code for handling IMG tags, and modifying that code. The existing IMG code was able to paint static images into the body of a page, based on an input file that described the image. This code would be modified to invoke an external program, which would return a static image that would then be pasted into the web page in the same manner as in an IMG tag. Such an implementation would not support interactivity within a web browser window."

Declaration of Edward W. Felten, at ¶ 50 (May 7, 2004) (accompanyind Applicants' Response (May 11, 2004)) (831 PH Ex. 10 at PH_001_0000785445]): "Raggett II is a brief email message, written in response to requests for "equation support," "eqn support," and support for "embedded Postscript in browsers. Equations, eqn data, and embedded postscript are all formats for specifying static data. The requesters ask for support for two rendering programs, eqn and ghostscript, both of which produce static images as output."

Notice of Intent to Issue a Reexam Certificate, at 8-9 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785917): "'interactive processing' is invoked not in response to a user event detected by the browser (as in the case of Raggett I, supra), but rather in response to the browser application parsing ***an 'embed text format' (i.e., an 'EMBED' tag, see col. 12, line 60, '906 patent)*** that is detected within the hypermedia document when the hypermedia document is first loaded by the browser…***immediately after an 'EMBED' tag is parsed*** and before the hypermedia document is completely displayed in the browser-controlled window."

iii. Interference 105,563 McK

Doyle Annotated Copy of Claims, at 2-3 (July 3, 2007) (563 PH Ex.02 at PH_001_0000787571-72):

1. A method for running an application program { **Fig. 5, item 210** } in a computer network environment { **Fig. 5, item 206** }, comprising:

providing at least one client workstation { **Fig. 5, item 200** } and one network server { **Fig. 5, item 204** } coupled to said network environment { **Fig. 5, item 206** }, wherein said network environment { **Fig. 5, item 206** } is a distributed hypermedia environment { **Fig. 2, item 100** };

executing, at said client workstation { **Fig. 5, item 200** }, a browser application { **Fig. 5, item 208** }, that parses { **Fig. 7A, step 256** } a first distributed hypermedia document { **Fig. 5, item 212** } to identify text formats { **Fig. 5, item 214** } included in said distributed hypermedia document { **Fig. 5, item 212** } and for responding to predetermined text formats { **Fig. 5, item 214** } to initiate processing specified by said text formats { **Fig. 5, item 214** }; utilizing said browser { **Fig. 5, item 208** } to display, on said client workstation { **Fig. 5, item 200** }, at least a portion of a first hypermedia document { **Fig. 5, item 212** } received over said network { **Fig. 5, item 206** } from said server { **Fig. 5, item 204** }, wherein the portion of said first hypermedia document { **Fig. 5, item 212** } is displayed within a first browser-controlled window { **Fig. 9, item 350** } on said client workstation { **Fig. 5, item 200** }, wherein said first distributed hypermedia document { **Fig. 5, item 212** } includes an embed text format { **Fig. 5, item 214** }, located at a first location in said first distributed hypermedia document { **Fig. 5, item 212** }, that specifies the location of at least a portion of an object { **Fig. 5, item 216** } external to the first distributed hypermedia document { **Fig. 5, item 212** }, wherein said object { **Fig. 5, item 216** } has type information associated with it utilized by said browser { **Fig. 5, item 208** } to identify and locate an executable application { **Fig. 5, item 210** } external to the first distributed hypermedia document { **Fig. 5, item 212** }, and wherein said embed text format { **Fig. 5, item 214** } is parsed { **Fig. 7A, step 256** } by said browser { **Fig. 5, item 208** } to automatically invoke { **Fig. 8A, step 290**} said executable application { **Fig. 5, item 210** } to execute on said client workstation { **Fig. 5, item 200** } in order to display said object { **Fig. 5, item 216** } and enable interactive processing of said object { **Fig. 5, item 216** } within a display area created at said first location within the portion of said first distributed hypermedia document { **Fig. 5, item 212** } being displayed in said first browser-controlled window { **Fig. 9, item 350** }.

iv. Second reexam (90/007,858)

Declaration of Edward W. Felten, at ¶ 12 (Sept. 27, 2007) (accompanying Applicants' Response (Sept. 27, 2007)) (858 PH Ex. 06 at PH_001_0000787053): "The claims of the '906 Patent describe a technology that allows web page authors to include, within the boundaries of a web page, interactive objects. ***This is done (briefly stated) by including in the web page's HTML text an embed text format, that provides information about where to get the object's data, along with information to identify and locate an executable application*** that will be invoked on the client computer to display the data and to provide interactivity with it, ***and by providing a web browser that knows how to parse the HTML to extract the embed text format***, how to use type information to identify and locate the executable application, how to invoke the executable application, to execute on the client computer, and how to interface to the executable application so as to allow the user to interact with it within the boundaries of the browser window."

Declaration of Edward W. Felten, at ¶ 21–25 (Sept. 27, 2007) (accompanying Applicants' Response (Sept. 27, 2007)) (858 PH Ex. 06 at PH_001_0000787055):

> "***Mosaic lacked the required embed text format of the '906 claims. Instead, Mosaic used an ordinary hyperlink*** to link to any data that was to be displayed with a helper application. . . .
>
> Hyperlinks in Mosaic were specified using an "A" (short for "Anchor") tag. For example, the HTML element
>
> link
>
> would cause the text "link" to be displayed, typically in underlined blue type. If the user clicked on the underlined word "link", the browser would follow the hyperlink and navigate to the URL "http://example.com/page.html".
>
> Similarly, the HTML element
>
> video
>
> would cause the text "video" to be displayed, in underlined blue type. If the user clicked on the underlined word "photo", the browser would download the file at the URL "http://example.com/video.mpg" and launch a helper application to display it.
>
> ***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***. The use of an embed text format was a significant improvement over the prior art Mosaic browser, as it allowed the browser to recognize immediately that an embedded object was present and special processing was needed."

Declaration of Edward W. Felten, at ¶ 47–55 (Sept. 27, 2007) (accompanying Applicants' Response (Sept. 27, 2007)) (858 PH Ex. 06 at PH_001_0000787058-59):

> Claim 6 of the '906 Patent requires that "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 .. 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 the first distributed hypermedia document ... " ('906 Patent at 18:14-29)
>
> The use of an embed text format is an important element of the invention defined in Claim 6 of the '906 Patent. One drawback of many prior art browsers, such as Mosaic, is that they lacked an embed text format. . . .

Cohen does not disclose the use of an embed text format.

The Examiner states that the "link description tags LDESC" of Cohen are the embed text format (Office Action at p. 31). For the reasons described below, I respectfully disagree with this conclusion.

The 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 in said first distributed hypermedia document ... to display said object and enable interactive processing of said object within a display area created at said first location ... " ('906 Patent at 18:14-28 . . .) ***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. . . .

The fact that the LDESC tag does not appear at a location in the book text is one reason why the LDESC tag cannot be the embed text format of the '906 claims.

The "link tag :L" of Cohen does appear in the book text, but it cannot be the embed text format either, because (e.g.) it lacks the required claim element of an "embed text format ... that specifies the location of at least a portion of an object external to the first distributed hypermedia document ... " ('906 Patent at 18:15- 18). The link tag of Cohen does not specify the location of an object, nor does it specify the location of anything that is external to the first distributed hypermedia document. This is one reason why the "link tag :L" of Cohen cannot be the embed text format.

Cohen's design strategy, of having a small, simple link tag that refers to a larger, more detailed link description in the document prologue, makes sense given the problem that Cohen was trying to solve. Cohen was designed for use with electronic books. These books, unlike Web pages, are large, multi-page files that often repeat graphic elements on different pages. By separating the link tag and link description, Cohen allowed an element to be repeated without having to repeat the full link description each time. Instead, there could be a single link description in the document prologue, and one small link tag at each place in the document where the object was to be used. The claimed '906 design, by contrast, is better suited for use on the Web where individual pages are provided separately. For at least this reason, the use of a single tag is not expressly found or inherently described in Cohen."

Applicants' Response, at 15 (Sept. 27, 2007) (858 PH Ex. 05 at PH_001_0000787044):

> The language of claim 6 recites several limitations relevant to the embed text format. First, the embed text format is located at a first location in a hypermedia document. Secondly, the embed text format specifies the location of at least a portion of an object external to the hypermedia document. Thirdly, the external object is displayed in a display area created at the first location, i.e., the location of the embed text format within the hypermedia document.

Applicants' Response, at 15-16 (Sept. 27, 2007) (858 PH Ex. 05 at PH_001_0000787044):

The claimed embed text format is not explicitly found in Cohen. . . .

The language of claim 6 recites several limitations relevant to the embed text format. First, the embed text format is located at a first location in a hypermedia document. Secondly, the embed text format specifies the location of at least a portion of an object external to the hypermedia document. Thirdly, the external object is displayed in a display area created at the first location, i.e., the location of the embed text format within the hypermedia document. Turning first to the LDESC tag of Cohen, ***the LDESC tag is not located at a first location in the document where a display window is created. Instead, the LDESC (link description) tag appears in the document file's prologue.*** . . .

Turning next to ***the link tag :L of Cohen***, the link tag :L ***does appear in the book text but it lacks the claimed feature that the embed text format specifies the location of at least a portion of an object external to the first distributed hypermedia document. The link tag of Cohen does not specify the location of an object, nor does it specify the location of anything that is external to the first distributed hypermedia document***. [Felten at paragraph 54] Turning finally to the requirement that the external object is displayed in a display area created at the first location, as is discussed in more detail in section 3 below, none of the citations in the office action point to a part of Cohen where the claimed display area is expressly found. In Cohen the I/O handlers are invoked to display objects, however there is no teaching relating the location of a display area. Accordingly, the claimed embed text format is not expressly found in Cohen."

Applicants' Response, at 11 (June 23, 2008) (858 PH Ex. 10 at PH_001_0000787271): "***The Patent Owner continues to respectfully assert the position argued in its response to the non-final office action that Cohen does not fairly teach or suggest many of the features of the claims, including*** at least interactive processing, ***an embed text format, a display area and type information*** as those features are defined by the unamended language of claims 1 and 6, and that this position is correct in view of the cases cited by the examiner . . ."

v. <u>'985 prosecution history (10/217,955)</u>

Applicants' Response, at 5 (March 11, 2005) (985 PH Ex. 03 at PH_001_0000784217, 29-30):

"***[T]he obviousness issues raised in [Office Actions mailed in connection with the reexamination of the parent patent application, A/N 08/324,443 (now the 906 patent)] are identical to the obviousness issues raised in the present Office Action.*** References to these declarations relevant to identical issues raised in the present office action will be made in the following argument."

Response to Office Action at 17–18 (March 11, 2005): "[T]he static images returned by external applications invoked in the Raggett system are inserted in line by the browser into the ordered set of static presentation formats comprising the displayable form of the hypermedia document. ***In Raggett I and II, the Raggett EMBED tag located at a first location in the hypermedia document is parsed, a rendering application is invoked that returns a static image and terminates, the static image is inserted at the first location in the set of static presentation formats, and the presentation form of the document is then displayed by the browser***. Since the rendering application has terminated before the set of static presentation formats is displayed by the browser, it is fundamentally incapable of providing interactive processing of an object being displayed in the display area of a hypermedia document being displayed in the browser controlled window.

Turning next to the Toye reference, NoteMail messages are formatted in MIME (Multipurpose Internet Mail Extension) and a new "Format" MIME data type is defined, for NoteMail to capture and preserve the spatial arrangement of information on a NoteMail page. The MIME "Format" data is stored separate from the text portions of the document [Felten II,

at paragraph 31]. ***There is no teaching in NoteMail of using text formats, within the document text, intended to initiate processes specified by those text formats. 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."***

Applicants' Response, at 22 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784634):

"***identifying an embed text format*** *which corresponds to a first location in the document,*

EXAMPLE SUPPORT:

14:27 'a check is made as to whether the current tag is the EMBED tag.'"

d. <u>Cited prior art</u>

Microsoft Product Support Services Application Note (Text File) GC0165:Rich-Text Format (RTF) Specification (Jun. 1992). cited by other . (Ex. L at PH_001_0000014636 – PH_001_0000014673).

D. "first location" (in various contexts)

Claim Term(s)	Defendants' Proposed Construction	Eolas's Proposed Construction
embed text format, located at a first location in said first distributed hypermedia document	tag located at the place in the received document where the embedded object will appear within the displayed document	embed text format located at a first location in the first distributed hypermedia document
embed text format [which] correspond[s/ing] to [a / said] first location in the document	tag located at the place in the received file where the embedded object will appear within the displayed document	embed text format which relates to a first location in the document

1. Defendants' intrinsic evidence

a. Claims

In the following chart, the term(s) for construction appear in all the claims marked with an "x." The letter "m" indicates a method claim, and "¶6" indicates a claim that the Defendants contend includes limitations subject to § 112, ¶ 6.

	'906 patent						'985 patent								
	m 1	m 4	m 5	¶6 6	¶6 9	¶6 10	m 1	¶6 16	m ¶6 20	m ¶6 24	¶6 28	m ¶6 32	m 36	m ¶6 40	m 44
embed text format, located at a first location in said first distributed hypermedia document	x	x	x	x	x	x									
embed text format [which] correspond[s/ing] to [a / said] first location in the document							x	x	x	x	x	x	x	x	x

b. Specification (all cites to '906 patent)

12:54–:65 (Detailed Description of a Preferred Embodiment): Table II, below, shows an example of an HTML tag format used by the present invention to embed a link to an application program within a hypermedia document.

TABLE II

```
<EMBED
    TYPE = "type"
    HREF = "href"
    WIDTH = width
    HEIGHT = height
>
```

6:66–:67 (Detailed Description of a Preferred Embodiment): As shown in Table II, the EMBED tag includes TYPE, HREF, WIDTH and HEIGHT elements."

13:38–:40 (Detailed Description of a Preferred Embodiment): "The routines within HTMLparse.c perform the task of ***parsing a hypermedia document and detecting the embed tag***. (985, 13:38-40).

14:14–:24 (Detailed Description of a Preferred Embodiment) and Fig. 7A: [I]t is assumed that a hypermedia document has been obtained at a user's client computer and that a browser program executing on the client computer displays the document and calls a first routine in the HTMLparse. c file called "HTMLparse". . . . ***[T]he document is parsed or scanned for HTML tags or other symbols.*** While the file HTMLparse.c includes routines to handle all possible tags and symbols that may be encountered, ***FIG. 7A . . . illustrates the handling of EMBED tags.***

14:24–:32 (Detailed Description of a Preferred Embodiment): Assuming there is more text to parse, execution proceeds to step 256 where routines in HTMLparse.c obtain the next item (e.g. word, tag or symbol) from the document. At step 258 a check is made as to whether the current tag is the EMBED tag. If not, execution returns to step 254 where the next tag in the document is obtained. If, at step 258, it is determined that the tag is the EMBED tag, execution proceeds to step 260 where an enumerated type is assigned for the tag.

14:33–:34 (Detailed Description of a Preferred Embodiment): Each occurrence of a valid EMBED tag specifies an embedded object.

c. Prosecution history

i. '906 prosecution history (08/324,443)

Applicants' Response, at 1–3 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784030): (amending claim to add the limitations "an embed text format, located at a first location in said first distributed hypermedia document" and enable interactive processing of said object within [the] a display area created at said first location within the portion of said first distributed hypermedia document being displayed . . ." to claims for the first time)

Applicants' Response, at 6 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784034): "The present invention, as defined for example in amended claim 1, includes the step of executing a browser that parses a first distributed hypermedia document to identify ***text formats included in the distributed hypermedia document***. . . . ***The first distributed hypermedia document includes an***

embed text format located at a first location in the document. . . . ***The external application displays***, and allows the user to interactively process, the object in a display window created within the portion of the document being displayed in the browser-controlled window, ***at the location within the document of the embed text format***."

Applicants' Response, at 11 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784039): "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***." (distinguishing Mosaic)

Applicants' Response, at 20 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784048): "Combining Hansen with any combination of Mosaic and Khoyi, while perhaps possible, would produce features that are irrelevant to the present application. Such a combination would involve modifying the hypermedia document data structure to allow multiple hierarchical subdocument windows to be contained within a parent document. This would involve substantial modifications to the Mosaic document rendering engine, as well as the development of a new version of the HTML document definition protocol to allow definition of hierarchical relationships within subdocument elements."

Declaration of Michael D. Doyle, at 19 (May 27, 1997) (accompanying Applicants' Response (June 2, 1997)) (906 PH Ex. 11 at PH_001_0000784049): "[T]he features of the claimed invention [were] . . . incorporated through . . . ***display[ing] an external object . . . within a display window created at the embed text format's location within the hypermedia document being displayed*** . . .

Applicants' Response, at 5 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783906): (amending what was then claim 24 to include "parsing said document to locate a reference to the external object included in the document, with the reference identifying and locating the external object")

Examiner Interview Summary Record, at 1 (Feb. 26, 1997) (906 PH Ex. 09 at PH_001_0000784011): "How Hypernet work and different from the present invention." as follows . . . 2) ***tag in document*** to activate external program (delayed binding) . . . Applicant's argument is persuasive to overcome the Hypernet ref."

Declaration of Michael D. Doyle, at 1–2, (Oct. 29, 1997) (accompanying Applicants' Remarks (Oct. 31, 1997)) (906 PH Ex. 13 at PH_001_0000784102-03): "Mosaic (the browser) interprets ***the HTML <EMBED> tag included in the document*** to create a drawing area widget in a document presentation and creates a shared window system buffer to receive visualization results."

Applicants' Response, at 2 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784132): "THE INVENTION OF CLAIM 1 . . . ***The hypermedia document includes an embed text format, located at a first location in the hypermedia document*** . . . . When the automatically-invoked application executes, it displays the object and enables interactive processing of said object ***within a display window created within the portion of the hypermedia document being displayed***."

Applicants' Response, at 16 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783894): Distinguishes CGI as follows: "[U]nlike 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."

Examiner Interview Summary, at 1 (Jan. 27, 1998) (906 PH Ex. 18 at PH_001_0000784173): "The applicant agreed to amend 'display window' in line 28 of claim 1 to -display area- to distinguish that it is an area ***within*** the hypermedia document that displays the object and not a separate window. The same amendment was made to claim 44, line 39."

Applicants' Response, at 2 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784132): "Mosaic provides display and interaction with an external object by launching an associated program in a ***separate window***." (distinguishing Mosaic)

Applicants' Response, at 2 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784132): "THE INVENTION OF CLAIM 1 … displays the object and enables interactive processing of said object within ***a display window created within the portion of the hypermedia document being displayed***."

Applicants' Response, at 7 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784137): "[In Mosaic t]he retrieved information either replaces the first hypermedia document, or is displayed in a separate window other than the window displaying the hypermedia document." (distinguishing Mosaic)

Applicants' Response, at 8 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784138): "[In Mosaic, t]he viewer program displays the full image ***in a separate 'window'*** (in a windowing environment) or on a separate screen." (distinguishing Mosaic)

Applicants' Response, at 11 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784039): "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." (distinguishing Mosaic)

Applicants' Response, at 20 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784040): "Furthermore, even if the above combination was operable, the external applications still could not be interacted with from within the hypermedia document, as required by the claimed invention, since both Mosaic and Khoyi must launch any external application into a separate window before the reader can interactively control it." (distinguishing prior art)

Applicants' Response, at 24 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784052): "Neither Mosaic, nor Khoyi, nor Hansen shows an executable application which is external to a document being displayed and interactively processed ***within that document's display window***, nor do they show such an application where said executable application is interactively controlled on said client workstation by interprocess communications between the external application and the browser. This feature produces surprising and unexpected results over the prior art, since it allows the reader to perform all necessary interactive functions with external applications without directing his or her attention away from the hypermedia document."

ii. Abandoned application (09/075,359)

See Applicants' Response, at 1-4 (March 9, 2001) ([PH_001_0000787808] – [PH_001_0000787813]) (359 PH Ex. 04 at PH_001_0000787808-11): (claims as amended)

Applicants' Response, at 2 (Nov. 29, 2001) (359 PH Ex.04 at PH_001_0000787824): "The present ***invention***, as defined for example in claim 62, is a computer program product for use in a system having at least one client workstation and one network server coupled to a network environment. The network environment is a distributed hypermedia environment, and the client workstation utilizes a browser to display. on the client workstation, at least a portion of a first hypermedia document received over the network from the server. The portion of the first hypermedia document is displayed within a first browser-controlled window on the client workstation. ***The first distributed hypermedia document includes an embed text format, located at***

a first location in the first distributed hypermedia document, that specifies, either directly or indirectly, the location of at least a portion of the object, where the portion is external to the first distributed hypermedia document, where the object has type information associated with it utilized to identify and locate computer readable program code external to the first distributed hypermedia document, and ***where the embed text format is parsed by the browser to automatically invoke the computer readable program code***. The claimed computer program product includes a computer usable medium having computer readable program code physically embodied therein, and further includes computer readable program code, identified by the type information, for being automatically invoked by the browser application to cause the client workstation to ***display an object and enable interactive processing of the object within the di1.play area created at the first location within the portion of the first distributed hypermedia document being displayed in the first browser controlled window.***"

Response to Office Action, at 7 (Nov. 29, 2001) (359 PH Ex. 06 at PH_001_0000787829): ***In the claimed invention***, the document itself coordinates the use of external program code with embed text formats, such as the Netscape <embed> tag or the ActiveX <object> tag, at ***locations in the document where the external computer readable code is to display and enable interactive processing of an external object.***

iii. First reexam (90/006,831)

Applicants' Remarks, at 3 (May 11, 2004) (831 PH Ex. 07 at PH_001_0000785361):

> ***The claims recite a browser*** application, executed on the client workstation, ***that parses a hypermedia document to identify text formats in the document*** and responds to predetermined text formats to initiate processing specified by the text formats.
>
> The browser displays a portion of a first distributed hypermedia document, received over the network from the network server, in a browser-controlled window. ***The hypermedia document includes an embed text format, located at a first location in the hypermedia document***, that specifies the location of at least a portion of an object external to the hypermedia document. The object has associated type information utilized by the browser to identify and locate an executable application external to the hypermedia document.
>
> When an embed text format is parsed by the browser, the executable application is automatically invoked, as a result of the parsing, to execute on the client workstation.
>
> When the automatically invoked application executes on the client workstation, ***the object is displayed within a display window created within the portion of the hypermedia document being displayed*** and interactive processing of said object is enabled.

Notice of Intent to Issue a Reexam Certificate, at 8–9 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785916-17)): "The instant claimed '906 'executable application' that provides the claimed 'interactive processing' is invoked not in response to a user event detected by the browser (as in the case of Raggett I, *supra*), but rather in response to the browser application parsing ***an 'embed text format' (i.e., an "EMBED" tag***, see col. 12, line 60, '906 patent) ***that is detected within the hypermedia document when the hypermedia document is first loaded by the browser."***

Notice of Intent to Issue a Reexam Certificate, at 9 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785917): "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."

Applicants' Remarks, at 4 (May 11, 2004) (831 PH Ex. 07 at PH_001_0000785362): "[In the prior art, t]he retrieved information either replaces the first hypermedia document or is displayed in a separate window other than the window displaying the hypermedia document." (distinguishing prior art)

iv. Second reexam (90/007,858)

Office Action, at 8 (July 30, 2007) (858 PH Ex. 03 at PH_001_0000786953): "[T]he helper program prior art submitted by the Third Party Requester . . . are not seen to teach of executing a browser application at said client workstation that parses a first distributed hypermedia document, ***having an embed text format included in the hypermedia document*** that specifies the location of at least a portion of an object external to the first distributed hypermedia document, and subsequently, to automatically invoke an external helper application to execute a processing to display the external object and enable interactive processing of the external object." ((emphasis in original; some emphasis omitted))

Office Action, at 26, 31 (July 30, 2007) (858 PH Ex. 03 at PH_001_0000786971): "[E]mbed text format[s are] . . . interpreted as the multimedia link description tags LDESC included within the document. . . in the prologue of the document. . . . The LID attribute refers to one or more LDESC document link tags."

Declaration of Edward W. Felten, at ¶¶ 51-52 (Sept. 27, 2007) (accompanying Applicants' Response (Oct. 1, 2007)) (858 PH Ex. 06 at PH_001_0000787059): "The 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 in said first distributed hypermedia document*** . . . ***to display said object and enable interactive processing of said object within a display area created at said first location*** . . .' ('906 Patent at 18:14-28) ***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 . . ."

Response to Office Action, at 15 (Oct. 1, 2007) (858 PH Ex. 05 at PH_001_0000787044): "[C]laim 6 recites [that] . . . the external object is displayed in a display area created at ***the first location, i.e., the location of the embed text format within the hypermedia document.*** Turning first to the LDESC tag of Cohen, ***the LDESC tag is not located at a first location in the document where a display window is created.*** Instead, the LDESC (link description) tag appears in the document file's prologue." (distinguishing prior art)

Declaration of Edward W. Felten, at ¶¶ 53 (Sept. 27, 2007) (accompanying Applicants' Response (Oct. 1, 2007)) (858 PH Ex. 06 at PH_001_0000787059): "The fact that the LDESC tag does not appear at a location in the book text is one reason why the LDESC tag cannot be the embed text format of the '906 claims."

Declaration of Edward W. Felten, at ¶ 34 (Sept. 27, 2007) (accompanying Applicants' Response (Oct. 1, 2007)) (858 PH Ex. 06 at PH_001_0000787056): "The ability of the claimed '906 technology to display and enable interactive processing within the browser window was a significant advance over the prior art Mosaic browser. ***Enabling display and interactivity within the browser window allowed the object to appear, seamlessly, as an integral part of the web page's display."***

v. '985 prosecution history (10/217,955)

Applicants' Response, at 8 (March 11, 2005) (985 PH Ex. 03 at PH_001_0000787056): "The invention, as recited for example in claim 1 [is one in which] . . .***[t]he hypermedia document includes an embed text format, located at a first location in the hypermedia document*** . . . . [which ultimately] cause[s] the client workstation to display an object and enable interactive processing of the object within a display window created at the first location of the portion of the hypermedia document being displayed in the first browser controlled window."

Applicants' Response, at 17 (March 11, 2005) (985 PH Ex. 03 at PH_001_0000784229): "In Raggett I and II, ***the Raggett EMBED tag located at a first location in the hypermedia document is parsed***, a rendering application is invoked that returns a static image and terminates, ***the static image is inserted at the first location in the set of static presentation formats***, and the presentation form of the document is then displayed by the browser. . . . ."

Applicants' Response, at 18 (March 11, 2005) (985 PH Ex. 03 at PH_001_0000784230): "The MIME 'Format' data is stored separate from the text portions of the document [Felten II, at paragraph 31]. There is no teaching in NoteMail of using text formats, within the document text, intended to initiate processes specified by those text formats. 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." (distinguishing prior art)

Applicants' Response, at 28 (March 11, 2005) (985 PH Ex. 03 at PH_001_0000784240): "[In a prior art system,] the external application for processing the 'dynamic object' is not automatically invoked ***when an embed text format within the document*** is parsed . . ." (distinguishing prior art)

Declaration of Edward W. Felten, at ¶ 18 (May 7, 2004) (accompanying Applicants' Response (March 11, 2005)) (985 PH Ex. 04 at PH_001_0000784248): "The claims of the '906 Patent describe a technology that allows web page authors to include, ***within the boundaries of a web page***, interactive objects. ***This is done (briefly stated) by including in the web page's HTML text an embed text format, . . . and by providing a web browser that knows how to parse the HTML to extract the embed text format . . . [and ultimately] how to interface to the executable application so as to allow the user to interact with it within the boundaries of the browser window***."

Declaration of Edward W. Felten, at ¶ 26 (May 7, 2004) (accompanying Applicants' Response (March 11, 2005)) (985 PH Ex. 04 at PH_001_0000784276): "Toye teaches no software application that parses distributed hypermedia documents, . . . [or] pars[es] ***an embed text format in such a document***." (distinguishing prior art)

Applicants' Response, at 9 (March 11, 2005) (985 PH Ex. 03 at PH_001_0000784221): "The retrieved information either replaces the first hypermedia document or is displayed in a separate window other than the window displaying the hypermedia document."

Declaration of Edward W Felten, at ¶ 26 (Oct. 6, 2004) (accompanying Applicants' Response, at 15 (March 11, 2005)) (985 PH Ex. 05 at PH_001_0000784276): "Toye does not teach the use of a hypermedia browser, as that term is used in the '906 claims. Toye teaches no software application that parses distributed hypermedia documents, and it does not teach other browser-related elements of the '906 claims, such as parsing of distributed hypermedia documents by a browser, identifying text formats in distributed hypermedia documents and responding to predetermined text formats to initiate processing specified by those formats, utilizing a browser to display at least a portion of a distributed hypermedia document in a browser-controlled window, and ***parsing an embed text format in such a document.***"

See Supplemental Response / Amendment, at 2–13 (April 11, 2008) (985 PH Ex. 11 at PH_001_0000784569-80): Amendments to claims.

Supplemental Amendment, at 18–24 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784630,34,36):

> "The examiner requested that citations to support in the specification for the elements and limitations of the pending claims be provided in the remarks section of a newly presented supplemental amendment. . . .
>
> "***identifying an embed text format*** which corresponds to a first location in the document,
> EXAMPLE SUPPORT:
> 14:27 'a check is made as to whether the current tag is the EMBED tag.' . . . (emphasis in original)
>
> "***while the object is being displayed within a display area*** created ***at the first location*** within the portion of the hypermedia document being displayed in the browser-controlled window.
> EXAMPLE SUPPORT:
> 16:8 'FIG. 9 is a screen display of the invention showing an interactive application object (in this case a three dimensional image object) in a window within a browser window. In FIG. 9, the browser is NCSA Mosaic version 2.4. The processes VIS, Panel and VRServer work as discussed above. FIG. 9 shows screen display 356 Mosaic window 350 containing image window 352 and a portion of a panel window 354. Note that image window 352 is within Mosaic window 350 while panel window 354 is external to Mosaic window 350. Another possibility is to have panel window 354 within Mosaic window 350.'" (emphasis in original)

Notice of Allowance, at 2 (March 20, 2009) (985 PH Ex. 15 at PH_001_0000784733): "The following is an examiner's statement of reasons for allowance: the claims are allowable as the claims contain the subject matter deemed allowable in both Re exam 90/006,831 and Re exam 90/007,838 for the same reasons as set forth in the NIRC of the two Re exams."

2. <u>Defendants' extrinsic evidence</u>

Testimony by inventor Michael Doyle from the *Eolas v. Microsoft* case, including without limitation:

> Michael Doyle Dep., Eolas *Techs Inc. v. Microsoft Corp.*, No. 99- C-626 (N.D. Ill. February 28-March 1, 2000), at 116:14–118:11 [Ex. FF (EOLASTX-E- 0000000180)]:
>
> THE WITNESS: Well, if you look at figure 1, prior art, it shows an image icon that is displayed within a hypermedia document.
> BY MR. PETERSEN:
> Q. Which element of figure 1 are you referring to?
> A. Element 22.
> Q. So your answer to the question is yes?
> A. Yeah, that's a display area within the document, as described in the spec.
> Q. But the prior art included the capability

page 117

of displaying an object within a display area?
A. As we stated in our specification.
Q. Did the prior art have the capability of displaying the object at a location within the document?
MS. CONLIN: And are you referring to any prior art now, or are you referencing Mosaic QuickTime? Are we still on the QuickTime, or are we moving into the more general?
MR. PETERSEN: Well, it's not really specifically QuickTime. QuickTime, I believe -- correct me if I'm wrong, Dr. Doyle -- you testified was a helper application? And I'm not talking about helper applications.
MS. CONLIN: He's talking generally now.
MR. PETERSEN: I'm talking about Mosaic at the time that you made the invention in the '906 patent by itself.
THE WITNESS: Displaying static images within --
MR. PETERSEN: In response to the --
THE REPORTER: One at a time.
THE WITNESS: I'm sorry, go ahead. I'll let you expand.

page 118

MR. PETERSEN: I think we're on the same page.
Q. Display static images in response to parsing the IMG tag, for example, right?
A. Uh-huh.
Q. That's the context I'm talking about. Do you understand?
A. I understand, yeah, sure.
Q. Now, did that operate to display an image at the location in the document where the IMG text format is located?
A. Yes.

Michael Doyle Dep., Eolas Techs Inc. v. Microsoft Corp., No. 99-C-626 (N.D. Ill. February 28-March 1, 2000), at 558:15–560:7 [Ex. FF (EOLASTX-E-0000000182)].

Q. And looking at the second page of the exhibit near the top, there's a portion of underlined language that says "located at a first location," and so on?
A. I see that.
Q. What does the location refer to there?
MS. CONLIN: The first location, Counsel, is that what you're referencing?
MR. PETERSEN: That's correct. It says, "located at a first location."

page 559

Q. So the question is what is the first location?
A. It is a location of an embed text format as defined in claim 6 of the '906 patent. Well, sorry, as defined in this amended claim 1, but it appears to refer to the same meaning as defined in claim 6 of the '906 patent.
Q. Now, does that refer to the location of the embed text format within the hypermedia document?
A. You'd have to define -- well, within the hypermedia document referred to in this claim? Yes, it says, "at a first location in said first distributed hypermedia document."
Q. Now, is that referring to the location of the embed text format relative to other text formats that may be in the hypermedia document?
A. It's referring to the location with respect to the ordered definition of elements within the hypermedia document.
Q. And is the ordered definition of elements in the hypermedia document the order in which they appear in the text file of the document?
MS. CONLIN: Objection as to form.

page 560

THE WITNESS: To the extent that there can be multiple mappings of the location in the hypermedia document with respect to anything on the display, that's referring to the location within the hypermedia document date, in this case, the specific embodiment described in the specification would be within the HTML file.

Testimony by inventor David Martin from the Eolas v. Microsoft case, including without limitation:

David Martin Dep., *Eolas Techs Inc. v. Microsoft Corp*., No. C-99-0212 (N.D. Ca. January 20-21, 2000), at 151:4–153:10; 164:10–166:4 [Ex. DD (EOLASTX-E-0000000174)].

Q. What does it mean to say that the embed text format is at a location in a hypermedia document?
A. Referring back to Column 14, at line 13, it says that "Returning to Figure 7, it is assumed that a hypermedia document is obtained" -- "has been obtained at a user's client computer and that a browser program executing on the client computer displays the document and calls a first routine in the HTMLparse.c file called 'HTMLparse.' This first routine, HTMLparse, is entered at step 252 where a pointer

to the start to the document portion is passed. Steps 254, 256 and 258 represent a loop where the document is parsed or scanned for HTML tags or other symbols."

The sequence in which that parsing occurs indicates a location within the hypermedia document.

MS. CONLIN: I know we haven't been quite going an hour, but I'd like to take a break when you get a chance.

page 152

MR. PETERSEN: Sure, that's fine.

MS. CONLIN: That would be fine.

VIDEO TECHNICIAN: Going off the video record. The time is now approximately 2:20 p.m.

(Whereupon, a short break was taken from 2:20 to 2:32 p.m.)

VIDEO TECHNICIAN: We're back on the video record. The time is now approximately 2:32 p.m.

BY MR. PETERSEN:

Q. So if I understand your testimony then, Mr. Martin, the location of the embed text format within a hypermedia document has to do with the order in which it's parsed by the browser?

A. As referred to in the specification, yes.

Q. So then what does it mean to say that an object is displayed at that location?

A. Can you refer me to --

Q. Yes, it was from the passage of the Claim 6 that you read before, line 25 of Column 18, it actually goes on for about three or four lines.

page 153

A. For example, do you mean beginning in Column 18, line 15, "includes an embed text format, located at a first location in said first hypermedia document"?

Q. Yes, we just talked about that and you said that location is where within the hypermedia document the embed text format is located in terms of the order in which the document is parsed; right?

A. That is correct.

* * * *

Q. Did you define the term "location" in your patent specification?

A. I believe that is defined, but I'd have to refer back to the text.
MS. CONLIN: Can I have the question back.
(Whereupon, the reporter read back the last question.)
THE WITNESS: I think the term "location" has been defined in different contexts for different purposes within the scope of the specification of the patent.
BY MR. PETERSEN:
Q. No, what I mean is the term "location" with quotes around it or the word "location" found in the patent specification

page 165

with a definition associated with it?
A. Can you point me to a place where the word "location" is used with the quotation marks around it?
Q. I can't find -- I don't think there is any definition of the word "location" in the patent specification, but you're the inventor so I thought I would ask you.
A. Well, it depends on the context in which the term is used.
Q. In the context of Claim 6, Column 18.
A. So if by that do you mean the first location of the first -- the first distributed hypermedia document or do you mean the specifies the location of at least a portion of an object external?
Q. I mean "located at a first location in said first hypermedia document," that was a fair question.
A. Yes, we've already reviewed that in terms of the parsing functionality.
Q. No, but it is the term definition itself, is the term "location" itself defined in the patent specification?

page 166

A. To one who is fluent in the art, the understanding of the reading of the specification in regard to the term "location" in regard to parsing of the hypermedia document is clear.

Testimony by inventor Cheong Ang from the *Eolas v. Microsoft* case, including without limitation:

Cheong Ang Dep., Eolas Techs Inc. v. Microsoft Corp., No. C-99-0212 (N.D. Ca. January 21-22, 2000), at 241:22–242:1; 243:1–12 [Ex. EE (EOLASTX-E-0000000177)]:

Q. In the system described here, the browser creates a display; is that correct?
A. There is "a display area created at said first location within the portion of said

page 242

first hypermedia document."

* * * *

Q. Well, in Figure 9 you can tell where the Block 352 is on the screen, can't you?
A. Okay. What the claim said here is "the display area was created at said first location within the portion of said first hypermedia document being displayed," so to the user it's the hypermedia document they're looking at.
Q. Well, what about the display area?
A. And the display area corresponds to how the hypermedia document, how it's laid out in the hypermedia document.

E. "specifies the location of at least a portion of [an / said] object"

Claim Term(s)	Defendants' Proposed Construction	Eolas's Proposed Construction
specifies the location of at least a portion of [an / said] object	specifies the location of at least a portion of [an / said] object[2]	specifies the location of at least part of an object

1. Defendants' intrinsic evidence

a. Claims

In the following chart, the term(s) for construction appear in all the claims marked with an "x." The letter "m" indicates a method claim, and "¶6" indicates a claim that the Defendants contend includes limitations subject to § 112, ¶ 6.

	'906 patent						'985 patent								
	m 1	m 4	m 5	¶6 6	¶6 9	¶6 10	m 1	¶6 16	m ¶6 20	m ¶6 24	¶6 28	m ¶6 32	m 36	m ¶6 40	m 44
specifies the location of at least a portion of [an / said] object	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

a. Specification (all cites to '906 patent)

2:44-47 (Background of the Invention): The mechanism for ***specifying*** and locating a linked object such as hypermedia document 14 is an HTML "element" that ***includes an object address in the format of a Uniform Resource Locator (URL)***.

14:32-33 (Detailed Description of a Preferred Embodiment): Each occurrence of a valid EMBED tag ***specifies*** an embedded object.

14:66-67 (Detailed Description of a Preferred Embodiment): [T]he data object ***specified by the URL*** in the EMBED tag.

b. Prosecution history

i. '906 prosecution history (08/324,443)

Applicants' Response, at 14–15 (Aug. 6, 1996) (906 PH Ex. 3 at PH_001_0000783892-93):

> The Mercury Project is an interactive Web page that utilizes CGI (Common Gateway Interface) scripts and the HTML <FORM> tag to facilitate interaction between the user and the Web page.

[2] Where "specifies" has its common meaning: "to name or state explicitly or in detail." *See Merriam-Webster's Collegiate Dictionary* 1132 (9th ed. 1991).

* * * * *

The <FORM> tag causes the browser to send a string of characters, entered into a form in the original HTML document, to the Web server application. The Web server invokes a CGI application identified by the ACTION= attribute of the tag and passes the string to the CGI application.

* * * * *

As described above, the Mercury Project utilizes CGI where a <FORM> tag identifies a program on the server but not an external object.

Applicants' Response, at 19–20 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784047-48):

Turning second to modifying Khoyi, any implementation of the functionality of Mosaic in the Khoyi operating system would be inoperable, due to the incompatibilities between the two systems' linking systems described below, and due to the differences between the two systems in storing, locating and processing data objects.

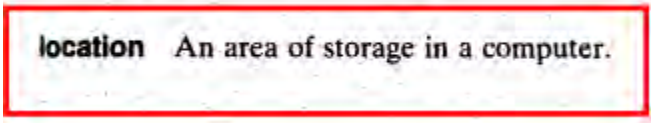
* * * * *

The two linking systems could not be combined into one, due to the architecture of Khoyi's object system. Since Khoyi does not represent links within the document itself, but rather uses a link table which is external to the document, some sort of mechanism would have to be created to allow such links to be fully defined by the document text itself. This modification would render Khoyi's object system inoperable, since Khoyi's entire application architecture depends upon the link tables being the source of all link definitions, and being accessible to all of the various programs that may have a need to operate on a given data object.

Furthermore, since a document in the Khoyi system does not allow the document author to explicitly define or control the definition of the link's internal details, the document itself cannot specify such details as the precise location of a data file on a remote network disk drive. The Web, on the other hand, employs a uniform resource locator (URL) construct to manage both link definition and object localization on networked systems, from within the Web document, under the precise control of the Web document author.

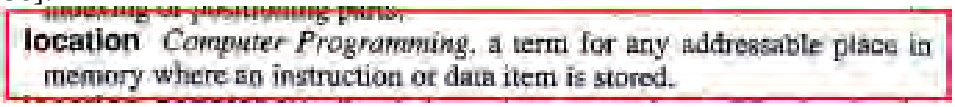
2. Defendants' extrinsic evidence

21st Centry Dictionary of Computer Terms 211 (1994) ("location") [Ex. X at PA-0000333440]:



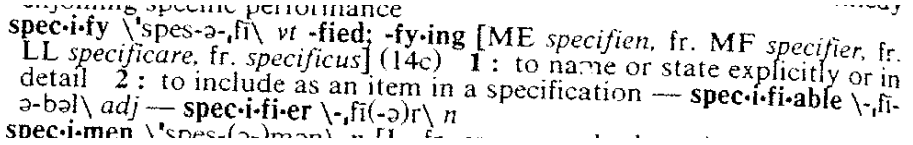
location An area of storage in a computer.

Academic Press Dictionary of Science and Technology 1262 (1992) ("location") [Ex. T at PA-0000333386]:



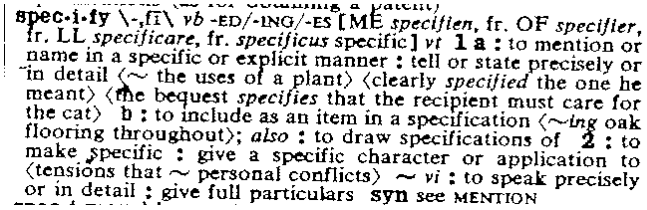
location *Computer Programming*, a term for any addressable place in memory where an instruction or data item is stored.

Merriam-Webster's Collegiate Dictionary 9th Edition 1132 (1991) ("specify") [Ex. S at PA-00333500]:



spec·i·fy \'spes-ə-ˌfī\ *vt* **-fied; -fy·ing** [ME *specifien*, fr. MF *specifier*, fr. LL *specificare*, fr. *specificus*] (14c) **1 :** to name or state explicitly or in detail **2 :** to include as an item in a specification — **spec·i·fi·able** \-ˌfī-ə-bəl\ *adj* — **spec·i·fi·er** \-ˌfī(-ə)r\ *n*

Webster's Third New International Dictionary 2187 (1993) ("specify") [Ex. W at PA-00333503]:



spec·i·fy \-ˌfī\ *vb* -ED/-ING/-ES [ME *specifien*, fr. OF *specifier*, fr. LL *specificare*, fr. *specificus* specific] *vt* **1 a :** to mention or name in a specific or explicit manner **:** tell or state precisely or in detail ⟨~ the uses of a plant⟩ ⟨clearly *specified* the one he meant⟩ ⟨the bequest *specifies* that the recipient must care for the cat⟩ **b :** to include as an item in a specification ⟨~*ing* oak flooring throughout⟩; *also* **:** to draw specifications of **2 :** to make specific **:** give a specific character or application to ⟨tensions that ~ personal conflicts⟩ ~ *vi* **:** to speak precisely or in detail **:** give full particulars **syn** see MENTION

F. "identify an embed text format" (in various contexts)

Claim Term(s)	Defendants' Proposed Construction	Eolas's Proposed Construction
identify[ing] an embed text format	detecting an embed text format during parsing of a hypermedia document	identifying an embed text format
an embed text format . . . is identified		an embed text format is identified

1. Defendants' intrinsic evidence

a. Claims

In the following chart, the term(s) for construction appear in all the claims marked with an "x." The letter "m" indicates a method claim, and "¶6" indicates a claim that the Defendants contend includes limitations subject to § 112, ¶ 6.

	'906 patent						'985 patent								
	m 1	m 4	m 5	¶6 6	¶6 9	¶6 10	m 1	¶6 16	m ¶6 20	m ¶6 24	¶6 28	m ¶6 32	m 36	m ¶6 40	m 44
identify[ing] an embed text format							x	x	x				x	x	
an embed text format . . . is identified										x	x	x			x

b. Specification

1:53–:60 & 2:23–:28 (Background of the invention): Other Internet standards are the HyperText Transmission Protocol ("HTTP") that allows hypertext documents to be exchanged freely among any computers connected to the Internet and ***HyperText Markup Language ("HTML") that defines the way in which hypertext documents designate links to information***. See, e.g., Berners-Lee, T. J., "The world-wide web," Computer Networks and ISDN Systems 25 (1992). … ***A hypermedia document is similar to a hypertext document***, except that the user is able 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.

2:43–:48 (Background of the invention): ***The mechanism for specifying and locating a linked object*** such as hypermedia document 14 ***is an HTML "element" that includes an object address in the format of a Uniform Resource Locator (URL)***.

5:24–:38 (Background of the invention): The Internet is said to provide an "open distributed hypermedia system." It is an "open" system since Internet 100 implements a standard protocol that each of the connecting computer systems, 106, 130, 120, 132 and 134 must implement (TCP/IP). ***It is a "hypermedia" system because it is able to handle hypermedia documents*** as described above

via standards such as the HTTP and ***HTML hypertext transmission and mark up standards***, respectively.

9:50–:58 (Detailed description of a preferred embodiment): This means that application client 210 can make requests over network 206 for data objects, such as multidimensional image objects. For example, application client 210 may request an object, such as object 1 at 216, located in server computer 204. Application client 210 may make the request by any suitable means. Assuming network 206 is the Internet, such a request would typically be made by using HTTP in response to a ***HTML-style link definition for embedded program link 214***.

14:64–:67 (Detailed description of a preferred embodiment): FIG. 8A is a flowchart for routine HTMLwidget. HTMLwidget creates display data structures and launches an external application program to handle the data object specified by the URL in the EMBED tag.

c. Prosecution history

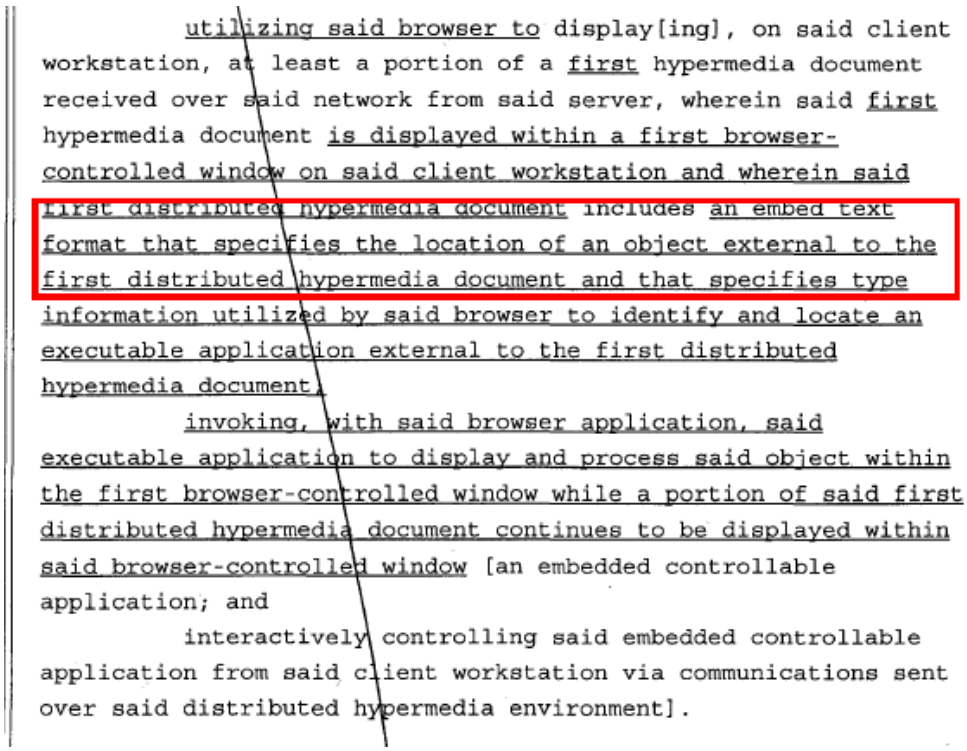
i. '906 prosecution history (08/324,443)

Amendment A, at 1-2 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783879-80):

1. (Amended) 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 distributed hypermedia document to identify text formats included in the distributed hypermedia document and for responding to predetermined text formats to initiate processes specified by the text format;



utilizing said browser to display[ing], on said client workstation, at least a portion of a first hypermedia document received over said network from said server, wherein said first hypermedia document is displayed within a first browser-controlled window on said client workstation and wherein said first distributed hypermedia document includes an embed text format that specifies the location of an object external to the first distributed hypermedia document and that specifies type information utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document,

invoking, with said browser application, said executable application to display and process said object within the first browser-controlled window while a portion of said first distributed hypermedia document continues to be displayed within said browser-controlled window [an embedded controllable application; and

interactively controlling said embedded controllable application from said client workstation via communications sent over said distributed hypermedia environment].

Amendment A, at 13 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783891):

The present invention, as defined for example in amended claim 1, includes the steps of executing, at the client workstation, a browser application that parses a distributed hypermedia document to identify text formats included in the distributed hypermedia document and for responding to text formats to initiate processes specified by that text format. The browser is also utilized to display at least a portion of the distributed hypermedia document within a browser-controlled window.

In order to insure cross-platform uniformity of document appearance, the document was defined through the use of ASCII text, where specific text formats, otherwise known as "tags," would be used within the document text to specify various aspects of the document's appearance and linkages to other documents or related data. Each browser, therefore, incorporated a parser which would distinguish the formatting tags from the document's narrative text, classify those tags into pre-defined categories, break each tag into its basic components, and then invoke appropriate browser subroutines to respond appropriately to the meanings of the tag components. Although the browser subroutines were built from machine-specific native code, this text tag mechanism allowed the design of a variety of browsers for various computing platforms that could respond in similar ways to similar types of text tags, and therefore result in similar-appearing documents on dissimilar computers. A binary document data format was avoided in order to promote cross-platform compatibility, due to the variation in binary data handling methodologies on various different operating systems, and to simplify the requirements for document creation tools. All that a Web document author needs in order to create a Web document file is a simple ASCII text editor, which is a pre-existing application in all commonly-found operating system packages.

ii. First reexam (90/006,831)

Declaration of Edward W. Felten, at ¶ 18 (May 11, 2004) (831 PH Ex. 10 at PH_001_0000785440):

18. The claims of the '906 Patent describe a technology that allows web page authors to include, within the boundaries of a web page, interactive objects. This is done (briefly stated) by including in the web page's HTML text an embed text format, that provides information about where to get the object's data, along with information to identify and locate an executable application that will be invoked on the client computer to display the data and to provide interactivity with it, and by providing a web browser that knows how to parse the HTML to extract the embed text format, how to use type information to identify and locate the executable application, how to invoke the executable application, to execute on the client computer, and how to interface to the executable application so as to allow the user to interact with it within the boundaries of the browser window.

Declaration of Edward W. Felten, at ¶ 25 (May 11, 2004) (831 PH Ex. 10 at PH_001_0000785441):

D. The Berners-Lee Reference

25. The Berners-Lee reference is a specification for the HTML markup language. HTML is a language used by Web page authors to describe the structure and desired contents of their pages. A browser parses an HTML document to determine its structure and then displays the visual representation of the specified items within a browser window.

Office Action, at 4 (Aug. 16, 2004) (831 PH Ex. 12 at PH_001_0000785558):

It would have been obvious to a skilled artisan to combine (1) the teachings of Berners-Lee regarding the processing of HTML documents performed by a browser, with (2) the HTML browser of the admitted prior art in light of the statement made by the admitted prior art that its hypermedia system is designed to handle hypermedia documents according the HTML markup standard. **(See USP '906: Col. 5, lines 28-31).**

Declaration of Edward W. Felten, at ¶ 27 (Oct. 6, 2004) (accompanying Applicants' Response (Oct. 12, 2004)) (831 PH Ex. 13 at PH_001_0000785581): For example, the "hypermedia browser" of the '906 claims ***must parse*** hyperlinks ***from within a text document***, but Toye does not provide that feature.

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 51 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785959):

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 Viola 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.

iii. Interference 105,563 McK

Doyle Annotated Copy of Claims, at 2-3 (July 3, 2007) (563 PH Ex. 02 at PH_001_0000787571-72):

1. A method for running an application program { **Fig. 5, item 210** } in a computer network environment { **Fig. 5, item 206** }, comprising:

providing at least one client workstation { **Fig. 5, item 200** } and one network server { **Fig. 5, item 204** } coupled to said network environment { **Fig. 5, item 206** }, wherein said network environment { **Fig. 5, item 206** } is a distributed hypermedia environment { **Fig. 2, item 100** };

executing, at said client workstation { **Fig. 5, item 200** }, a browser application { **Fig. 5, item 208** }, that parses { **Fig. 7A, step 256** } a first distributed hypermedia document { **Fig. 5, item 212** } to identify text formats { **Fig. 5, item 214** } included in said distributed hypermedia document { **Fig. 5, item 212** } and for responding to predetermined texr formats { **Fig. 5, item 214** } to initiate processing specified by said text formats { **Fig. 5, item 214** }; utilizing said browser { **Fig. 5, item 208** } to display, on said client workstation { **Fig. 5, item 200** }, at least a portion of a first hypermedia document { **Fig. 5, item 212** } received over said network { **Fig. 5, item 206** } from said server { **Fig. 5, item 204** }, wherein the portion of said first hypermedia document { **Fig. 5, item 212** } is displayed within a first browser-controlled window { **Fig. 9, item 350** } on said client workstation { **Fig. 5, item 200** }, wherein said first distributed hypermedia document { **Fig. 5, item 212** } includes an embed text format { **Fig. 5, item 214** }, located at a first location in said first distributed hypermedia document { **Fig. 5, item 212** }, that specifies the location of at least a portion of an object { **Fig. 5, item 216** } external to the first distributed hypermedia document { **Fig. 5, item 212** }, wherein said object { **Fig. 5, item 216** } has type information associated with it utilized by said browser { **Fig. 5, item 208** } to identify and locate an executable application { **Fig. 5, item 210** } external to the first distributed hypermedia document { **Fig. 5, item 212** }, and wherein said embed text format { **Fig. 5, item 214** } is parsed { **Fig. 7A, step 256** } by said browser { **Fig. 5, item 208** } to automatically invoke { **Fig. 8A, step 290**} said executable application { **Fig. 5, item 210** } to execute on said client workstation { **Fig. 5, item 200** } in order to display said object { **Fig. 5, item 216** } and enable interactive processing of said object { **Fig. 5, item 216** } within a display area created at said first location within the portion of said first distributed hypermedia document { **Fig. 5, item 212** } being displayed in said first browser-controlled window { **Fig. 9, item 350** }.

iv. '985 prosecution history (10/217,955)

Applicants' Response, at 9 (March 11, 2005) (985 PH Ex. 03 at PH_001_0000784221):

> The specification of the '906 patent (Applicants' Admitted Prior Art) describes a browser application, e.g., Mosaic, that functions as a viewer to view HTML documents. There are several ways to retrieve an HTML document from a network server, all of which require user interaction with the browser. [Felten I, paragraph 8]. The browser then retrieves a selected published source HTML document from a network server by utilizing a uniform resource locator (URL) that locates the HTML document on the network and stores a temporary local copy of the HTML source document in a cache on the client workstation.
>
> The browser application then parses the local copy of the HTML document, renders the temporary local copy of the HTML document into a Web page , and displays the rendered Web page in a browser-controlled window. [Felten I, at paragraph 21]. During the rendering step, the browser may retrieve information external to the local copy of the HTML document, such as source files referenced by IMG tags, render the images from the retrieved files as static graphic images, and insert the images into the Web page of the HTML document, for display to the user.

Applicants' Response, at 17 (April 11, 2008) (985 PH Ex. 11 at PH_001_0000784584): "The other parts of the Confirmation recite other findings supporting the determination that the reexamination claims are not unpatentable over the cited references and these findings also support a determination that the pending claims of the present application are not unpatentable over the cited references. A complete copy of the Confirmation is appended to this response."

Response to Office Action, at 22 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784614, 33-34):

> "***responding to text formats*** *to initiate processing specified by the text formats*;
>
> **EXAMPLE SUPPORT**:
>
> **9:24** 'Once hypermedia document 212 has been loaded into client computer 200, browser client 208 parses hypermedia document 212. In parsing hypermedia document 212, browser client 208 detects links to data objects as discussed above in the Background of the Invention section.'
>
> …
>
> ***identifying an embed text format*** *which corresponds to a first location in the document,*
>
> **EXAMPLE SUPPORT**:
>
> **14:27** 'a check is made as to whether the current tag is the EMBED tag.'"

Notice of Allowability, at 2 (March 20, 2009) (985 PH Ex. 15 at PH_001_0000784733):

The following is an examiner's statement of reasons for allowance: the claims are allowable as the claims contain the subject matter deemed allowable in both Re exam 90/006,831 and Re exam 90/007,838 for the same reasons as set forth in the NIRC of the two Re exams.

2. Defendants' extrinsic evidence

Que's Computer Programmer's Dictionary 302 (1993) ("parse") [Ex. V at PA-0000333392]: "To decompose an expression and categorize its components. The term can apply to natural languages such as English to programming languages and to any other structured input data For example compiler usually begins by parsing the source code."

Barron's Dictionary of Computer Terms 230 (2d ed. 1989) ("parsing") [Ex. Q at PA-0000333377]: "Parsing is the analysis, by computer, of the structure of statements in a human or artificial language."

G. "object"

Claim Term(s)	Defendants' Proposed Construction	Eolas's Proposed Construction
object	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 or other types of information that is presentable to a user of a computer system

1. Defendants' intrinsic evidence

a. Claims

In the following chart, the term(s) for construction appear in all the claims marked with an "x." The letter "m" indicates a method claim, and "¶6" indicates a claim that the Defendants contend includes limitations subject to § 112, ¶ 6.

	'906 patent						'985 patent								
	m 1	m 4	m 5	¶6 6	¶6 9	¶6 10	m 1	¶6 16	m ¶6 20	m ¶6 24	¶6 28	m ¶6 32	m 36	m ¶6 40	m 44
object	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

b. Specification (all cites to '906 patent)

(Abstract): A system allowing a user of a browser program on a computer connected to an open distributed hypermedia system to access and execute an embedded ***program object***. The ***program object*** is embedded into a hypermedia document much like ***data objects***. The user may select the ***program object*** from the screen. Once selected the ***program object*** executes on the user's (client) computer or may execute on a remote server or additional remote computers in a distributed processing arrangement. After launching the ***program object***, the user is able to interact with the ***object*** as the invention provides for ongoing interprocess communication between the ***application object*** (program) and the browser program. One application of the embedded ***program object*** allows

a user to view large and complex ***multi-dimensional objects*** from within the browser's window. The user can manipulate a control panel to change the viewpoint used to view the image. The invention allows a program to execute on a remote server or other computers to calculate the viewing transformations and send frame data to the client computer thus providing the user of the client computer with interactive features and allowing the user to have access to greater computing power than may be available at the user's client computer.

1:20–:23 (Background of the Invention): This invention relates generally to manipulating data in a computer network, and specifically to retrieving, presenting and manipulating embedded program objects in distributed hypermedia systems.

1:24–:45 (Background of the Invention): Computer networks are becoming increasingly popular as a medium for locating and accessing a wide range of data from locations all over the world. The most popular global network is the Internet with millions of computer systems connected to it. The Internet has become popular due to widely adopted standard protocols that allow a vast interconnection of computers and localized computer networks to communicate with each other. Computer systems connected to a network such as the Internet may be of varying types, e.g., mainframes, workstations, personal computers, etc. The computers are manufactured by different companies using proprietary hardware and operating systems and thus have incompatibilities in their instruction sets, busses, software, file formats and other aspects of their architecture and operating systems. Localized computer networks connected to the Internet may be incompatible with other computer systems and localized networks in terms of the physical layer of communication including the specific hardware used to implement the network. Also, different networks use differing, incompatible protocols for transferring information and are not able to communicate with each other without a translation mechanism such as a "gateway".

1:61–2:6 (Background of the Invention): A hypertext document is a document that allows a user to view a text document displayed on a display device connected to the user's computer and to access, retrieve and view other ***data objects*** that are linked to hypertext words or phrases in the hypertext document. In a hypertext document, the user may "click on," or select, certain words or phrases in the text that specify a link to other documents, or ***data objects***. In this way, the user is able to navigate easily among ***data objects***. The ***data objects*** may be local to the user's computer system or remotely located over a network. An early hypertext system is Hypercard, by Apple Computer, Inc. Hypercard is a standalone system where the ***data objects*** are local to the user's system.

2:14–:27 (Background of the Invention): ***Objects*** may be text, images, sound files, video data, documents or other types of information that is presentable to a user of a computer system. When a document is primarily text and includes links to other ***data objects*** according to the hypertext format, the document is said to be a hypertext document. When graphics, sound, video or other media capable of being manipulated and presented in a computer system is used as the ***object*** linked to, the document is said to be a hypermedia document. A hypermedia document is similar to a hypertext document, except that the user is able 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.

3:27–:32 (Background of the Invention): Returning to FIG. 1, another type of **data object** is a ***sound object*** shown as sound icon 24 within the hypermedia document. When the user selects sound icon 24, the user's computer accesses sound data shown symbolically by data file 40. The accessed sound data plays through a speaker or other audio device. (See also Fig. 1.)

3:34–:50 (Background of the Invention): As discussed above, hypermedia documents allow a user to access different ***data objects***. The ***objects*** may be text, images, sound files, video, additional documents, etc. As used in this specification, a ***data object*** is information capable of being retrieved and presented to a user of a computer system. Some ***data objects*** include executable code combined with data. An example of such a combination is a "self-extracting" ***data object*** that includes code to

"unpack" or decompress data that has been compressed to make it smaller before transferring. When a browser retrieves an **object** such as a self-extracting ***data object*** the browser may allow the user to "launch" the self-extracting ***data object*** to automatically execute the unpacking instructions to expand the ***data object*** to its original size. Such a combination of executable code and data is limited in that the user can do no more than invoke the code to perform a singular function such as performing the self-extraction after which time the ***object is a standard data object***.

3:51–:59 (Background of the Invention): Other existing approaches to embedding interactive ***program objects*** in documents include the Object Linking and Embedding (OLE) facility in Microsoft Windows, by Microsoft Corp., and OpenDoc, by Apple Computer, Inc. At least one shortcoming of these approaches is that neither is capable of allowing a user to access embedded interactive ***program objects*** in distributed hypermedia documents over networks.

5:14–:23 (Background of the Invention): Similarly, ***image object*** 16 and sound data file 40 may reside at any of the computers shown in FIG. 2. Assuming ***image object*** 16 resides on server C when user 110 clicks on image icon 22, client computer 108 generates a command to retrieve ***image object*** 16 to server C. Server C receives the command and transfers a copy of ***image object*** 16 to client computer 108. Alternatively, an ***object***, such as sound data file 40, may reside on server A so that it is not necessary to traverse long distances via the Internet in order to retrieve the ***data object***.

5:24–:38 (Background of the Invention): The Internet is said to provide an "open distributed hypermedia system." It is an "open" system since Internet 100 implements a standard protocol that each of the connecting computer systems, 106, 130, 120, 132 and 134 must implement (TCP/IP). It is a "hypermedia" system because it is able to handle hypermedia documents as described above via standards such as the HTTP and HTML hypertext transmission and mark up standards, respectively. Further, it is a "distributed" system because ***data objects*** that are imbedded within a document may be located on many of the computer systems connected to the Internet. An example of an open distributed hypermedia system is the so-called "world-wide web" implemented on the Internet and discussed in papers such as the Berners-Lee reference given above.

5:39–:56 (Background of the Invention): The open distributed hypermedia system provided by the Internet allows users to easily access and retrieve different ***data objects*** located in remote geographic locations on the Internet. However, this open distributed hypermedia system as it currently exists has shortcomings in that today's large ***data objects*** are limited largely by bandwidth constraints in the various communication links in the Internet and localized networks, and by the limited processing power, or computing constraints, of small computer systems normally provided to most users. Large ***data objects*** are difficult to update at frame rates fast enough (e.g., 30 frames per second) to achieve smooth animation. Moreover, the processing power needed to perform the calculations to animate such images in real time does not exist on most workstations, not to mention personal computers. Today's browsers and viewers are not capable of performing the computation necessary to generate and render new views of these large ***data objects*** in real time.

6:26–:39 (Background of the Invention): Due to the relatively low bandwidth of the Internet (as compared to today's large ***data objects***) and the relatively small amount of processing power available at client computers, many valuable tasks performed by computers cannot be performed by users at client computers on the Internet. Also, while the present open distributed hypermedia system on the Internet allows users to locate and retrieve ***data objects*** it allows users very little, if any, interaction with these ***data objects***. Users are limited to traditional hypertext and hypermedia forms of selecting linked ***data objects*** for retrieval and launching viewers or other forms of external software to have the ***data objects*** presented in a comprehensible way.

6:40–:47 (Background of the Invention): Thus, it is desirable to have a system that allows a user at a small client computer connected to the Internet to locate, retrieve and manipulate ***data objects*** when the ***data objects*** are bandwidth-intensive and compute-intensive. Further, it is desirable

to allow a user to manipulate ***data objects*** in an interactive way to provide the user with a better understanding of information presented and to allow the user to accomplish a wider variety of tasks.

6:50–:62 (Summary of the Invention): The present invention provides a method for running embedded ***program objects*** in a computer network environment. The method includes the steps of providing at least one client workstation and one network server coupled to the network environment where the network environment is a distributed hypermedia environment; displaying, on the client workstation, a portion of a hypermedia document received over the network from the server, where the hypermedia document includes an embedded controllable application; and interactively controlling the embedded controllable application from the client workstation via communication sent over the distributed hypermedia environment.

6:63–7:6 (Summary of the Invention): The present invention allows a user at a client computer connected to a network to locate, retrieve and manipulate ***objects*** in an interactive way. The invention not only allows the user to use a hypermedia format to locate and retrieve ***program objects***, but also allows the user to interact with an application program located at a remote computer. Interprocess communication between the hypermedia browser and the embedded application program is ongoing after the ***program object*** has been launched. The user is able to use a vast amount of computing power beyond that which is contained in the user's client computer.

9:24–:39 (Detailed Description of a Preferred Embodiment): Once hypermedia document 212 has been loaded into client computer 200, browser client 208 parses hypermedia document 212. In parsing hypermedia document 212, browser client 208 detects links to ***data objects*** as discussed above in the Background of the Invention section. In FIG. 5, hypermedia document 212 includes an embedded program link at 214. Embedded program link 214 identifies application client 212 as an application to invoke. In this present example, the application, namely, application client 210, resides on the same computer as the browser client 208 that the user is executing to view the hypermedia document. Embedded program link 214 may include additional information, such as parameters, that tell application client 210 how to proceed. For example, embedded program link 214 may include a specification as to a ***data object*** that application client 210 is to retrieve and process.

9:46–:58 (Detailed Description of a Preferred Embodiment): An example of the type of processing that application client 210 may perform is multidimensional image visualization. Note that application client 210 is in communication with network 206 via the network protocol layer of client computer 200. This means that application client 210 can make requests over network 206 for ***data objects***, such as multidimensional ***image objects***. For example, application client 210 may request an ***object***, such as ***object* 1** at 216, located in server computer 204. Application client 210 may make the request by any suitable means. Assuming network 206 is the Internet, such a request would typically be made by using HTTP in response to a HTML-style link definition for embedded program link 214.

9:59–:65 (Detailed Description of a Preferred Embodiment): Assuming application client 210 has made a request for the ***data object*** at 216, server process 218 ultimately receives the request. Server process 218 then retrieves ***data object*** 216 and transfers it over network 206 back to application client 210. To continue with the example of a multidimensional visualization application, ***data object*** 216 may be a three dimensional view of medical data for, e.g., an embryo.

11:52–12:8 (Detailed Description of a Preferred Embodiment): Another type of possible application of this invention would involve embedding a program which runs only on the client machine, but which provides the user with more functionality than exists in the hypermedia browser alone. An example of this is an embedded client application which is capable of viewing and interacting with images which have been processed with Dr. Doyle's MetaMAP invention (U.S. Pat. No. 4,847,604). This MetaMAP process uses object-oriented color map processing to allow individual color index ranges within paletted images to have ***object identities***, and is useful for the

creation of, for example, interactive picture atlases. It is a more efficient means for defining irregular "hotspots" on images than the ISMAP function of the World Wide Web, which uses polygonal outlines to define ***objects in images***. A MetaMAP-capable client-based image browser application can be embedded, together with an associated image, within a hypermedia document, allowing ***objects*** within the MetaMAP-processed image to have URL addresses associated with them. When a user clicks with a mouse upon an **object** within the MetaMAP-processed image, the MetaMAP client application relays the relevant URL back to the hypermedia browser application, which then retrieves the HTML file or ***hypermedia object*** which corresponds to that URL.

12:66–13:18 (Detailed Description of a Preferred Embodiment): As shown in Table II, the EMBED tag includes TYPE, HREF, WIDTH and HEIGHT elements. The 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. In the case where TYPE is "application/x-vis" this means that the ***object*** at the URL address is a three dimensional ***image object*** since the program "x-vis" is a data visualization tool designed to operate on three dimensional ***image objects***. However, any manner of application program may be specified by the TYPE element so that other types of applications, such as a spreadsheet program, database program, word processor, etc. may be used with the present invention. Accordingly, the **object** reference by the HREF element would be, respectively, a ***spreadsheet object, database object, word processor document object***, etc.

13:19–:31 (Detailed Description of a Preferred Embodiment): WIDTH and HEIGHT elements specify the width and height dimensions, respectively, of a Distributed Hypermedia Object Embedding (DHOE) window to display an external ***application object*** such as the three dimensional ***image object*** or ***video object*** discussed above.

13:32–:36 (Detailed Description of a Preferred Embodiment): WIDTH and HEIGHT elements specify the width and height dimensions, respectively, of a Distributed Hypermedia Object Embedding (DHOE) window to display an external ***application object*** such as the three dimensional ***image object*** or ***video object*** discussed above.

14:64–:67 (Detailed Description of a Preferred Embodiment): FIG. 8A is a flowchart for routine HTMLwidget. HTMLwidget creates display data structures and launches an external application program to handle the ***data object*** specified by the URL in the EMBED tag. (See also Fig. 8A.)

15:39–:48 (Detailed Description of a Preferred Embodiment): If, at step 286, the type is determined not to be an ***application object*** (e.g., a three dimensional ***image object*** in the case of application "x-vis") a check is made at step 288 to determine if the type is a ***video object***. If so, step 292 is executed to launch a video player application. Parameters are passed to the video player application to allow the player to display the ***video object*** within the DrawingArea within the display of the portion of hypermedia document on the client's computer. Note that many other ***application objects*** types are possible as described above.

15:58–16:8 (Detailed Description of a Preferred Embodiment): The present invention allows a user to have interactive control over ***application objects*** such as three dimensional ***image objects*** and ***video objects***. In a preferred embodiment, controls are provided on the external applications' user interface. In the case of a VIS/panel application, a process, "panel" creates a graphical user interface (GUI) thru which the user interacts with the data. The application program, VIS, can be executing locally with the user's computer or remotely on a server, or on one or more different computers, on the network. The application program updates pixmap data and transfers the pixmap data (frame image data) to a buffer to which the browser has access. The browser only needs to respond to the refresh request to copy the contents from the updated pixmap to the DrawingArea. The Panel process

sends messages as "Msg" sending performed by routines such as vis_-send_-msg and vis_-handle panel_-msg to send events (mousemove, keypress, etc.) to the external application.

16:9–:28 (Detailed Description of a Preferred Embodiment): FIG. 9 is a screen display of the invention showing an interactive ***application object*** (in this case a three dimensional ***image object***) in a window within a browser window. In FIG. 9, the browser is NCSA Mosaic version 2.4. The processes VIS, Panel and VRServer work as discussed above. FIG. 9 shows screen display 356 Mosaic window 350 containing image window 352 and a portion of a panel window 354. Note that image window 352 is within Mosaic window 350 while panel window 354 is external to Mosaic window 350. Another possibility is to have panel window 354 within Mosaic window 350. By using the controls in panel window 354 the user is able to manipulate the image within image window 352 in real time do perform such operations as scaling, rotation, translation, color map selection, etc. In FIG. 9, two Mosaic windows are being used to show two different views of an embryo image. One of the views is rotated by six degrees from the other view so that a stereoscopic effect can be achieved when viewing the images. Communication between Panel and VIS is via "Tooltalk" described in, e.g., "Tooltalk 1.1.1 Reference Manual," from SunSoft. (See also Figs. 9-10.)

c. Prosecution history

i. '906 prosecution history (08/324,443)

Original Application, at 29 (Oct. 17, 1994) (906 PH Ex. 01 at PH_001_0000783829):

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;

displaying, on said client workstation, at least a portion of a hypermedia document received over said network from said server, wherein said hypermedia document includes an embedded controllable application; and

interactively controlling said embedded controllable application from said client workstation via communications sent over said distributed hypermedia environment.

Original Application, at 29 (Oct. 17, 1994) (906 PH Ex. 01 at PH_001_0000783829):

3. The method of claim 2, wherein instructions for controlling said embedded controllable application reside on said network server, wherein said step of interactively controlling said embedded controllable application includes the following substeps:

issuing, from the client workstation, one or more commands to the network server;

executing, on the network server, one or more instructions in response to said commands;

sending information from said network server to said client workstation in response to said executed instructions; and

processing said information at the client workstation to interactively control said embedded controllable application.

Original Application, at 30 (Oct. 17, 1994) (906 PH Ex. 01 at PH_001_0000783830):

6. The method of claim 3, wherein said embedded controllable application is a multi-dimensional viewer.

7. The method of claim 3, wherein said embedded controllable application is a spreadsheet program.

8. The method of claim 3, wherein said embedded controllable application is a database program.

9. The method of claim 3, wherein said embedded controllable application is a word processor.

10. The method of claim 3, wherein said substeps of issuing and sending are via an open protocol.

Original Application, at 31 (Oct. 17, 1994) (906 PH Ex. 01 at PH_001_0000783831):

15. 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, said network including a plurality of general purpose workstations, wherein said network environment is a distributed hypermedia environment;

displaying, on said client workstation, at least a portion of a hypermedia document received over said network from said server, wherein said hypermedia document includes at least a first embedded multi-dimensional data visualization application; and

interactively controlling said embedded multi-dimensional data visualization application from said client workstation via communications sent over said distributed hypermedia environment wherein data image rendering is performed by said plurality of general purpose workstations using distributed processing.

Originial Application, at 31 (Oct. 17, 1994) (906 PH Ex. 01 at PH_001_0000783831):

16. The method of claim 15, wherein the step of displaying is performed by using a hypermedia browser application.

17. The method of claim 15, wherein the multi-dimensional data visualization includes volume visualization.

18. The method of claim 15, wherein the multi-dimensional data visualization includes two dimensional image processing.

19. The method of claim 15, wherein the multi-dimensional data visualization includes image analysis.

Original Application, at 32 (Oct. 17, 1994) (906 PH Ex. 01 at PH_001_0000783832):

20. The method of claim 15, wherein the multi-dimensional data visualization includes the display of animated sequences.

21. The method of claim 15, wherein the multi-dimensional data visualization includes a geometric data viewer to display computer aided design files.

22. The method of claim 15, wherein the multi-dimensional data visualization includes displaying molecular modeling data.

Original Application, at 32-33 (Oct. 17, 1994) (906 PH Ex. 01 at PH_001_0000783832-33):

24. A method for interactively controlling an embedded object in a document displayed on a client computer, wherein the client computer includes a processor coupled to a display device and to a user input device, wherein the processor is further coupled to a computer network, wherein the computer network is coupled to a server computer and one or more additional computers, wherein the server computer includes a local storage device containing a document, wherein the document includes an embedded object, wherein an application program for manipulating the embedded object resides on a first additional computer, the method comprising the following steps:

transferring, over the network, at least a portion of the document from the server computer to the client computer;

accepting first signals from the user input device that indicate that the embedded object is to be manipulated;

issuing commands from the client computer to the first additional computer in response to the first signals;

executing, by using the first additional computer, instructions in the application program in response to the issued commands, wherein the executed instructions generate information about manipulating the embedded object;

communicating, via the network, the information about manipulating the embedded object from the first additional computer to the client computer; and

using the client computer to manipulate the embedded object according to the communicated information.

Original Application, at 34-35 (Oct. 17, 1994) (906 PH Ex. 01 at PH_001_0000783834-35):

34. A method for displaying a three dimensional image object on a client computer, wherein the client computer includes a processor coupled to a display device, wherein the processor is further coupled to a computer network, wherein the computer network is coupled to a server computer and one or more additional computers, wherein the server computer includes a local storage device containing a hypermedia document, wherein the hypermedia document includes a three dimensional image object embedded within the hypermedia document, wherein the three dimensional image object is displayable in a plurality of orientations, the method comprising the following steps:

transferring, over the network, at least a portion of the hypermedia document from the server computer to the client computer;

displaying on the display device, by using the processor, at least a portion of the hypermedia document, wherein the displayed portion of the hypermedia document includes the three dimensional image object displayed in a first orientation;

using the client computer to issue commands over the network;

executing instruction on a first additional computer in response to the issued commands, wherein the executed instructions determine a second orientation for display of the three dimensional image object;

communicating, via the network, information about the second orientation from the first additional computer to the client computer; and

using the client computer to redisplay the three dimensional image object in the second orientation.

(*See also* Original Application, at 29-36 (Oct. 17, 1994) (906 PH Ex. 01 at PH_001_0000783829).)

Amendment A, at 15 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783893):

> There is no disclosure of the claimed step of 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. As described above, the Mercury Project utilizes CGI where a <FORM> tag identifies a program on the server but not an external object. Additionally, the claimed step of invoking the executable object to display and process the object within the first browser-controlled window while a portion of the first hypermedia document is displayed is not disclosed.

Amendment A, at 16 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783894):

> The claimed combination is fundamentally different from the Mercury Project. In the claimed combination, the external object and executable object are embedded by reference in the HTML document and the object is displayed and processed within the same window where a portion of the original document is displayed. In the Mercury Project information is passed back to the server and a new document is generated and displayed. There is no display and processing the external object within the window in which a portion of the original document is displayed.

Amendment A, at 16 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783894):

A major difference between CGI and the claimed combination is that in CGI there is no enforced continuity between the documents. The CGI responds to form information by generating new documents each being a static document independent of the previous document which generated the form string passed to the Web server. For example, in the Mercury Project separate, independent HTML documents are generated for each position of the arm. There is no disclosure of the claimed step of invoking the executable application to display and process said object within the window while a portion of the first distributed hypermedia document is displayed in the window.

Thus, 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.

ii. Abandoned application (09/075,359)

Original Application, at 29 (May 8, 1998) (359 PH Ex. 01 at PH_001_0000787729):

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;

displaying, on said client workstation, at least a portion of a hypermedia document received over said network from said server, wherein said hypermedia document includes an embedded controllable application; and

interactively controlling said embedded controllable application from said client workstation via communications sent over said distributed hypermedia environment.

Original Application, at 29 (May 8, 1998) (359 PH Ex. 01 at PH_001_0000787729):

3. The method of claim 2, wherein instructions for controlling said embedded controllable application reside on said network server, wherein said step of interactively controlling said embedded controllable application includes the following substeps:

issuing, from the client workstation, one or more commands to the network server;

executing, on the network server, one or more instructions in response to said commands;

sending information from said network server to said client workstation in response to said executed instructions; and

processing said information at the client workstation to interactively control said embedded controllable application.

Original Application, at 30 (May 8, 1998) (359 PH Ex. 01 at PH_001_0000787730):

6. The method of claim 3, wherein said embedded controllable application is a multi-dimensional viewer.

7. The method of claim 3, wherein said embedded controllable application is a spreadsheet program.

8. The method of claim 3, wherein said embedded controllable application is a database program.

9. The method of claim 3, wherein said embedded controllable application is a word processor.

10. The method of claim 3, wherein said substeps of issuing and sending are via an open protocol.

Original Application, at 31 (May 8, 1998) (359 PH Ex. 01 at PH_001_0000787731):

15. 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, said network including a plurality of general purpose workstations, wherein said network environment is a distributed hypermedia environment;

displaying, on said client workstation, at least a portion of a hypermedia document received over said network from said server, wherein said hypermedia document includes at least a first embedded multi-dimensional data visualization application; and

interactively controlling said embedded multi-dimensional data visualization application from said client workstation via communications sent over said distributed hypermedia environment wherein data image rendering is performed by said plurality of general purpose workstations using distributed processing.

Original Application, at 31 (May 8, 1998) (359 PH Ex. 01 at PH_001_0000787731):

16. The method of claim 15, wherein the step of displaying is performed by using a hypermedia browser application.

17. The method of claim 15, wherein the multi-dimensional data visualization includes volume visualization.

18. The method of claim 15, wherein the multi-dimensional data visualization includes two dimensional image processing.

19. The method of claim 15, wherein the multi-dimensional data visualization includes image analysis.

Original Application, at 32 (May 8, 1998) (359 PH Ex. 1 at PH_001_0000787732):

20. The method of claim 15, wherein the multi-dimensional data visualization includes the display of animated sequences.

21. The method of claim 15, wherein the multi-dimensional data visualization includes a geometric data viewer to display computer aided design files.

22. The method of claim 15, wherein the multi-dimensional data visualization includes displaying molecular modeling data.

(*See also* Original Application, at 29-36 (May 8, 1998) ('359 PH Ex. 01 at PH_001_0000787729 – PH_001_0000787736).)

Preliminary Amendment, at 3 (May 8, 1998) (359 PH Ex. 02 at PH_001_0000787772):

50. (New) The method of claim 6, wherein said executable application is a computer program which runs other computer programs.

51. (New) The method of claim 7, wherein said other computer programs are transferred to said computer system over a computer network environment.

Preliminary Amendment, at 7 (May 8, 1998) (359 PH Ex. 02 at PH_001_0000787776):

59 ~~58~~. (New) The method of claim 15, wherein said executable application is a computer program which runs other computer programs.

60. ~~59~~. (New) The method of claim 16, wherein said other computer programs are transferred to said computer system over a computer network environment.

(*See also* Preliminary Amendment, at 2-7 (May 8, 1998) (359 PH Ex. 02 at (PH_001_0000787771-76).)

Office Action, at 4 (Sept. 6, 2000) (359 PH Ex. 03 at PH_001_0000787797):

As per claims 50 and 59, there is no disclosure in the specification that the "executable application is a computer program which runs other programs".

Office Action, at 4 (Sept. 6, 2000) (359 PH Ex. 03 at PH_001_0000787797):

As per claims 51 and 60, there is no disclosure in the specification of transferring of the other computer program over a computer network environment.

The disclosure specifically disclose that the external application is preinstalled on the client. There is no disclosure of the external application invoking another program nor retrieving the other program over the network.

iii. First reexam (90/006,831)

Applicants' Response, at 16 (Oct. 12, 2004) (831 PH Ex. 16 at PH_001_0000785818):

A distributed hypermedia system is a "distributed" system because data objects that are imbedded within a document may be located on many of the computer systems connected to the Internet." ['906 at col. 5, lines 25-38].

Notice of Intent to Issue Ex Parte Reexamination Certificate, at 52-54 (Sept. 27, 2005) (831 PH Ex. 19 at PH_001_0000785960-61):

II. VIOLA SCRIPTS (OR CORRESPONDING BYTE-CODE FORMS) DO NOT ANTICIPATE NOR FAIRLY SUGGEST THE EXTERNAL "OBJECT" AS CLAIMED IN THE '906 PATENT.

If the Viola <VOBJF> tags are considered as arguably corresponding to the instant claimed '906 "embed text format" (in the sense that the Viola <VOBJF> tags specify "the location of at least a portion of an object external to the first distributed hypermedia document" as claimed in '906 claims 1 and 6), then the Viola script program specified between the <VOBJF> tags is not equivalent to the instant '906 claimed external "object" when the

claimed '906 external "object" is interpreted in a manner consistent with the specification of the '906 patent.

The Viola, "clock.v" script is a high-level source code PROGRAM. In contrast, the scope of the claimed '906 external "object" broadly encompasses myriad types of data objects, including self-extracting data objects [see '906 patent, col. 3, lines 33-51].

The scope of the claimed '906 external "object" is broad when construed in a manner consistent with the specification (i.e., see '906 patent, col. 3, lines 36-39: "a data object is information capable of being retrieved and presented to a user of a computer system."). However, the scope of the claimed '906 external "object" clearly does not read upon a high-level source code PROGRAM, such as a Viola script, nor does it read upon an object in byte-code form.

When the scope of the claimed '906 external "object" is construed in a manner consistent with the specification, it is clear that any executable component of the claimed '906 external data "object" is limited to performing self-extraction of the compressed data object:

See '906 patent, col. 3, lines 43-51:

When a browser retrieves an object such as a self-extracting **data object** the browser may allow the user to "launch" the self-extracting **data object** to automatically execute the unpacking instructions to expand the data object to its original size. Such a combination of executable code and data is limited in that the user can do no more than invoke the code to perform a singular function such as performing the self-extraction after which time the object is a standard data object.

Although a self-extracting data object typically includes executable code to expand the compressed data object to its original size, this type of self-extraction extracts DATA that has no relationship to a high-level source code PROGRAM in the form of a Viola script, or a byte-code file, or the like.

(*See also* Notice of Intent to Issue Supplemental Ex Parte Reexamination Certificate, at 52-54 (Jan. 20, 2006) (831 PH Ex. 20 at PH_001_0000786047).)

iv. Second reexam (90/007,858)

Applicants' Response, at 20 (Sept. 27, 2007) (858 PH Ex. 05 at PH_001_0000787049):

Additionally, the design of Cohen is inconsistent with the use of type information recited in claim 6. The identify and locate step, and the fact that that step is done by the browser, is an important aspect of the claimed '906 invention. For example, this step provides an important security protection. Users often want to display distributed hypermedia documents that come from untrusted sources, such as Web pages that come from arbitrary sites. If the author of such a site can cause an executable application of his choice to be invoked on the user's system, then the site author can use that application to gain access to the user's private files or modify the state of the user's computer, for example to install spyware or a virus. [Felten at paragraph 65]

Having the browser – a program trusted by the user – identify and locate the executable application lets the browser protect the user from this danger. A properly written browser will only allow trusted applications to be run, thereby protecting the user against security problems. A hostile site author cannot run a malicious application on the user's computer, because it is the browser, not the site author, that is identifying and locating the application that will be run. References in which the browser does not utilize type information to identify and locate the executable application lack this protection. [Felten at paragraph 66]

Declaration of Edward W. Felten, at 10 (Sept. 27, 2007) (accompanying Applicants' Response (Sept. 27, 2007)) (858 PH Ex. 06 at PH_001_0000787061):

64. In Cohen, the book reader (which the Examiner equates to a browser) does not utilize the "type information" to identify and locate anything. All the book reader does with this information is to pass it on, unexamined, to the operating system, which invokes the application. The book reader does not have any kind of algorithm or procedure that it follows to identify and locate an application to be used.

65. The identify and locate step, and the fact that that step is done by the browser, is an important aspect of the invention defined in Claim 6 of the '906 Patent. For example, this step provides an important security protection. Users often want to display distributed hypermedia documents that come from untrusted sources, such as Web pages that come from arbitrary sites. If the author of such a site can cause an executable application of his choice to be invoked on the user's system, then the site author can use that application to gain access to the user's private files or modify the state of the user's computer, for example to install spyware or a virus.

66. Having the browser – a program trusted by the user – identify and locate the executable application lets the browser protect the user from this danger. A properly written browser will only allow trusted applications to be run, thereby protecting the user against security problems. A hostile site author cannot run a malicious application on the user's computer, because it is the browser, not the site author, that is identifying and locating the application that will be run. References in which the browser does not utilize type information to identify and locate the executable application lack this protection.

67. The importance of this claim element was reinforced by the *Eolas v. Microsoft* litigation, in which the question of which references had this claim element was an important one.

d. Cited references

Eolas, et al. v. Microsoft, Corp., No. 99-C-626 (N.D.Ill. 2003) Jury Instructions (Aug. 7, 2003) (Ex. P at PH_001_0000622922):

Jury Instructions

One: Executable application. A key part of the invention is the ability of the browser to automatically invoke some external program vis-a-vis the hypermedia document to process the data object. Generic examples of such programs are image viewers, word processors and spread sheets, programs that display and allow the user to interact with data. Such programs are referred to in the claims as an executable application.

I have defined executable application as 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.

Eolas, et al. v. Microsoft, Corp., No. 99-C-626 (N.D.Ill. 2003) Jury Instructions (Aug. 7, 2003) (Ex. P at PH_001_0000622922):

The word external means external to a web page. Executable application may be a stand-alone program, a component such as a dynamic link library, or multiple components working together. A component may be used to perform more than one function and may be used by more than one program.

2. <u>Defendants' extrinsic evidence</u>

"FYI… press release", at 1 (Aug. 30, 1994) [Ex. AA at PA-0000333362]:

Michael D. Doyle (*mddoyle@netcom.com*)
Tue, 30 Aug 1994 23:15:10 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Dave Raggett: "Re: Results of the VRML Survey"
- **Previous message:** Joe Andrieu: "Re: Office Space ;) and coordinate systems"
- **Next in thread:** Gavin Nicol: "Re: FYI... press release"

The technology described below may present an alternative strategy for
delivering certain types of VR applications through the Web.

>*PRESS RELEASE - PRESS RELEASE - PRESS RELEASE - PRESS RELEASE - PRESS RELEASE*
>
>| *EMBEDDED PROGRAM OBJECTS IN DISTRIBUTED HYPERMEDIA SYSTEMS*
>|
>| *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.*
>| *The researchers' first application, already ported to an X Window*
>| *environment, enhances NCSA Mosaic -TM- -a client program for the WWW- so*
>| *that the user can interactively control the display of a 3D medical image*
>| *-generated from a gigabyte-sized data set- on a window within a Mosaic*
>| *document. The user can rotate the object at will, zoom in or out, do*
>| *oblique sectioning, highlight hidden parts, alter the color mapping and*
>| *image contrast, all under real-time control. These capabilities represent*
>| *a substantial improvement over current image display applications in*
>| *Mosaic, which are limited to display of static images and non-interactive*
>| *playback modes.*
>| *It is anticipated that many additional applications which require*
>| *advanced real-time large-scale data processing could be developed from*
>| *this technology. Numerous commercial opportunities for software sales -to*
>| *both clients and servers- and database subscription services could arise*
>| *from products based on the UC embedded program objects. A small sampling of*
>| *appropriate applications might include:*
>|

"FYI… press release", at 1 (Aug. 31, 1994) [Ex. BB at PA-0000333364]:

>| *EMBEDDED PROGRAM OBJECTS IN DISTRIBUTED HYPERMEDIA SYSTEMS*

That's just screaming out to virus makers, "Yo! A new, sexy infection
vector!" This technique will have to be treated with EXTREME caution.

-- Ken Jenks, NASA/JSC/SD5, Space Biomedical Research Institute
kjenks@gothamcity.jsc.nasa.gov (713) 483-4368

"FYI… press release", at 1 (Aug. 31, 1994) [Ex. CC at PA-0000333365]:

I wrote:
>>That's just screaming out to virus makers, "Yo! A new, sexy

>>infection vector!" This technique will have to be treated with

>>EXTREME caution.

Mike replied:
>How so? This system uses a remote process server that maps its output

>into the local Web client's document space. A single local module

>handles all remote "program objects." There is no way that a virus

>could enter the system, since no new code is being run on the local

>system.

I misunderstood. I thought the embedded program onjects were being run on the local system.

-- Ken Jenks, NASA/JSC/SD5, Space Biomedical Research Institute
kjenks@gothamcity.jsc.nasa.gov (713) 483-4368

Testimony of inventor Michael Doyle, Trial Tr., *Eolas Techs Inc. v. Microsoft Corp.*, No. 99-C-626 (N.D. Ill. 2003), at 282:6–283:17 (July 9, 2003) [Ex. GG (EOLASTX-E-0000000644)]:

Q So with the '906 invention, if I understand you, the browser is asking what kind of object do I have in the window, is it a game or something else, and where is the code located to run that object?
A That's correct.
MR. PRITIKIN: Objection, leading, Your Honor.
THE COURT: I'll let that one stand.
BY MS. CONLIN:
Q What are we seeing in this next part of the flow chart?
A As we saw earlier, the browser had determined what kind of data it was dealing with. At this point it's reaching a decision point in the program, and it's saying, well, if the data object is of one type, run one kind of program, or run a program to allow the user to work with that data, and if it's a different type, run a specific separate program.

Q At the time you came up with your '906 invention, Dr. Doyle, did the Mosaic browser have this capability to identify and locate?

A No, it didn't.

Now what we're seeing is the browser has gone out and it's coming back with a data set. It's hard to see, but there's actually a tiny little embryo in that arrow. And the browser launches the code with that data, and we can actually interactively control that, and the '906 invention allows for doing that in any way, any way you can interactively control the data object that is being displayed.

Q And was that application for running or interacting with this visible embryo automatically launched then?

A Yes, it was. When the browser pulled up the page that specified that embed text format, the browser then automatically launches the application with the data all seamlessly integrated for the end user, so the end user doesn't have to do what they had to do before, which was, you know, worry about, you know, somebody sending me some data, what kind of program am I going to use to load this data and go and find the program. It's all seamlessly choreographed for the user by the browser.

H. "hypermedia document" / "distributed hypermedia document" / "file containing information"

Claim Term(s)	Defendants' Proposed Construction	Eolas's Proposed Construction
[first] hypermedia document	a document received by the browser that includes hyperlinks 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
[first] distributed hypermedia document		[first] hypermedia document that allows a user to access a remote data object over a network
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	a file containing information received by the browser that includes hyperlinks to graphics, sound, video or other media	the file contains information to allow the browser application to display at least part of a distributed hypermedia document

1. Defendants' intrinsic evidence

a. Claims

In the following chart, the term(s) for construction appear in all the claims marked with an "x." The letter "m" indicates a method claim, and "¶6" indicates a claim that the Defendants contend includes limitations subject to § 112, ¶ 6.

	'906 patent						'985 patent								
	m 1	m 4	m 5	¶6 6	¶6 9	¶6 10	m 1	¶6 16	m ¶6 20	m ¶6 24	¶6 28	m ¶6 32	m 36	m ¶6 40	m 44
[first] hypermedia document	x	x	x	x	x	x	x	x	x	x	x		x	x	x
[first] distributed hypermedia document	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
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							x	x	x	x	x	x	x	x	x

'906-1, -4, -5: 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*** . . . ; 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 ***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 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.

'906-4, -5: 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*** . . . ; 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 . . . .

'906-6: causing said client workstation to execute a browser application to parse a ***first distributed hypermedia document*** to identify text formats included in ***said distributed hypermedia document*** . . . causing said client workstation to utilize 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 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.

'906-9, -10: causing said client workstation to execute a browser application to parse a ***first distributed hypermedia document*** to identify text formats included in ***said distributed hypermedia document*** . . . causing said client workstation to utilize 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;

'985-1: receiving, at the client workstation from the network server over the network environment, at least one ***file containing information to enable a browser application to display at least a portion of a distributed hypermedia document*** within a browser-controlled window . . . displaying at least a portion of ***the document*** . . . automatically invoking the executable application, in response to the identifying of the embed text format, to execute on the client workstation in order to display the object and enable an end-user to directly interact with the object while the object is being displayed within a display area created at the first location within the portion of ***the hypermedia document*** being displayed in the browser-controlled window.

'985-16: receive, at the client workstation from the network server over the network environment, at least one ***file containing information to enable a browser application to display at least a portion of a distributed hypermedia document*** within a browser-controlled window . . . display at least a portion of ***the document*** . . . and enable an end-user to directly interact with the object while the object is being displayed within a display area created at the first location within the portion of ***the hypermedia document*** being displayed in the browser-controlled window.

'985-20: receive, over said network environment from said server, at least one ***file containing information to enable a browser application to display at least a portion of a distributed hypermedia document*** . . . displaying, on said client workstation, at least a portion of ***the document*** . . . enable an end-user to directly interact with the object while the object is being displayed within a display area created at the first location within the portion of ***the hypermedia document*** being displayed in the browser-controlled window.

'985-24: enabling an end-user to directly interact with an object by utilizing said executable application to interactively process said object while the object is being displayed within a display area created at a first location within a portion of ***a hypermedia document*** being displayed in a browser-controlled window, … wherein said client workstation receives, over said network environment from said server, at least one ***file containing information to enable said browser application to display, on said client workstation, at least said portion of said distributed hypermedia document*** within said browser-controlled window . . . wherein at least said portion of ***the document*** is displayed within the browser-controlled window

'985-28: cause the client workstation to display an object and enable an end-user to directly interact with said object while the object is being displayed within a display area created at a first location within a portion of ***a hypermedia document*** being displayed in a browser-controlled window, … wherein said client workstation receives, over said network environment from said server, at least one ***file containing information to enable said browser application to display, on said client workstation, at least said portion of said distributed hypermedia document*** within said browser-controlled window . . . wherein at least said portion of ***the document*** is displayed within the browser-controlled window

'985-32: receive at said client workstation, over said computer network environment from said server, at least one ***file containing information to enable a browser application to display, on said client workstation, at least a portion of a distributed hypermedia document*** within a browser-controlled window; utilize an executable application external to said file to enable an end-user to directly interact with an object while the object is being displayed within a display area created at a first location within the portion of ***the distributed hypermedia document*** being displayed in the browser-controlled window, with said network server coupled to said computer network environment . . . wherein at least said portion of ***the document*** is displayed within the browser-controlled window

'985-36: receiving, at the client workstation from the network server over the distributed hypermedia network environment, at least one ***file containing information to enable a browser application to display at least a portion of a distributed hypermedia document*** within a browser-controlled window . . . displaying at least a portion of ***the document*** within the browser-controlled window . . . automatically invoking the executable application, in response to the identifying of the embed text format, in order to enable an end-user to directly interact with the object, while the object is being displayed within a display area created at the first location within the portion of ***the hypermedia document*** being displayed in the browser-controlled window,

'985-40: receive, over said computer network environment from the network server, at least one ***file containing information to enable a browser application to display at least a portion of a distributed hypermedia document*** within a browser-controlled window . . . displaying, on said client workstation, at least a portion of ***the document*** within the browser-controlled window . . . automatically invoking the executable application, in response to the identifying of the embed text format, in order to enable an end-user to directly interact with the object while the object is being displayed within a display area created at the first location within the portion of ***the hypermedia document*** being displayed in the browser-controlled window

'985-44: wherein the client workstation receives, over the computer network environment from the server, at least one ***file containing information to enable a browser application to display, on the client workstation, at least a portion of a distributed hypermedia document*** within a browser-controlled window . . . wherein at least said portion of ***the document*** is displayed within the browser-controlled window . . . wherein the executable application is automatically invoked by the browser, in response to the identifying of the embed text format, to enable an end-user to directly interact with the object while the object is being displayed within a display area created at the first location within the portion of ***the hypermedia document*** being displayed in the browser-controlled window

b. <u>Specification (all cites to '906 patent)</u>

1:53–:60 & 2:23–:28 (Background of the Invention): Other Internet standards are the HyperText Transmission Protocol ("HTTP") that allows hypertext documents to be exchanged freely among any computers connected to the Internet and HyperText Markup Language ("HTML") that defines the way in which hypertext documents designate links to information. See, e.g., Berners-Lee, T. J., "The world-wide-web," Computer Networks and ISDN Systems 25 (1992). … ***A hypermedia document is similar to a hypertext document***, except that the user is able 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.

5:24–:38 (Background of the invention): The Internet is said to provide an "open distributed hypermedia system." It is an "open" system since Internet 100 implements a standard protocol that each of the connecting computer systems, 106, 130, 120, 132 and 134 must implement (TCP/IP). It is a "hypermedia" system because it is able to handle ***hypermedia documents*** as described above ***via standards such as*** the HTTP and ***HTML hypertext transmission and mark up standards***, respectively.

1:61–2:6 (Background of the Invention): A ***hypertext document*** is a document that allows a user to view a text document displayed on a display device connected to the user's computer and to access, retrieve and view other data objects that are linked to hypertext words or phrases in the ***hypertext document***. In a ***hypertext document***, the user may "click on," or select, certain words or phrases in the text that specify a link to other documents, or data objects. In this way, the user is able to navigate easily among data objects. The data objects may be local to the user's computer system or remotely located over a network. An early hypertext system is Hypercard, by Apple

Computer, Inc. Hypercard is a standalone system where the data objects are local to the user's system.

2:7–:13 (Background of the Invention): When a user selects a phrase in a ***hypertext document*** that has an associated link to another document, the linked document is retrieved and displayed on the user's display screen. This allows the user to obtain more information in an efficient and easy manner. This provides the user with a simple, intuitive and powerful way to "branch off" from a main document to learn more about topics of interest.

2:14–:27 (Background of the Invention): Objects may be text, images, sound files, video data documents or other types of information that is presentable to a user of a computer system. When a document is primarily text and includes links to other data objects according to the hypertext format, the document is said to be a ***hypertext document***. When graphics, sound, video or other media capable of being manipulated and presented in a computer system is used as the object linked to, the document is said to be a ***hypermedia document***. A ***hypermedia document*** is similar to a ***hypertext document***, except that the user is able 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***.

2:38–:47 & fig.1 (Background of the Invention): When the user clicks on the phrase "hypermedia," software running on the user's computer obtains the link associated with the phrase, symbolically shown by arrow 30, to access ***hypermedia document*** 14. ***Hypermedia document*** 14 is retrieved and displayed on the user's display screen. Thus, the user is presented with more information on the phrase "hypermedia." The mechanism for specifying and locating a linked object such as ***hypermedia document*** 14 is an HTML "element" that includes an object address in the format of a Uniform Resource Locator (URL).

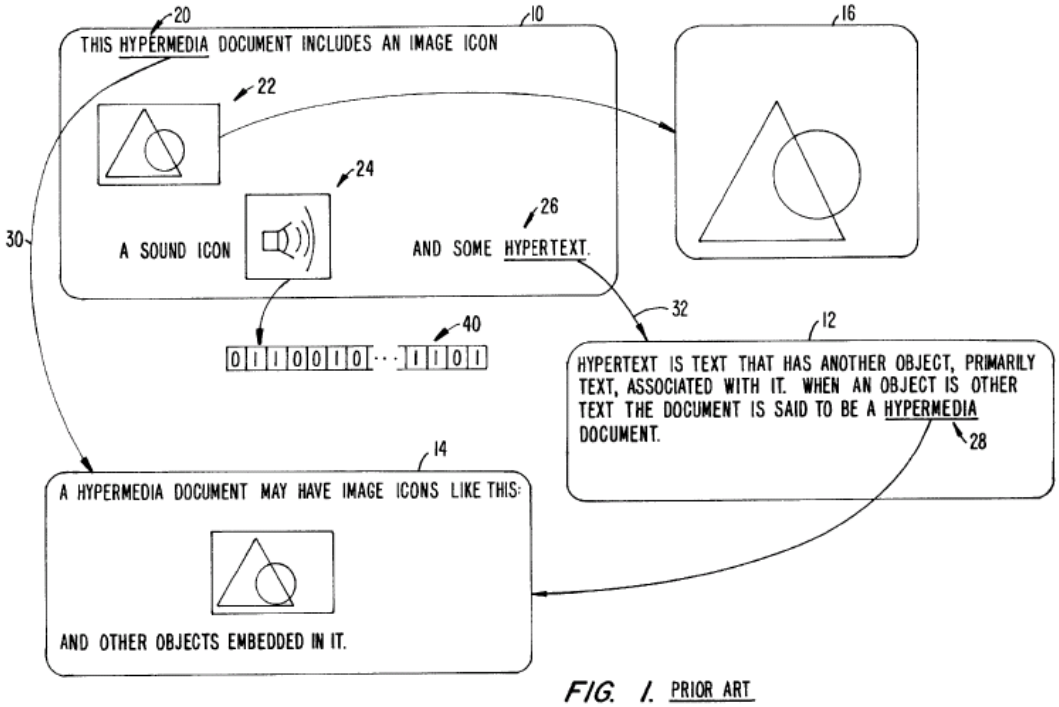


FIG. 1. PRIOR ART

2:56–:65 (Background of the Invention): Documents, and other data objects, can be referenced by many links from many different source documents. FIG. 1 shows document 14 serving as a target link for both documents 10 and 12. A ***distributed hypertext or hypermedia document*** typically has many links within it that specify many different data objects located in computers at

different geographical locations connected by a network. ***Hypermedia document*** 10 includes image icon 22 with a link to image 16. One method of viewing images is to include an icon, or other indicator, within the text.

4:60–5:13 (Background of the Invention): Referring again to FIG. 1, data objects such as distributed ***hypermedia documents*** 10, 12 and 14, image 16 and sound data file 40, may be located at any of the computers shown in FIG. 2. Since these data objects may be linked to a document located on another computer the Internet allows for remote object linking. For example, ***hypertext document*** 10 of FIG. 1 may be located at user 110's client computer 108. When user 110 makes a request by, for example, clicking on hypertext 20 (i.e., the phrase "hypermedia"), user 110's small client computer 108 processes links within ***hypertext document*** 10 to retrieve document 14. In this example, we assume that document 14 is stored at a remote location on server B. Thus, in this example, computer 108 issues a command that includes the address of document 14. This command is routed through server A and Internet 100 and eventually is received by server B. Server B processes the command and locates document 14 on its local storage. Server 14 then transfers a copy of the document back to client 108 via Internet 100 and server A. After client computer 108 receives document 14, it is displayed so that user 110 may view it.

5:14–:23 (Background of the Invention): Similarly, image object 16 and sound data file 40 may reside at any of the computers shown in FIG. 2. Assuming image object 16 resides on server C when user 110 clicks on image icon 22, client computer 108 generates a command to retrieve image object 16 to server C. Server C receives the command and transfers a copy of image object 16 to client computer 108. Alternatively, an object, such as sound data file 40, may reside on server A so that it is not necessary to traverse long distances via the Internet in order to retrieve the data object.

12:50–:65 (Detailed Description of a Preferred Embodiment):

> Next, a discussion of the software processes that perform parsing of a hypermedia document and launching of an application program is provided in connection with Table II and FIGS. **7**A, **7**B, **8**A and **8**B.
>
> Table II, below, shows an example of an HTML tag format used by the present invention to embed a link to an application program within a hypermedia document.

TABLE II

```
<EMBED
    TYPE = "type"
    HREF = "href"
    WIDTH = width
    HEIGHT = height
>
```

13:37–:50 & fig.7A (Detailed Description of a Preferred Embodiment): FIG. 7A is a flowchart describing some of the functionality within the HTMLparse.c file of routines. The routines in HTMLparse.c perform the task of parsing a ***hypermedia document*** and detecting the EMBED tag. In a preferred embodiment, the enhancements to include the EMBED tag are made to an HTML library included in public domain NCSA Mosaic version 2.4. Note that much of the source code in is pre-existing NCSA Mosaic code. Only those portions of the source code that relate to the new functionality discussed in this specification should be considered as part of the invention. The new functionality is identifiable as being set off from the main body of source code by conditional compilation macros such as "#ifdef . . . #endif" as will be readily apparent to one of skill in the art.

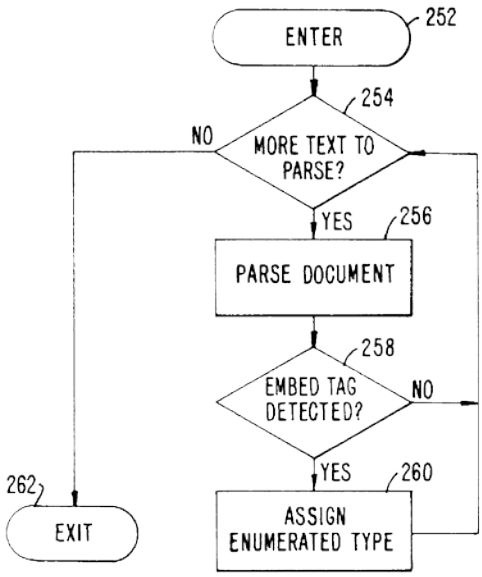


FIG. 7A.

14:24–:42 (Detailed Description of a Preferred Embodiment): Assuming there is more text to parse, execution proceeds to step 256 where routines in HTMLparse.c obtain the next item (e.g., word, tag or symbol) from the document. At step 258 a check is made as to whether the current tag is the EMBED tag. If not, execution returns to step 254 where the next tag in the document is obtained. If, at step 258, it is determined that the tag is the EMBED tag, execution proceeds to step 260 where an enumerated type is assigned for the tag. Each occurrence of a valid EMBED tag specifies an embedded object. HTMLParse calls a routine "get_mark" in HTMLparse.c to put sections of HTML document text into a "markup" text data structure. Routine get_mark, in turn, calls ParseMarkType to assign an enumerated type. The enumerated type is an idientifier with a unique integer associated with it that is used in later processing described below. Once all of the hypermedia text in the text portion to be displayed has been parsed, execution of HTMLparse.c routines terminates at step 262.

c. Prosecution history

i. '906 prosecution history (08/324,443)

Office Action, at 2 (May 6, 1996) (906 PH Ex. 02 at PH_001_0000783864):

> As per claim 1, the Mercury Project operated using a method essentially as claimed:
>
> providing client workstation [user computer browsing the WWW] and network server [the Mercury Project server]coupled to a distributed hypermedia network environment [World Wide Web;
>
> displaying at the client workstation a portion of a hyper media document [HTML document] wherein the document includes an embedded controllable application [controlling a robot arm and air pulse - see page 1 of "USC Mercury Project: Interface"];

Amendment A, at 1-3 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783879):	Applicants' Response, at 2 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784132):
1. (Amended) A method for running an application program in a computer network environment, comprising:	THE INVENTION OF CLAIM 1
… a browser application, that parses ***a distributed hypermedia document***…	A browser application parses a ***hypermedia document*** …
… text formats included in ***the distributed hypermedia document*** …	
… at least a portion of ***a first hypermedia document***…	
… wherein ***said first hypermedia document*** is displayed…	… The browser displays a portion of a ***first distributed hypermedia*** document …
… wherein ***said first distributed hypermedia document*** includes an embed text format...	…The ***hypermedia document*** includes an embed text format, located at a first location in ***the hypermedia document***, …
… location of an object external to ***the first distributed hypermedia document***…	…the location of at least a portion of an object external to ***the hypermedia document***. …
… executable application external to ***the first distributed hypermedia document***…	
… ***said first distributed hypermedia document*** continues to be displayed…	

Amendment A, at 1-3 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783910):

> 14. (Amended) … wherein HyperText Markup Language is used to specify said [embedded] controllable application within ***said hypermedia document***.

See also, e.g., Amendment A, at 7 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783885) (setting forth amendments to claim 44.)

Amendment A, at 13 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783891):

The present invention, as defined for example in amended claim 1, includes the steps of executing, at the client workstation, a browser application that parses a distributed hypermedia document to identify text formats included in the distributed hypermedia document and for responding to text formats to initiate processes specified by that text format. The browser is also utilized to display at least a portion of the distributed hypermedia document within a browser-controlled window.

Amendment A, at 16 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783894):

The claimed combination is fundamentally different from the Mercury Project. In the claimed combination, the external object and executable object are embedded by reference in the HTML document and the object is displayed and processed within the same window where a portion of the original document is displayed. In the Mercury Project information is passed back to the server and a new document is generated and displayed. There is no display and processing the external object within the window in which a portion of the original document is displayed.

Amendment A, at 17 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783895):

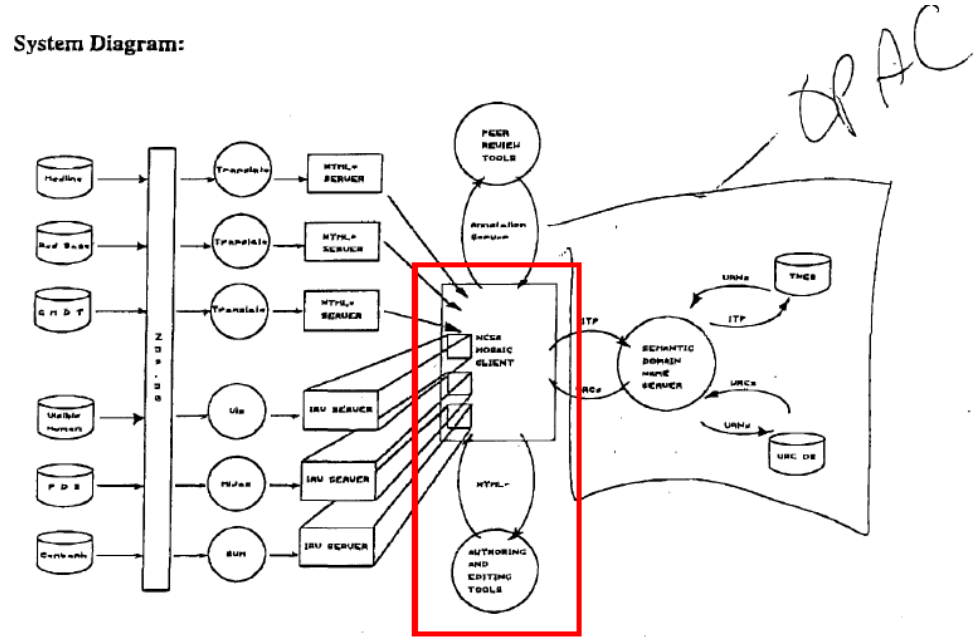
In view of the above, it is believed that the claims are not obvious over the disclosed prior art in view of Hansen. There is no disclosure in the references, singly or in combination, of displaying a hypermedia document in a first window including a text format specifying the location of an external object and identifying an external executable application or of invoking the external application to display and process the external object within the first window.

Applicants' Response, at 2 (Jan. 8, 1997) (906 PH Ex. 06 at PH_001_0000783957):

In claim 1, the distributed hypermedia document includes an embed text format that specifies the location of an object external to the distributed hypermedia document and that specifies type information utilized by the browser to identify and locate an executable application external to the distributed hypermedia document. The browser invokes the executable application to display and process the object within the browser window.

Applicants' Response, at Attachment I, p. 2 of 6 (Jan. 8, 1997) (906 PH Ex. 06 at PH_001_0000783962):

System Diagram:



Definitions: HTML+: Hypertext Mark-up Language -- This is the language that World Wide Web databases are encoded in, and that Mosaic interprets.

Applicants' Response, at Attachment I, p. 3 of 6 (Jan. 8, 1997) (905 PH Ex. 06 at PH_001_0000783963):

> These databases will reside behind a Z39.50 interface layer which yeilds, to the requesting client, the respective dataset in its native form. This data then goes through a translation layer where the data is either translated directly into HTML+ (Medline, Red Sage, CMDT) or loaded into a native-data visualization tool (Visible Human, PDB, Genbank). The HTML+ code is then passed to a set of HTML+ servers, which can be browsed by the Mosaic client. The visualization data is handled differently. The graphical I/O of the relevant visualization tool is passed to an interactive remote visualization (IRV) server, which handles both mapping of the display output from the visualization tool onto embedded live-visualization windows within the Mosaic-browsable HTML+ documents, as well as capture of user-entered mouse and keyboard events within the visualization windows and transmission of those mouse and keyboard events back to the relevant visualization tools. The user, browsing the system with the project's enhanced version of the Mosaic client, is presented with data and visualizations derived from these various databases, yet embedded into coherent, multimedia Mosaic documents.

Office Action, at 3 (Jan. 24, 1997) (906 PH Ex. 07 at PH_001_0000783999):

Wynne did not specifically disclose the document having text formats. It is not clear whether the HyperNet's hypermedia documents disclosed by Wynne uses text formats. However it is well known in the art at the time of the invention to form hypermedia documents using text formats (e.g. SGML, HTML, etc.). Hansen teaches to use text because it is machine independent so the result is more portable [p.257 4th paragraph]. Hence, one of ordinary skill in the art would have been motivated to use text formats to form hypermedia document.

Examiner Interview Summary Record, at 1 (Feb. 26, 1997) (906 PH Ex. 09 at PH_001_0000784011):

Description of the general nature of what was agreed to if an agreement was reached, or any other comments:

1) HyperNet is a compiled system, 2) Tag in document to activate external program (delayed binding), 3) display and process by the external application within Browser's controlled window.

Applicant's argument is persuasive to overcome the Hypernet ref. The claims are distinguished over the prior art of record.

Amendment B, at 1-2 (June 2, 1997) (906 PH Ex. 11at PH_001_0000784029-30): … created at said first location within the portion of said first ***distributed hypermedia document being displayed*** …

Amendment B, at 13 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784041):

The Applicants' invention 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. This allows the hypermedia document to act as a platform for entirely new kinds of applications that could not have been possible before the invention.

Amendment B, at 18 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784046):

Next, some sort of data interchange interface would have to be constructed between Mosaic and the Khoyi virtual machine to allow Khoyi's "packs" to create data structures that could be transferred to Mosaic, and for messages created by Mosaic to be transferred to Khoyi.

Mosaic would then have to be modified to allow data object components of the document to be "linked" to Khoyi's applications which can process the data. These links would be defined by an external link table, and the linking relationship would not be affected by the text of the document.

These links would be distinguished from the HTML anchor links defined in the hypermedia document, which would require incorporating two incompatible linking systems to be maintained by the system. Mosaic teaches that a major advantage of the HTML document format is that all links should be defined by the document text. This teaches away from combining the two systems in the proposed way, since the result would be awkward, overly complex and difficult to maintain.

Declaration of Michael D. Doyle, at 10 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784071):

identifier, which is referenced in a document, for example, as a "link marker." The actual definition of the link referenced by any particular link marker is located in an operating system data structure called a "link table." The document itself does not allow the document author to explicitly define or control the definition of the link's internal details, such as the precise location of a data file on a disk drive. The Web, on the other hand, employs a uniform resourse locator (URL) construct to manage both link definition and object localization on networked systems, from within the Web document, under the precise control of the Web document author. It appears that the URL mechanism would be incompatible with the linking mechanism requirements imposed by the Khoyi operating system. Since the HTML-based mechanism for linking and object management is one of the major requirements for a successful Web browser, such an incompatibility would likely render the resulting system useless for its intended purpose.

Office Action, at 4 (Aug. 25, 1997) (906 PH Ex. 12 at PH_001_0000784094):

HTML is a text tag structure document encoding. It is apparent the prior art as modified would have had a text tag for indicating links to an in-place interactive object.

Declaration of Michael D. Doyle, at 2-3 (Oct. 29, 1997) (accompanying Applicants' Remarks (Oct. 31, 1997)) (906 Ex. 15 at PH_001_0000784129-30):

2. The subject matter claimed in the above patent application was reduced to practice in this country prior to April 15, 1994, the filing date of the parent of the Koppolu reference cited by the examiner.

3. The reduction to practice of the claimed invention is evidenced by ATTACHMENTS A and B. ATTACHMENT A is a copy of a paper entitled "Integrated Control of Distributed Volume Visualization Through the World-Wide-Web", by Ang, Martin, and Doyle. This paper was submitted for publication prior to April 15, 1994. 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.

4. As stated in ATTACHMENT A, at page 5, paragraph 3.2, Mosaic (the browser) interprets the HTML <EMBED> tag included in a document to create a drawing area widget in a document presentation and creates a shared window system buffer to receive visualization results. In addition, when the browser parses an <EMBED> tag in the document, the browser automatically launches the external application specifying the location of the visual object to render and identify the shared image buffer. The format and operation of an EMBED tag for 3D image data is described at paragraph 3.1.

5. As stated in ATTACHMENT B, starting at the bottom of page 2, interface and control software had been developed that allows the embedding of a visualization application within a Mosaic document. As is apparent from the photographs, the object is displayed and processed within the browser-controlled window. The visualization application is external to the hypermedia document displayed by the browser. Automatic launching of the external application when an HTML document is opened by the browser is depicted in the video.

Declaration of Michael D. Doyle, at Attachment A, pp. 4-5 (Oct. 31, 1997) (accompanying Applicants' Remarks (Oct. 31, 1997)) (906 PH Ex. 13 at PH_001_0000784108):

We have enhanced the Mosaic W3 browser to support both a three-dimensional data object and communication with VIS as a cooperating application (figure 2). Mosaic provides the user with the ability to locate and browse information available from a wide variety of sources including FTP, WAIS, and Gopher. HTTP servers respond to requests from clients, e.g. Mosaic, and transfer hypertext documents. Those documents may contain text and images as intrinsic elements and may also contain external links to any arbitrary data object (e.g. audio, video, etc...). Mosaic may also communicate with other Internet servers, e.g FTP, either directly – translating request results into HTML on demand – or via a gateway that provides translation services. As a W3 client, Mosaic communicates with the server(s) of interest in response to user actions (e.g. selecting a hyperlink), initiating a connection and requesting the document specified by the URL. The server delivers the file specified in the URL, which may be a HTML document or a variety of multimedia data files (for example, images, audio files, and MPEG movies) and Mosaic uses the predefined SGML DTD for HTML to parse and present the information. Data types not directly supported b Mosaic are displayed via user-specifiable external applications and we have extended that paradigm to both include three-dimensional volume data as well as to integrate the external application more completely with Mosaic.

Applicants' Response, at 4 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784134):

The different functions and purposes of Mosaic and Koppolu (OLE) are reflected in the different document structure. In Mosaic, since HTML documents are designed to be platform independent, the document structure is simple ASCII text. A browser parses a received document to identify HTML tags which specify various aspects of the document's appearance and links to other documents. In Koppolu, a container application creates a complex file structure which is utilized to render a document. There is no text parsing in Koppolu to render the compound document.

Applicants' Response, at 17 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784147):

In order to insure cross-platform uniformity of document appearance, the document was defined through the use of ASCII text, where specific text formats, otherwise known as "tags," would be used within the document text to specify various aspects of the document's appearance and linkages to other documents or related data. Each browser, therefore, incorporated a parser which would distinguish the formatting tags from the document's narrative text, classify those tags into pre-defined categories, break each tag into its basic components, and then invoke appropriate browser subroutines to respond appropriately to the meanings of the tag components. Although the browser subroutines were built from machine-specific native code, this text tag mechanism allowed the design of a variety of browsers for various computing platforms that could respond in similar ways to similar types of text tags, and therefore result in similar-appearing documents on dissimilar computers. A binary document data format was avoided in order to promote cross-platform compatibility, due to the variation in binary data handling methodologies on various different operating systems, and to simplify the requirements for document creation tools. All that a Web document author needs in order to create a Web document file is a simple ASCII text editor, which is a pre-existing application in all commonly-found operating system packages.

Response to Final Rejection, at 25 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784155): Additionally, because the claimed embed text formats in the document cause the browser to automatically invoke the external application, the hypermedia ***document*** itself, and by implication the author of that document, ***directly*** control the extension of the functionality of the browser. As a consequence of the features of the claimed invention, the ***document,*** rather than the browser, **becomes the application;** that is, the document together with its embedded program objects, exposes all the functionality that the user needs to interact with and process the entire content of the compound hypermedia document.

ii. Abandoned application (09/075,359)

Applicants' Response, at 1-2 (March 9, 2001) (359 PH Ex. 04 at PH_001_0000787808-09):

62. (New) A computer program product for use in a system …

… at least a portion of ***a first hypermedia document*** received over said network…

… the portion of ***said first hypermedia document*** is displayed …

… wherein ***said first distributed hypermedia document*** includes…

… located at a first location in ***said first distributed hypermedia document***…

… external to ***said first distributed hypermedia document***…

… external to the ***first distributed hypermedia document*** …

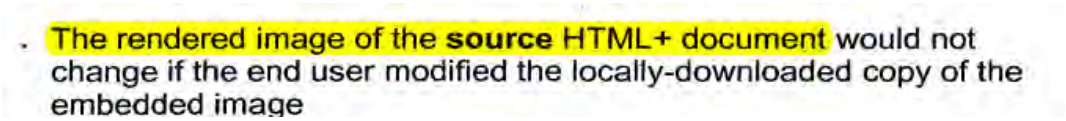
… within the portion of ***the first distributed hypermedia document***…

Response to Office Action, at 7 (Nov. 29, 2001) (359 PH Ex. 06 at PH_001_00007878329):

> In Risberg a user customizes the application by utilizing scripts and setting up alarm limits. In the claimed invention, the document itself coordinates the use of external program code with embed text formats, such as the Netscape <embed> tag or the ActiveX <object> tag, at locations in the document where the external computer readable code is to display and enable interactive processing of an external object.
>
> Thus, Risberg and the claimed computer program product implement completely different ·paradigms. In Risberg, a user, having access to the application running on the workstation, customizes the application using many features, such as scripts and tools, built into the application. In the invention of claim 62, the document itself causes the browser to automatically invoke external program code to perform customized functions selected by the hypermedia document author rather than the user of the program.

iii. First reexam (90/006,831)

Doyle Presentation, April 27, 2004 (accompanying Ex Parte Reexamination Interview Summary (April 27, 2004)) (831 PH Ex. 06 at PH_001_0000785323):



Applicants' Remarks, at 2 (May 11, 2004) (831 PH Ex. 07 at PH_001_0000785360):

> d. The rendered image of the source HTML+ document displayed by the browser would not change if the end user utilized an external editor to modify the locally-downloaded copy of the embedded image.

Applicants' Remarks, at 3-4 (May 11, 2004) (831 PH Ex. 07 at PH_001_00007853601-62):

The browser application then parses the local copy of the HTML document, renders the temporary local copy of the HTML document into a Web page , and displays the rendered Web page in a browser-controlled window. [Felten, paragraph 21] During the rendering step, the browser may retrieve information external to the local copy of the HTML document, such as source files referenced by IMG tags, render the images from the retrieved files as static graphic images, and insert the images into the Web page of the HTML document, for display to the user.

There is no further interaction with the source HTML document or the local copy of the source HTML document subsequent to its being rendered and displayed. If a user believes the source HTML document has changed (s)he can click a refresh button in the browser GUI which causes the browser application to retrieve the source HTML document from the network server again, store a local copy again, parse and render again the newly retrieved local copy of the source HTML document, and replace the display of the previous version of the retrieved source HTML document with the subsequently retrieved version in the browser-controlled window or another window. For example, if the source HTML document were a price list of goods the user might refresh the document to determine if the prices had changed.

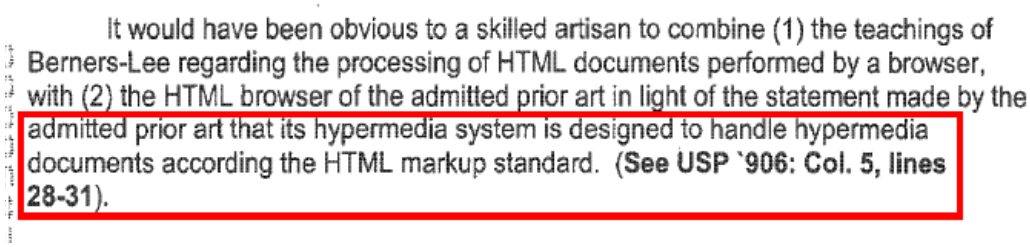
Applicants' Remarks, at 13 (May 11, 2004) (831 PH Ex. 07 at PH_001_0000785371):

d. The rendered image of the source HTML+ document displayed by the browser would not change if the end user utilized an external editor to modify the locally-downloaded copy of the embedded image

As described above, the World Wide Web of Mosaic and Berners-Lee is a write-once-publish-many paradigm in which all users would see the same basic document content that the Web page author originally designed. This refers to the fact that a Web document author would create a single document file and publish that document merely by making it accessible on an Internet server. An unlimited number of users could then retrieve and view that document by simply entering the Internet address of the document (the URL) into their Internet-connected Web browsers executing on their client computers. The browsers would use a simple request/response protocol to retrieve specified documents and related data from remotely-networked Web server programs. [Felten, paragraph 20]

24. In addition, because web pages were written in one format (HTML) and viewed in another (visual representation), it did not make sense to talk about editing and viewing a document in the same window. Web page authors would typically work with two separate windows open, one (a browser) to see what the visual representation looked like, and another (an external editor) to actually modify the page's HTML representation. An author would fiddle with the HTML, then click the save button in the editor and the refresh button in the browser to see what the visual representation of the page looked like, then fiddle with the HTML some more, and so on until he was satisfied with the page's appearance.

Office Action, at 4 (Aug. 16, 2004) (831 PH Ex. 12 at PH_001_0000785558)):


It would have been obvious to a skilled artisan to combine (1) the teachings of Berners-Lee regarding the processing of HTML documents performed by a browser, with (2) the HTML browser of the admitted prior art in light of the statement made by the admitted prior art that its hypermedia system is designed to handle hypermedia documents according the HTML markup standard. **(See USP `906: Col. 5, lines 28-31).**

Applicants' Response, at 18 (Oct. 12, 2004) (831 PH Ex. 16 at PH_001_0000785820): Another important principle of the Web model taught by the Mosaic, Berners-Lee, Raggett I and II combination is that of referential integrity. In the Web model, the ***HTML document*** author can ***specify the specific locations***, contained in "hypertext links," from which the browser will retrieve new ***HTML documents*** when users click upon those links. ***These links are easily specified through embed text formats in the document text***. … .

Applicants' Response, at 20-21 (Oct. 12, 2004) (831 PH Ex. 16 at PH_001_0000785822-23):

PART III The obviousness rejection is based on a false premise and therefore reaches a false conclusion.

a. Toye does not disclose a distributed hypermedia system in which a hypermedia browser allows a user to interactively process an object embedded within a distributed hypermedia document.

The Office Action, at page 6, lines 21-26, states that Toye discloses a distributed hypermedia system in which a hypermedia browser allows a user to interactively process an object embedded within a distributed hypermedia document. However, this statement is incorrect in view of the precise meaning of the various terms defined in the Mosaic, Berners-Lee, Raggett I and II combination.

The admitted prior art describes a hypertext document as "a document that allows a user to view a text document displayed on a display device connected to the user's computer and to access, retrieve and view other data objects that are linked to hypertext words or phrases in the hypertext document. In a hypertext document, the user may "click on," or select, certain words or phrases in the text that specify a link to other documents, or data objects." ['906 at col. 1, line 61] "A hypermedia document is similar to a hypertext document, except that the user is able 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." ['906 at col. 2, line 23]. When the hypermedia document is displayed on a browser program the browser responds to the selection of a link to retrieve and display the hypermedia document or data object referenced by the link.

A distributed hypermedia system "is a "distributed" system because data objects that are imbedded within a document may be located on many of the computer systems connected to the Internet." ['906 at col. 5, lines 25-38].

The use of the HTML allows the Internet to be an open system where a standard protocol is implemented by each computer connected to the internet. The structure of the document is defined by the author utilizing particular sets of characters that have a universal meaning.

In contrast, Toye teaches a system that is not a distributed system but requires that all referenced objects be stored in a single data base called DIS. [Toye, page 40, column 2, first and second paragraphs below the heading "Distributed Information Service (DIS)].

The NoteMail pages described in Toye use DIS as the central repository for referenced objects in contrast to the ability of a distributed hypermedia document to reference objects located in computers at different geographic locations. Thus, the Toye system does not teach or suggest using distributed hypermedia documents and its principle of operation is incompatible with the use of distributed hypermedia documents. [Felten II, at paragraph 24].

Also, for the same reasons Toye does not teach the use of a "distributed hypermedia environment" as that term is defined in the admitted prior art and used in claims 1 and 6 of the '906 patent. The use of the centralized storage of referenced objects is crucial to the intended purpose of the Toye system and contradicts the basic requirements of a distributed hypermedia environment. [Felten II, at paragraph 25].

Toye does not teach a hypermedia browser application, as that term is defined in the admitted prior art, Berners-Lee, and Raggett I and II, understood by the PHOSA at the time the application was filed, and as used in claims 1 and 6 of the '906 patent. Toye teaches no software application that parses distributed hypermedia documents or that uses text formats, and it does not teach other browser-related elements of the '906 claims, such as parsing of distributed hypermedia documents by a browser, identifying text formats in distributed hypermedia documents and responding to predetermined text formats to initiate processing specified by those formats, utilizing a browser to display at least a portion of a distributed hypermedia document in a browser-controlled window, and parsing an embed text format in such a document. [Felten II, at paragraphs 26-27].

Further, the Toye reference teaches that information can be organized by adding links between objects where the links themselves are objects stored in the DIS database. [Toye, page 41, col. 1, first partial paragraph]. Thus, Toye is not a hypermedia system because, in the admitted prior art, Berners-Lee, and Raggett I and II combination, links are defined by the author as text formats in the hypermedia document and resolved by the browser application.

The Mosaic, Berners-Lee, Raggett I and II combinations teaches the use of a hypermedia document that is a text document where some characters within the text are interpreted as mark-up tags specified by the HTML standard. The mark-up "tags" give structure to the document. [Berners-Lee, page 5, Felten II, at paragraph 14].

In contrast, Toye teaches that the structure, i.e., spatial arrangement of information in a NoteMail page, is preserved by a non-standard MIME "Format" data type defined by the Toye authors for the specific NoteMail system being described. [Toye, page 40, first column, last partial paragraph, Felten II, at paragraph 31]. Accordingly, Toye does not teach the use of a hypermedia

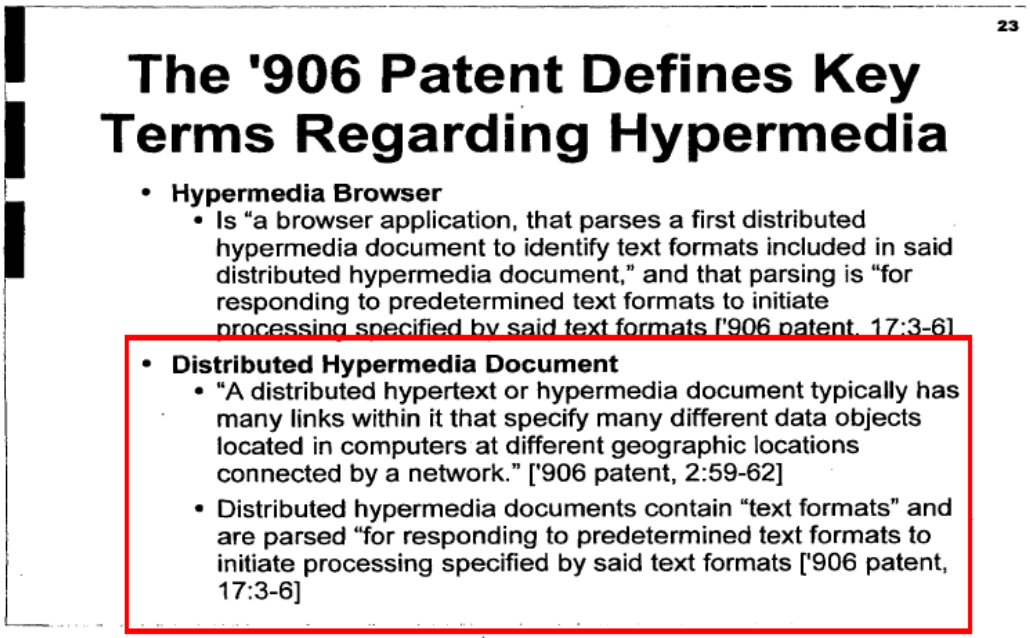
document, in the sense of the Mosaic, Berners-Lee, Raggett I and II combination, or the embedding of an object in such a hypermedia document. NoteMail pages are therefore not analogous to Web-style hypermedia documents.

Also, there is no teaching in Toye of interactively processing an object embedded in a hypermedia document. Toye teaches that data displayed in a NoteMail page must be selected via a mouse click by the user to restart an application in order to update and edit data. The type of application described in Toye is any application that displays through an X-server. [Toye page 40, second column, first full paragraph]. There is no teaching of modifying such an application to process an object embedded in a hypermedia document. Further, Toye teaches that most data remains physically under the control of the application that created it, suggesting that the data must be processed using the normal interface for the application. [Felten II, at paragraphs 36-37].

Declaration of Edward W. Felten, at ¶ 23 (Oct. 6, 2004) (accompanying Applicants' Response (Oct. 12, 2004)) (831 PH Ex. 13 at PH_001_0000785580):

> 23. Contrary to the Office Action's assertion, Toye does not teach the use of a "distributed hypermedia document," as that term is used in the '906 claims. The term's meaning, as understood by a PHOSA, was reiterated in the '906 Patent's specification. For example:
>
> > A distributed hypertext or hypermedia document typically has many links within it that specify many different data objects located in computers at different geographic locations connected by a network.
>
> ('906 Patent at 2:59-62)

Doyle Presentation, at slide 23 (Aug. 18, 2005) (831 PH Ex. 17 at PH_001_0000785888).



iv. Interference 105,563 McK

Doyle Annotated Copy of Claims, at 2-3 (July 3, 2007) (563 PH Ex. 02 at PH_001_0000787571-72):

1. A method for running an application program { **Fig. 5, item 210** } in a computer network environment { **Fig. 5, item 206** }, comprising:

providing at least one client workstation { **Fig. 5, item 200** } and one network server { **Fig. 5, item 204** } coupled to said network environment { **Fig. 5, item 206** }, wherein said network environment { **Fig. 5, item 206** } is a distributed hypermedia environment { **Fig. 2, item 100** };

executing, at said client workstation { **Fig. 5, item 200** }, a browser application { **Fig. 5, item 208** }, that parses { **Fig. 7A, step 256** } a first distributed hypermedia document { **Fig. 5, item 212** } to identify text formats { **Fig. 5, item 214** } included in said distributed hypermedia document { **Fig. 5, item 212** } and for responding to predetermined texr formats { **Fig. 5, item 214** } to initiate processing specified by said text formats { **Fig. 5, item 214** }; utilizing said browser { **Fig. 5, item 208** } to display, on said client workstation { **Fig. 5, item 200** }, at least a portion of a first hypermedia document { **Fig. 5, item 212** } received over said network { **Fig. 5, item 206** } from said server { **Fig. 5, item 204** }, wherein the portion of said first hypermedia document { **Fig. 5, item 212** } is displayed within a first browser-controlled window { **Fig. 9, item 350** } on said client workstation { **Fig. 5, item 200** }, wherein said first distributed hypermedia document { **Fig. 5, item 212** } includes an embed text format { **Fig. 5, item 214** }, located at a first location in said first distributed hypermedia document { **Fig. 5, item 212** }, that specifies the location of at least a portion of an object { **Fig. 5, item 216** } external to the first distributed hypermedia document { **Fig. 5, item 212** }, wherein said object { **Fig. 5, item 216** } has type information associated with it utilized by said browser { **Fig. 5, item 208** } to identify and locate an executable application { **Fig. 5, item 210** } external to the first distributed hypermedia document { **Fig. 5, item 212** }, and wherein said embed

text format { **Fig. 5, item 214** } is parsed { **Fig. 7A, step 256** } by said browser { **Fig. 5, item 208** } to automatically invoke { **Fig. 8A, step 290**} said executable application { **Fig. 5, item 210** } to execute on said client workstation { **Fig. 5, item 200** } in order to display said object { **Fig. 5, item 216** } and enable interactive processing of said object { **Fig. 5, item 216** } within a display area created at said first location within the portion of said first distributed hypermedia document { **Fig. 5, item 212** } being displayed in said first browser-controlled window { **Fig. 9, item 350** }.

v. Second reexam (90/007,858)

Applicants' Response, at 4 (Sept. 27, 2007) (858 PH Ex. 05 at PH_001_0000787033):

A. The Claimed Invention.

The invention, as recited for example in claims 6, is for use in a system having at least one client workstation and one network server coupled to a network environment.

The claims recite a browser application, executed on the client workstation, that parses a hypermedia document to identify text formats in the document and responds to predetermined text formats to initiate processing specified by the text formats.

The browser displays a portion of a first distributed hypermedia document, received over the network from the network server, in a browser-controlled window. The hypermedia document includes an embed text format, located at a first location in the hypermedia document, that specifies the location of at least a portion of an object external to the hypermedia document. The object has associated type information utilized by the browser to identify and locate an executable application external to the hypermedia document.

vi. '985 prosecution history (10/217,955)

Applicants' Supplemental Amendment after Non-Final Rejection, at 15-17 (April 11, 2008) (985 PH Ex. 11 at PH_001_0000784582-84):

The REEXAMINATION REASONS FOR PATENTABILITY/CONFIRMATION (hereinafter "the Confirmation"), included with the Notice of Intent to Issue Ex Parte Reexamination Certificate (mailed 09/27/2005), which concluded the first reexamination includes four parts analyzing whether the claims in reexamination are unpatenable over the admitted prior art in the U.S. Patent No. 5,838,906 ('906 patent), the teachings of Berners-Lee, Raggett I, and Raggett II (the "four-way combination") and the newly cited teaching of Toye.

* * * * *

Claim 19 of the present application recites "automatically invoke the executable application, in response to the identifying of the embed text format, to execute on the client workstation in order to display the object and enable interactive processing of the object while the object is being displayed within a display area created at the first location within the portion of the hypermedia document being displayed in the browser-controlled window".

As determined in Part I of the Confirmation, neither the four-way combination of the patent owner's admitted prior art (APA), Berners-Lee, Raggett I and Raggett II nor Toye, singularly or in combination, fairly teach or suggest automatically invoking an external application to enable interactive processing of an object being displayed within the display area.

The four-way combination was determined to teach displaying a static object. Toye was found to display a "static snapshot" of external content where interactive processing was not automatically invoked but required manual selection by the user.

Accordingly, since the recited automatically invoking to enable interactive processing is not taught or suggested by the cited references a *prima facie* case of obviousness has not been established.

Independent claims 4, 23, 27, 31, 35, 39, 43 and 47 recite similar limitations as claim 19 and are thus allowable for the same reasons. The remaining claims are dependent claims which are allowable for the same reasons as the claims on which they depend.

The other parts of the Confirmation recite other findings supporting the determination that the reexamination claims are not unpatentable over the cited references and these findings also support a determination that the pending claims of the present application are not unpatentable over the cited references. A complete copy of the Confirmation is appended to this response.

Supplemental Amendment, at 2-14 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784614,16-17,19-20,22,24-25):

4. (currently amended) A method for running an application program in a distributed hypermedia network environment... comprising:

… to display at least a portion of ***a distributed hypermedia document*** …

… displaying at least a portion of ***the document***…

… a first location in ***the document***…

… within the portion of ***the hypermedia document*** being displayed…

19. (currently amended) One or more computer readable media encoded with software comprising computer executable instructions … operable to:

… display at least a portion of ***a distributed hypermedia document*** ...

… display at least a portion of ***the document*** …

… a first location in ***the document***…

… the first location within the portion of ***the hypermedia document*** being displayed…

23. (currently amended) A method of serving digital information in a computer network environment … comprising:

… at least a portion of ***a distributed hypermedia document***…

… displaying … at least a portion of ***the document***…

… a first location in ***the document***…

… the first location within the portion of ***the hypermedia document*** being displayed …

27. (currently amended) A method for running an executable application in a computer network environment … the method comprising:

…a first location within a portion of a ~~the~~ ***hypermedia document*** being displayed…

… display … at least said portion of ***said distributed hypermedia document*** …

…at least said portion of ***the document*** is displayed …

… said first location in ***the document*** is identified…

31. (currently amended) One or more computer readable media encoded with software … operable to:

… a ~~the~~ portion of a ~~the~~ ***hypermedia document*** being displayed…

… display … said portion of said ***distributed hypermedia document*** …

… at least said portion of ***the document*** is displayed …

… said first location in ***the document*** is identified…

39. (currently amended) A method for running an application program in a distributed hypermedia network environment … the method comprising:

… display at least a portion of ***a distributed hypermedia document***…

… displaying at least a portion of ***the document*** …

… a first location in ***the document***…

… the first location within the portion of ***the hypermedia document***…

43. (currently amended) A method of serving digital information in a computer network environment … the method comprising:

… display at least a portion of ***a distributed hypermedia document*** …
… displaying … ***the document***…
… a first location in ***the document***…
… within the portion of ***the hypermedia document***…

47. (currently amended) A method for serving digital information in a computer network environment … said method comprising:
… display … at least a portion of a ***distributed hypermedia document***…
… said portion of ***the document*** is displayed
… a first location in ***the document***…
… the first location within the portion of ***the hypermedia document***…

Supplemental Amendment, at 18-19 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784623,31-32):

the method comprising:

receiving, at the client workstation from the network server over the network environment, at least <u>one file</u>

EXAMPLE SUPPORT:

2:14 "Objects may be text, images, sound files, video data, documents or other types of information that is presentable to a user of a computer system. When a ***document*** is primarily text and includes links to other data objects according to the hypertext format, the document is said to be a ***hypertext document***. When graphics, sound, video or other media capable of being manipulated and presented in a computer system is used as the object linked to, ***the document is said to be a hypermedia document***. A hypermedia document is similar to a hypertext document, except that the user is able 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. FIG. 1 shows examples of hypertext and hypermedia documents and links associating data objects in the documents to other data objects. Hypermedia document 10 includes hypertext 20, an image icon at 22, a sound icon at 24 and more hypertext 26. FIG. 1 shows hypermedia document 10 substantially as it would appear on a user's display screen."

3:34 "As discussed above, ***hypermedia documents*** allow a user to access different data objects. The objects may be text, images, sound files, video, additional documents, etc. As used in this specification, a data object is information capable of being retrieved and presented to a user of a computer system."

9:20 "In this example, hypermedia document 212 has been retrieved from a server connected to network 206 and has been loaded into, e.g., client computer 200's RAM or other storage device."

Supplemental Amendment, at 19-20 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784632-33):

> ***containing information to enable a browser application to display at least a portion of a distributed hypermedia document***
>
> **EXAMPLE SUPPORT:**
>
> **1:61** "A hypertext document is a document that allows a user to view a text document displayed on a display device connected to the user's computer and to access, retrieve and view other data objects that are linked to hypertext words or phrases in the hypertext document."
>
> **2:14** "Objects may be text, images, sound files, video data, documents or other types of information that is presentable to a user of a computer system. When a document is primarily text and includes links to other data objects according to the hypertext format, the document is said to be a hypertext document. When graphics, sound, video or other media capable of being manipulated and presented in a computer system is used as the object linked to, the document is said to be a hypermedia document. A hypermedia document is similar to a hypertext document, except that the user is able 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."
>
> **9:24** "Once hypermedia document 212 has been loaded into client computer 200, browser client 208 parses hypermedia document 212. In parsing hypermedia document 212, browser client 208 detects links to data objects as discussed above in the Background of the Invention section."

Applicants' Response, at 9 (March 11, 2005) (985 PH Ex. 03 at PH_001_0000784221):

> The specification of the '906 patent (Applicants' Admitted Prior Art) describes a browser application, e.g., Mosaic, that functions as a viewer to view HTML documents. There are several ways to retrieve an HTML document from a network server, all of which require user interaction with the browser. [Felten I, paragraph 8]. The browser then retrieves a selected published source HTML document from a network server by utilizing a uniform resource locator (URL) that locates the HTML document on the network and stores a temporary local copy of the HTML source document in a cache on the client workstation.
>
> The browser application then parses the local copy of the HTML document, renders the temporary local copy of the HTML document into a Web page , and displays the rendered Web page in a browser-controlled window. [Felten I, at paragraph 21]. During the rendering step, the browser may retrieve information external to the local copy of the HTML document, such as source files referenced by IMG tags, render the images from the retrieved files as static graphic images, and insert the images into the Web page of the HTML document, for display to the user.

Notice of Allowability, at 2 (Mar. 20, 2009) (985 PH Ex. 15 at PH_001_0000784733):

The following is an examiner's statement of reasons for allowance: the claims are allowable as the claims contain the subject matter deemed allowable in both Re exam 90/006,831 and Re exam 90/007,838 for the same reasons as set forth in the NIRC of the two Re exams.

2. Defendants' extrinsic evidence

Microsoft Press Computer Dictionary 178 (1st ed. 1991) (418–19) ("hypermedia") [Ex. R at PA-00333469]:

hypermedia The integration of graphics, sound, video, or any combination into a primarily associative system of information storage and retrieval. Hypermedia, especially in an interactive format where choices are controlled by the user, is structured around the idea of offering a working and learning environment that parallels human thinking—that is, an environment that allows the user to make associations between topics rather than move sequentially from one to the next, as in an alphabetic list. Hypermedia topics are thus linked in a manner that allows the user to jump from subject to related subject in searching for information. For example, a hypermedia presentation on navigation might include links to such topics as astronomy, bird migration, geography, satellites, and radar. If the information is primarily in text form, the product is hypertext; if video, music, animation, or other elements are included, the product is hypermedia. *See also* hypertext.

I. "distributed application"

Claim Term(s)	Defendants' Proposed Construction	Eolas's Proposed Construction
distributed application	an application in which tasks are broken up and performed in parallel on two or more computers	an application that may be broken up and performed among two or more computers

1. Defendants' intrinsic evidence

a. Claims

In the following chart, the term(s) for construction appear in all the claims marked with an "x." The letter "m" indicates a method claim, and "¶6" indicates a claim that the Defendants contend includes limitations subject to § 112, ¶ 6.

	'906 patent						'985 patent								
	m 1	m 4	m 5	¶6 6	¶6 9	¶6 10	m 1	¶6 16	m ¶6 20	m ¶6 24	¶6 28	m ¶6 32	m 36	m ¶6 40	m 44
distributed application													x	x	x

b. Specification (all cites to '906 patent)

7:7–:40 (Summary of the Invention):

> In one application, high resolution three dimensional images are processed in a distributed manner by several computers located remotely from the user's client computer. This amounts to providing parallel distributed processing for tasks such as volume rendering or three dimensional image transformation and display. Also, the user is able to rotate, scale and otherwise reposition the viewpoint with respect to these images without exiting the hypermedia browser software. The control and interaction of viewing the image may be provided within the same window that the browser is using assuming the environment is a "windowing" environment. The viewing transformation and volume rendering calculations may be performed by remote distributed computer systems.
>
> Once an image representing a new viewpoint is computed the frame image is transmitted over the network to the user's client computer where it is displayed at a designated position within a hypermedia document. By transmitting only enough information to update the image, the need for a high bandwidth data connection is reduced. Compression can be used to further reduce the bandwidth requirements for data transmission.
>
> Other applications of the invention are possible. For example, the user can operate a spreadsheet program that is being executed by one or more other computer systems connected via the network to the user's client computer. Once the spreadsheet program has calculated results, the results may be sent over the network to the user' s client computer for display to the user. In this way, computer systems located remotely on the network can be used to provide the computing power that may be required for certain tasks and to reduce the data bandwidth by only transmitting results of the computations.

10:33–11:39 & figs. 5-6 (Detailed Description of a Preferred Embodiment):

Another embodiment of the present invention uses an application server process executing on server computer **204** to assist in processing that may need to be performed by an external program. For example, in FIG. 5, application server **220** resides on server computer **204**. Application server **220** works in communication with application client **210** residing on client computer **200**. In a preferred embodiment, application server **220** is called VRServer, also a part of Doyle Group's approach. Since server computer **204** is typically a larger computer having more data processing capabilities and larger storage capacity, application server **220** can operate more efficiently, and much faster, than application client **210** in executing complicated and numerous instructions.

In the present example where a multidimensional image object representing medical data for an embryo is being viewed, application server **220** could perform much of the viewing transformation and volume rendering calculations to allow a user to interactively view the embryo data at their client computer display screen. In a preferred embodiment, application client **210** receives signals from a user input device at the user's client computer **200**. An example of such input would be to rotate the embryo image from a current position to a new position from the user's point of view. This information is received by application client 210 and processed to generate a command sent over network **206** to application server **220**. Once application server **220** receives the information in the form of, e.g., a coordinate transformation for a new viewing position, application server **220** performs the mathematical calculations to compute a new view for the embryo image. Once the new view has been computed, the image data for the new view is sent over network **206** to application client **210** so that application client **210** can update the viewing window currently displaying the embryo image. In a preferred embodiment, application server **220** computes a frame buffer of raster display data, e.g., pixel values, and transfers this frame buffer to application client **210**. Techniques, such as data compression and delta encoding, can be used to compress the data before transmitting over network **206** to reduce the bandwidth requirement.

It will be readily seen that application server **220** can advantageously use server computer **204**'s computing resources to perform the viewing transformation much more quickly than could application client **210** executing on client computer **200**. Further, by only transmitting the updated frame buffer containing a new view for the embryo image, the amount of data sent over network **206** is reduced. By using appropriate compression techniques, such as, e.g., MPEG (Motion Picture Experts Group) or JPEG (Joint Photographic Experts Group), efficient use of network **206** is preserved.

FIG. **6** shows yet another embodiment of the present invention. FIG. **6** is similar to FIG. **5**, except that additional computers **222** and **224** are illustrated. Each additional computer includes a process labeled "Application (Distributed)." The distributed application performs a portion of the task that an application, such as application server **220** or application client **210**, perform. In the present example, tasks such as volume rendering may be broken up and easily performed among two or more computers. These computers can be remote from each other on network **206**. Thus, several computers, such as server computer **204** and additional computers **222** and **224** can all work together to perform the task of computing a new viewpoint and frame buffer for the embryo for the new orientation of the embryo image in the present example. The coordination of the distributed processing can be performed at client computer **200** by application

client **210**, at server computer **204** by application server **220**, or by any of the distributed applications executing on additional computers, such as **222** and **224**. In a preferred embodiment, distributed processing is coordinated by a program called "VIS" represented by application client **210** in FIG. **6**.

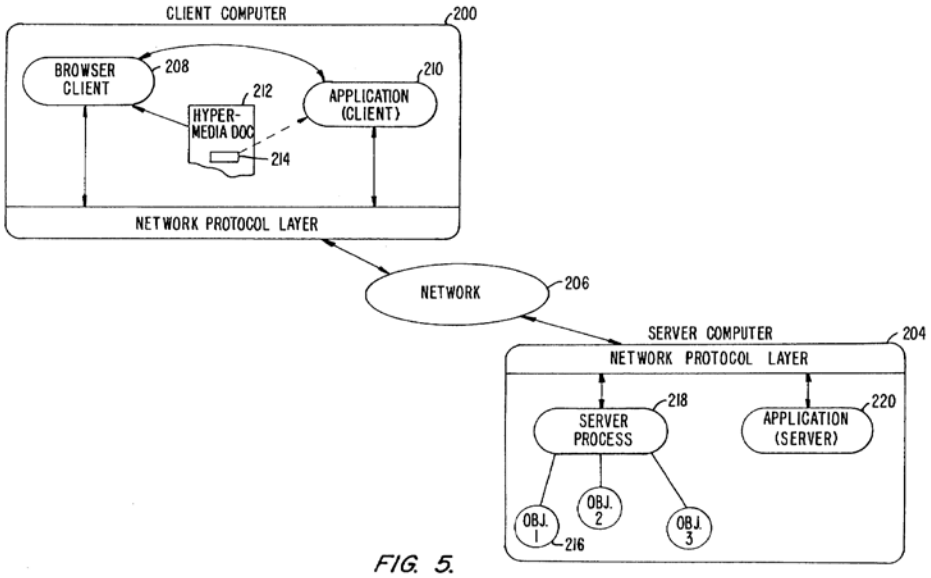


FIG. 5.

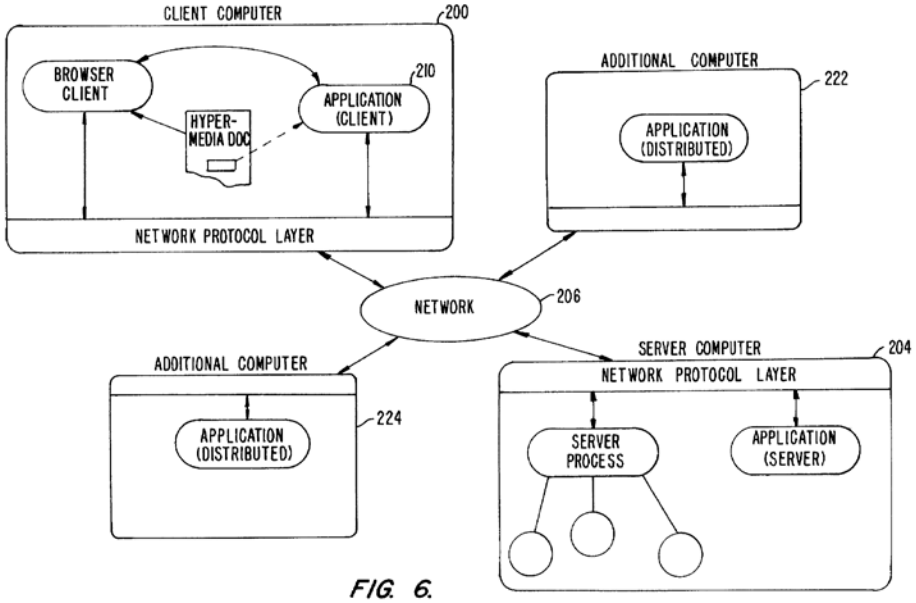


FIG. 6.

c. Prosecution history

i. '906 prosecution history (08/324,443)

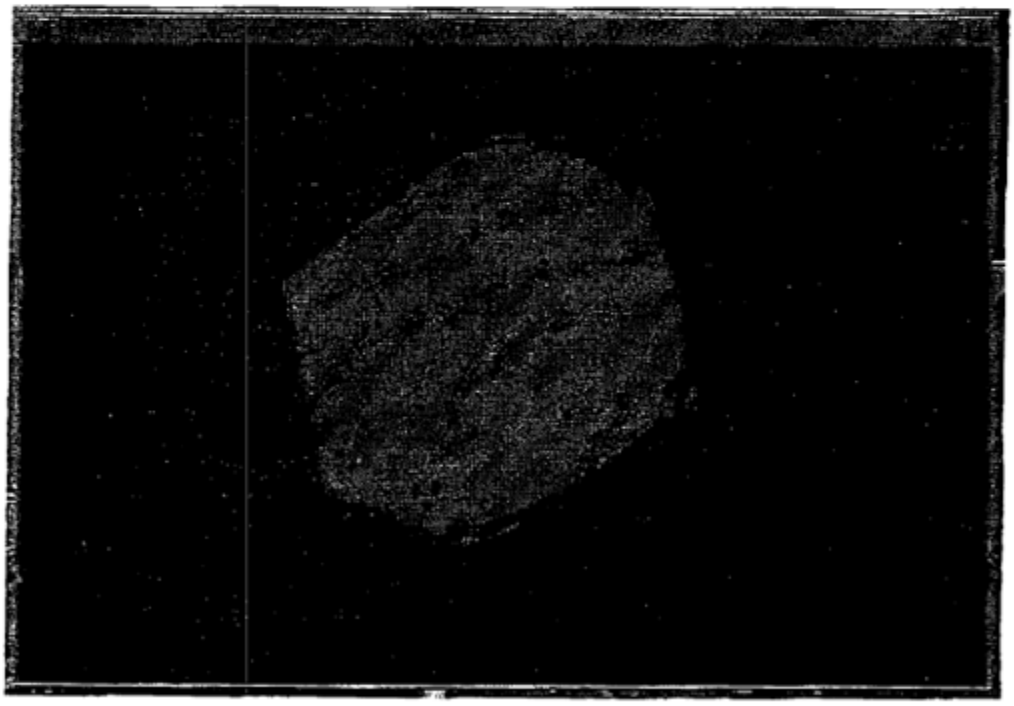
Declaration of Michael D. Doyle, at Attachment A (Jan. 2, 1997) (accompanying Applicants' Response (Jan. 8, 1997)) (906 PH Ex. 05 at PH_001_0000783945-55):

M.D. Doyle, et. al., *The Virtual Embryo: VR Applications in Human Developmental Anatomy,* presented at "Medicine Meets Virtual Reality II, Interactive Technology and Healthcare: Visionary Applications for Simulation, Visualization & Robotics," sponsored by the UCSD School of Medicine and the Advanced Projects Research Agency, San Diego, CA, **January 27, 1994.**

Video Presentation Transcript:

This is a status report of some of the work that's been accomplished during the first years of the Visible Embryo Project.

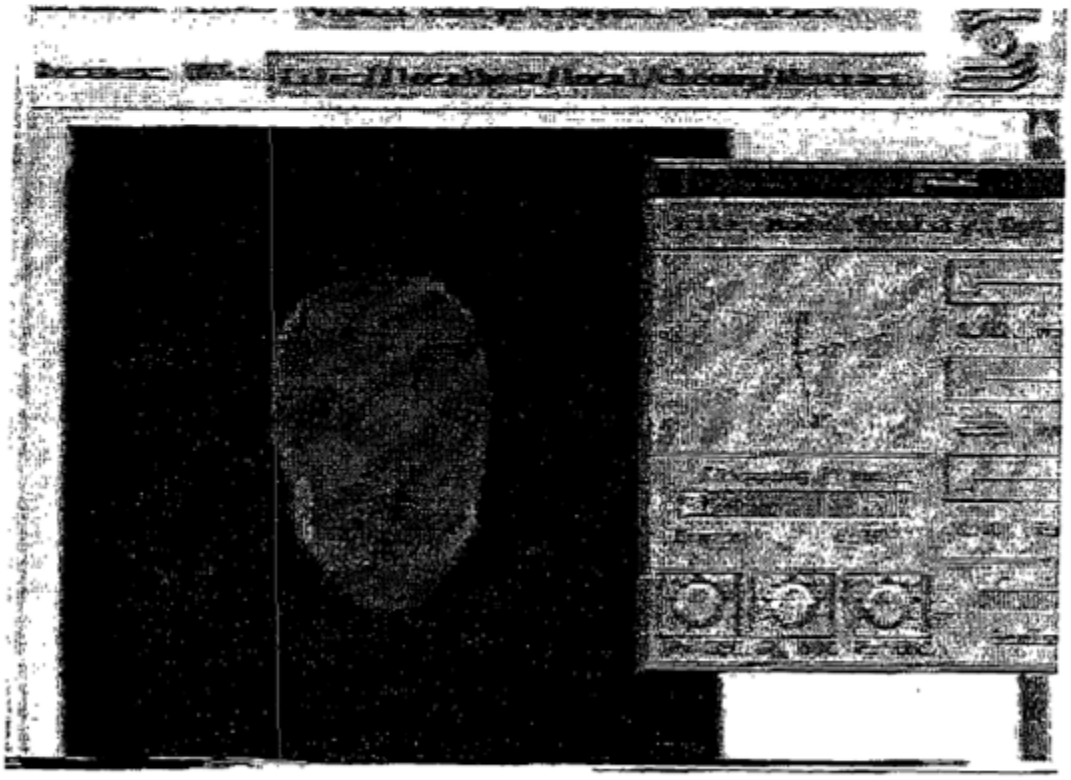
One of our first tasks was to develop some volume visualization software that we could use for imaging and analysis of the embryo reconstructions that we planned to create during the full term of the embryo project. One thing that was an absolute requirement was that this software be able to distribute its computational load across a network of graphics computers that weren't necessarily all in the same place. Basically we wanted to be able to have computers that could be all over the country connected by high -speed networking, able to contribute to a computation of three-dimensional datasets.



What you see here is a package called "VIS," which was developed in our group, for three-dimensional volume visualization. This is a very portable

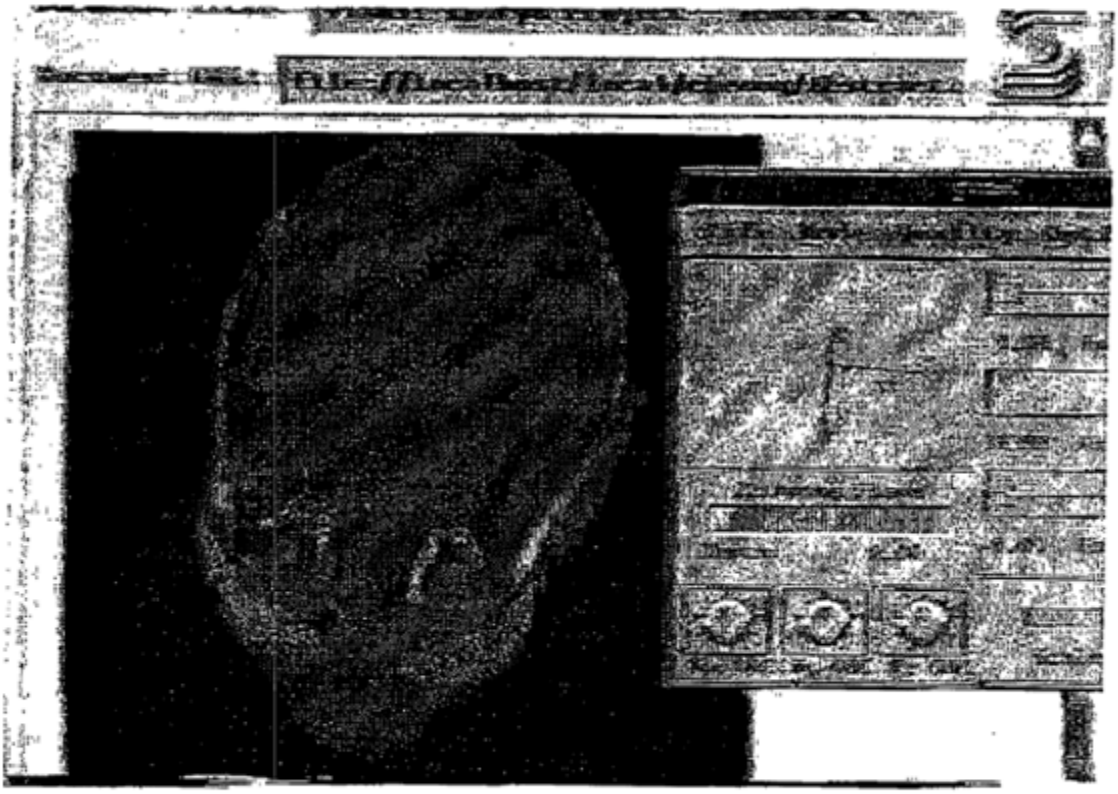
package that has been generated using as generic code as possible, although this particular image that you see here is running on a Silicon Graphics Reality Engine II, which is optimized for volume visualization. We need to use that kind of high-speed optimization to accomplish the real-time interactivity that we need to accomplish for this project. And as you see, as this rotates, and it starts rotating faster and faster, only a very powerful graphics - basically a graphics supercomputer - can accomplish this much computation on a three dimensional dataset. One of our goals is to allow anybody on the Internet with a very low-level access workstation to accomplish this kind of interactivity through their network connection, and the way that we do that is through a client-server architecture where we have a very powerful server computer accessed by a very low-end client machine.

We decided early on to use NCSA's Mosaic program and the World Wide Web to integrate access to this system. One problem with Mosaic is, as it exists today, is that the images within Mosaic are typically static or passive-playback images.



What you see here is an enhancement that we've created to the Mosaic interface and control software that allows the embedding of a dynamic real-time

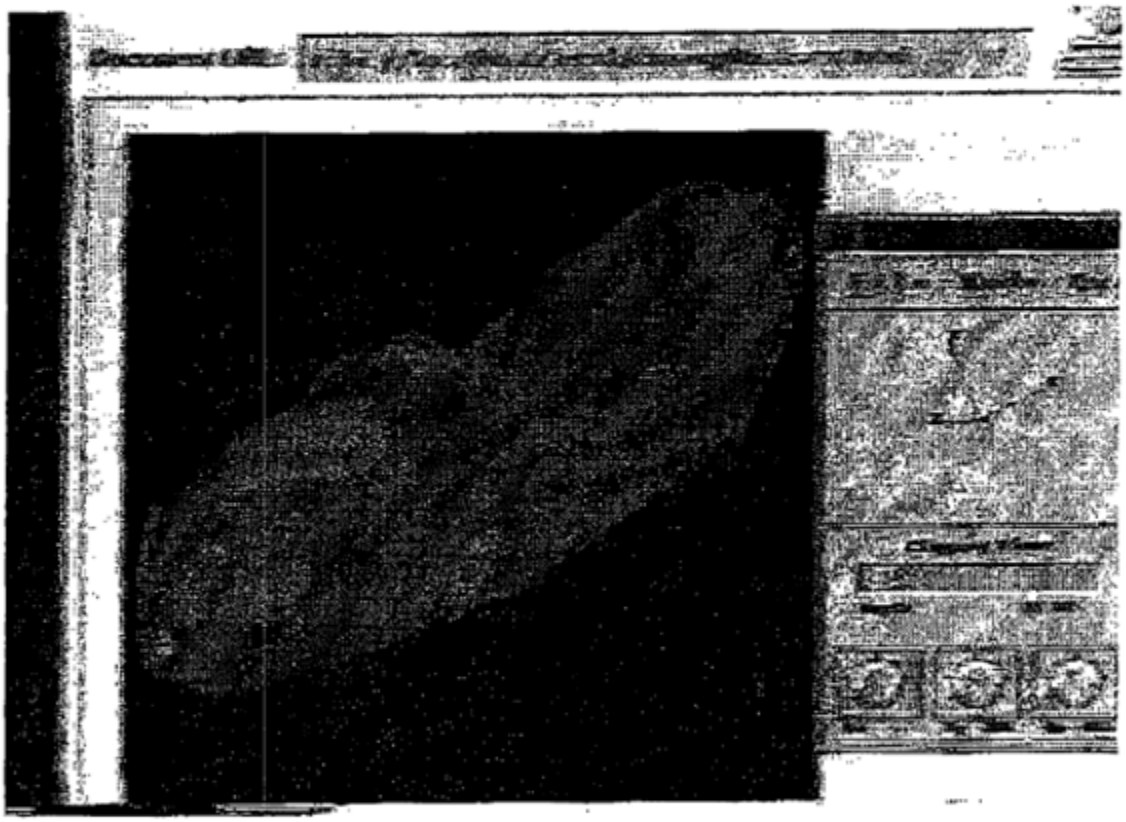
visualization application within a Mosaic document. You see a head, a volume MRI model of a human head, that's being rotated in real-time interactively by the viewer. You see a little panel to the right which is, it's a control panel that's popped up - it's really external to Mosaic itself but it can talk to the internal control programs that drive the Mosaic client - and allow you to interactively control the display from within Mosaic. This is actually controlling the volume visualization software that you just saw a few minutes ago.
By moving around the controls in the control panel we can do things like rotate, we can control slices through the dataset, and so on. As you can see, there's rotation in x, y, and z planes. We can also compute arbitrary oblique sections through the data and look at the internal anatomy. Here we can see the brain of this individual; we can rotate and view that section of the dataset from a variety of vantage points. There are zooming capabilities that allow us to zoom up on the data or zoom back to look at things in more or less detail, and you can see here that once zoomed, we're moving our cutting plane down through the dataset and looking at more and more inferior levels.



Normally graphics in Mosaic are static, as I said, but this embedding of graphic applications within Mosaic is really going to form the basis of how we integrate information access through the Visible Embryo Project. What you are about to

see is our prototype system of a Visible Embryo Mosaic document that has embedded realtime visualizations within it.

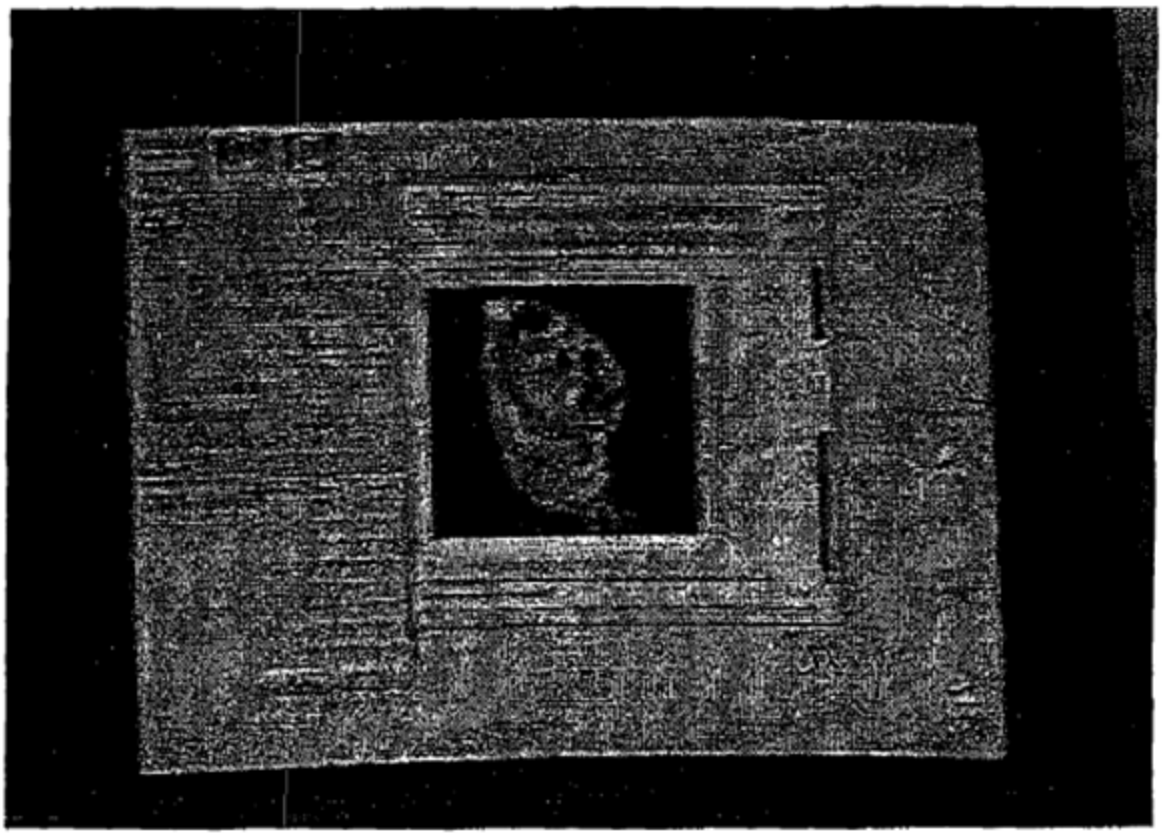
We just loaded the Visible Embryo Mosaic page, the system is about to scroll down this page - it is an abstract about the Visible Embryo Project - and then we see a window. It looks like a static image just like is normally found within the World Wide Web databases that you can access today through Mosaic, but you can see that suddenly, by moving controls on the control panel, we can zoom in and see that this is a reconstruction of a seven week old human embryo. This is a reconstruction from approximately 2900 serial cross sections of an embryo sectioned about in the 1930s. It's part of the Carnegie Collection of Human Embryology.



We're looking at this in volume visualization mode, we can rotate the embryo around, we can see internal structures, neurological structures; just in the lower abdomen area, we can see the liver, the arms are very evident - so we are actually looking through the dataset. We can also slice through this dataset obliquely, and look at the internal anatomy that way as well. We can load a volume visualization table that allows us to interactively enter tissue characteristic numbers that control the translucency, transparency, or opacity of various ranges of voxal intensity. And what we've done now is we've made the

exterior of the embryo a little more opaque so that as we rotate around with an oblique cutting plane we can see the difference between the cutting plane and the exterior of the embryo a little better. So now we're looking, slicing - we've rotated the embryo to an inferior view and we're looking up at the embryo from below. And we can see, we've just gone through the heart region, we're going through the liver, we're moving inferiorly and we start to get to an area where we can see the herniated gut.

Now the real key to all of this is that these are embedded visualizations. We're actually creating documents that are - I guess you'd call them currently compound documents where you have the traditional type of information, but you've also got, within that document, links to the raw data rather than just pictures generated from data. This allows you to tie together representations of data with the actual data themselves as well as with notes and different kinds of descriptive textual information based on that data.



Our basic objective here is really to create what we're calling a national meta-center which is going to be a computational resource for the entire nation that allows people interested in many different areas, including developmental biology, also multi-dimensional imaging and high-speed networking, and parallel computing. All these people can access this database. And the parallel

nature of the computation that's taking place is invisible to the user. They log in through a Mosaic window, and that window is giving them very high-performance control of interactive visualizations of datasets. By scrolling the window, we can see that this is actually embedded within the Mosaic document.

During a recent demonstration of this technology at the corporate briefing center at Silicon Graphics Corporation in Mountain View, California, I discussed some of the implications of this technology for researchers of human genetics.

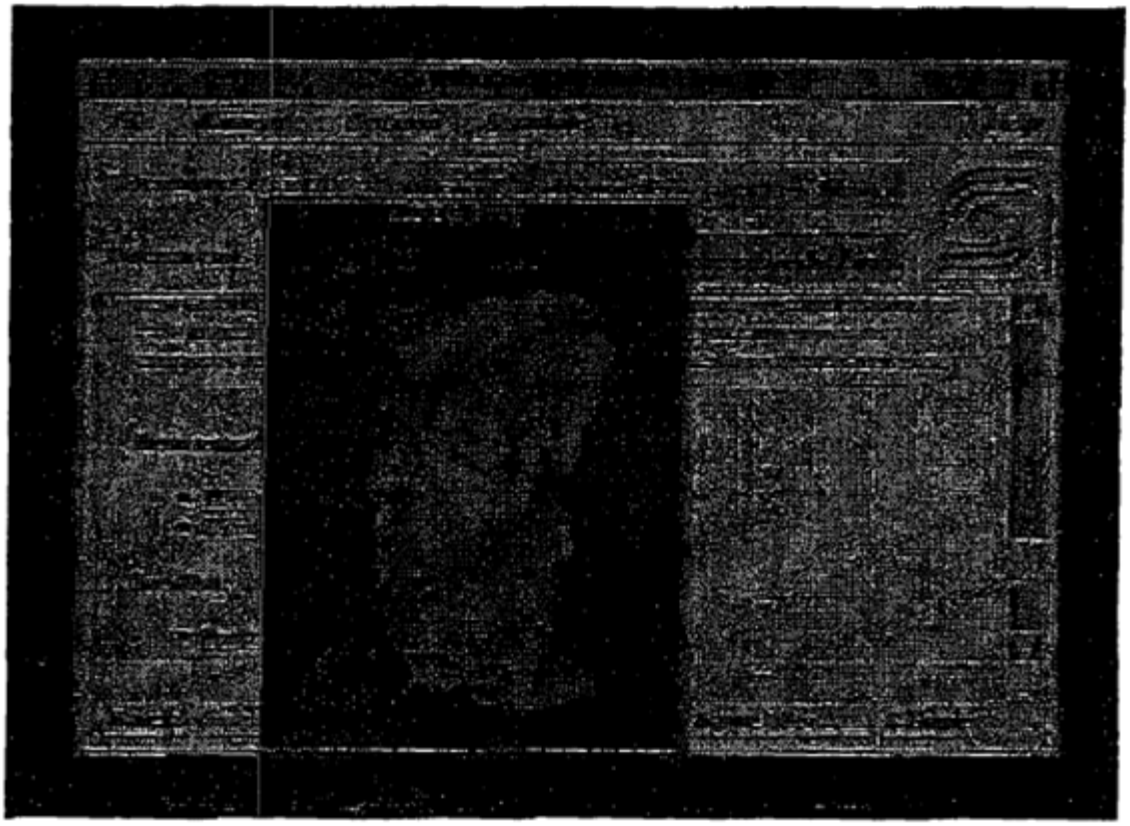
"We're also looking at using these models as a basis for creating three-dimensional maps of gene expression, which is a way to correlate the findings of the Human Genome project within a context that everyone uses. It sort of sets up a standard space within which everybody can report their findings, so that you can finally have some way of comparing studies that happen in different laboratories.

If you're studying the three-dimensional distribution of gene expression of a gene relating to heart development, what you do now is you have a little fluorescent marker that glows under an ultraviolet light, and you use confocal

microscopy to develop a three-dimensional model of it, and then you say, well, it's on here and it's on there and it's on there and you try to describe it in anatomical terms but it is a qualitative description, right?.

But here there would be a standard anatomy space that people could use to describe their findings so that they could, rather than say, yeah, we saw five different studies that said that this was expressed at the bifurcation of the aorta with the Common Carotid artery - you don't have to do in terms of verbal descriptions, you can do it in terms of a true measurable Cartesian coordinate system.

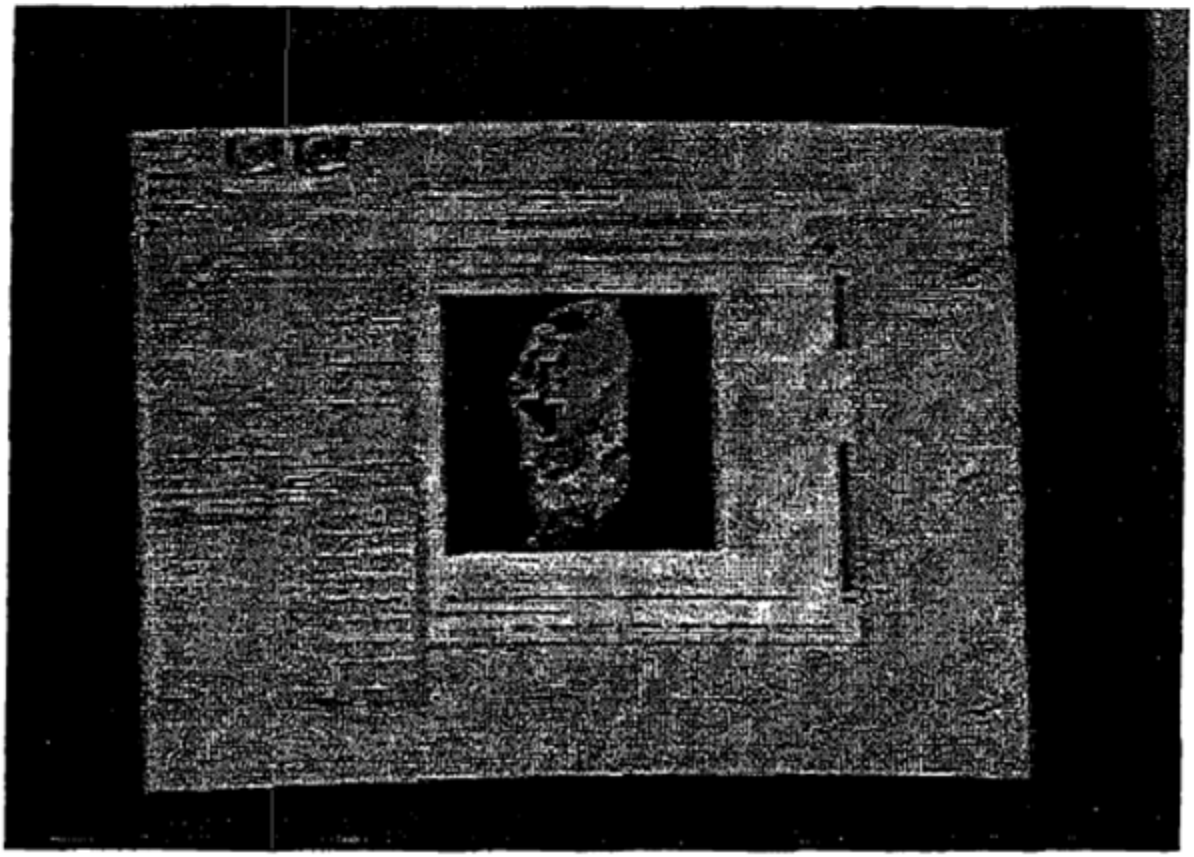
If you take your current version of Mosaic, the kind that is accessible for free through the Internet today, and log onto our home page of the Visible Embryo Project, what you'll see is a series of multi-media documents that basically give you information about the status of the Visible Embryo Project and the status of our current proposal development efforts. You'll be able to load MPEG movies of visualizations of human embryos, as you can see here.



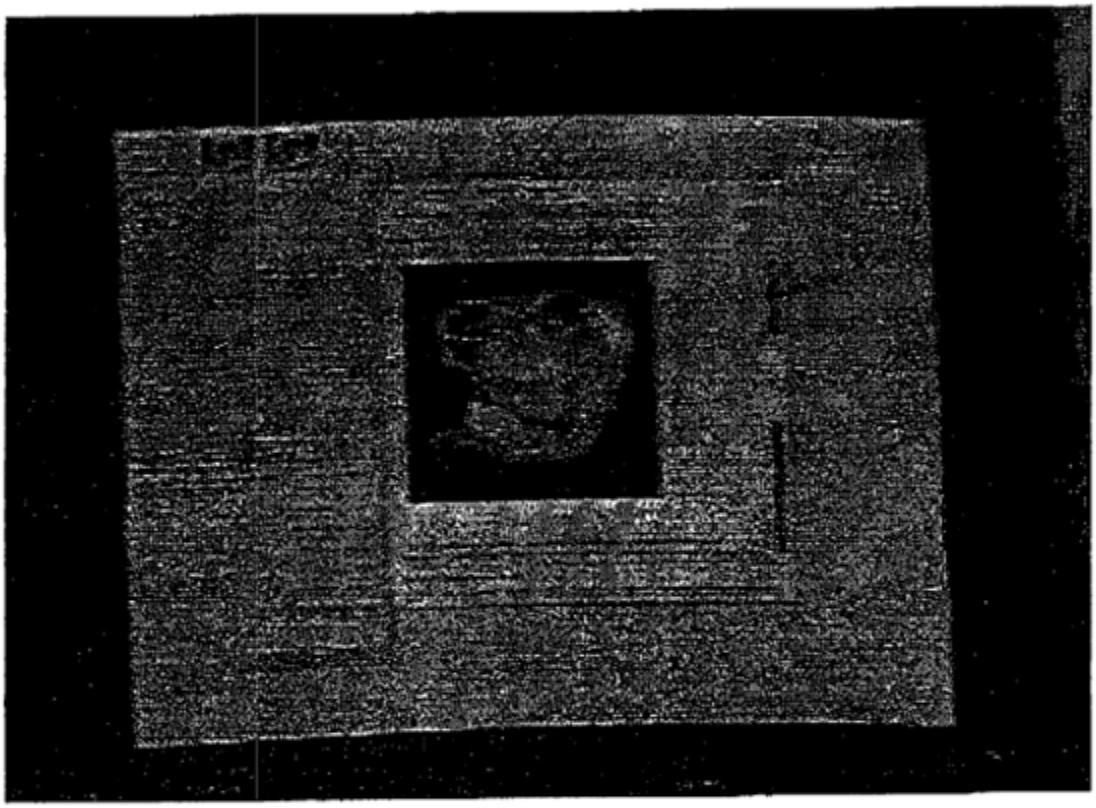
One thing you should keep in mind is that this little MPEG movie is just a canned movie, it's not interactive. Once you hit Play it just goes and it plays and then it goes away, but you can't stop it and interact with it, and rotate that embryo, for instance, to different vantage points.

Also available is an image that shows some of the early work, some screen shots representing the volume visualization tool current as of about last summer. We've come much farther and in fact a lot of the video that you saw earlier in this presentation shows you the current status of this volume visualization package. Last summer, that imaging package was separate from Mosaic, as you see it's off to the right, and Mosaic could just call it but couldn't actually embed visualizations. Now, everything is tied together into a single multi-media document.

You'll also see articles that relate to the Visible Embryo Project. This project has been going on for several years now, mostly on the coattails of other research projects, collaborators funding it wherever they could find the money. But already a significant body of literature is starting to be built up around this project.



Of course, in the very near future, you'll be able to log on through an enhanced version of Mosaic. **We're working together with the National Center for Supercomputing Applications on enhancing the standard release of Mosaic to allow these capabilities.** And you'll be able to access interactive dynamic visualizations that are being served by a network of high-end supercomputers across the country. Even if you're only accessing the system with a machine like a Macintosh or PC, you'll still be able to access the power of these supercomputers from your own location.



What I've attempted to demonstrate in the last several minutes in this presentation is that there's been a considerable amount of work already done in this project that we call the Visible Embryo Project. Many collaborators across the country have worked together to create a set of enabling technologies to allow this project to accomplish its goals. The Visible Embryo Project represents an effort to serve the needs of both the biology community as well as the information science community, in that we are attempting through current applications in information technology to break through barriers that have prevented biological researchers from asking and answering questions about the most fundamental mechanisms of human growth and development. We're also creating a technology development testbed for the information sciences that will allow researchers to push the envelope, so to speak, of information technologies to their very limits.

SPEAKER CONSENT FOR AUDIO/VIDEO RECORDINGS

We would like to audio/video tape your presentation at **Medicine Meets Virtual Reality II: Interactive Technology and Healthcare,** January 27-30, 1994.

Audio and video tapes will be available for immediate distribution to attendees and will be marketed and sold after the conference. Your colleagues will be able to benefit from your remarks by listening to cassettes and viewing tapes whether or not they were in attendance. The tapes will be copyrighted and marketed under Title 17 of the U.S. Code or other law as may be enacted, and all rights to royalties, if any, in conjunction with cassette sales shall hereby be assigned to Medicine Meets Virtual Reality. This agreement does not preclude the publication by you of your paper, speech or comments at any time and in any format.

Please cooperate with the audio/visual taping staff to obtain the best possible recordings by using the microphone at all times and repeating the questions that are asked by persons not near a microphone.

Sign this form below and return it to us at your earliest convenience.

Thank You.

I hereby give permission to ***Medicine Meets Virtual Reality*** to record and distribute audio and video tapes containing my presentation as outlined above.

Signature Michael D Doyle Date 11-30-93

Please print name Michael D. Doyle

Thank you very much for your cooperation. Remember to pick up the complimentary audiotape of your presentation before leaving the meeting.

Please return this completed form to:
MEDICINE MEETS VIRTUAL REALITY
P.O. Box 23220, San Diego, CA 92193
For further information call 619/751-8841,
or Fax 619/751-8842.

Applicants' Response, at Attachment I (Jan. 8, 1997) (906 PH Ex. 06 at PH_001_0000783961-68):

NSF/ARPA/NASA Digital Libraries RFP Response

Title: A Knowledge Managment Environment through the World Wide Web

Principal Investigator: Michael D. Doyle, Ph.D., UCSF Library and Center for Knowledge Managment

Specific Aims:

1) To develop a prototype knowledge management environment for the biomedical sciences which integrates access to online representations of the scientific literature, bibliographic databases, high-performance visualization technologies, large-scale scientific databases, and tools for authoring new-generation scientific publications.

1.a) To explore and evaluate the applicability of these tools in the areas of radiology and developmental & molecular biology.

2) To provide a means for relating digital forms of spatial, functional, and conceptual information as a basis for linking the biomedical scientific literature, through the Red Sage electronic journals project, to data resources provided through the Visible Human Project, The Human Brain Project, The Visible Embryo Project, The Human Genome Project, The Protein Database, and other large-scale biomolecular and biostructural databases.

2.a) To exploit these linking strategies in the creation of a set of integrated semi-automatic front ends to varied scientific databases accessible through the Internet.

2.b) To incorporate these linking methodologies into interactive authoring and editorial tools, allowing the creation of online publications that can embed visualizations and simulations which draw data from these Internet-accessible scientific databases.

3) To develop tools which provide access to interactive visualization and analysis of massive biomedical datasets through the Internet's World Wide Web distributed hypermedia network.

3.a) To refine and extend our existing algorithms enabling distributed visualization and analysis software "engines" which can be efficiently accessed by remote users through the Internet.

3.b) To refine and extend our existing algorithms to allow the display and real-time interactive control of three-and four-dimensional data visualization and analysis tools within hypermedia documents viewed using NCSA's Mosaic graphical browser to the World Wide Web.

3.c) To develop algorithms which use novel compression technologies for the optimized interactive remote control of computationally-intensive graphical applications through the Internet.

3.d) To integrate a,b & c into a system which allows real-time remote access to distributed parallel computational applications for visualization and analysis resources within a distributed hypermedia environment.

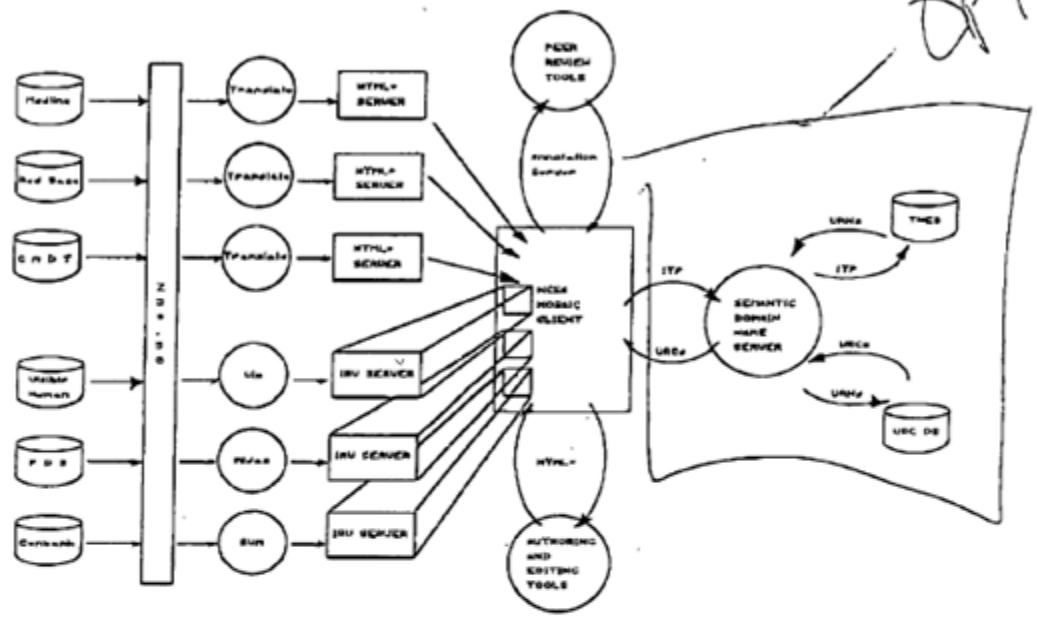
4) To explore extensions of the paradigm of scientific publishing which are made possible through use of current multimedia technologies in a networked environment, including:

4.a) publishing multidimensional datasets integrated with articles, eg: MRI and molecular data, perferred views, animations, interactive visualizations, interactive mathematical models, and

4.b) development of scientific authoring tools for publications which exist only in the networked environment.

4.b.1) This will include integration of HTML+ WYSIWYG authorial and editorial tools, multidimensional data visualization applications, molecular modelling and database management tools into an interactive scientific publishing environment.

System Diagram:



Definitions:

HTML+:	Hypertext Mark-up Language -- This is the language that World Wide Web databases are encoded in, and that Mosaic interprets.
IRV Server:	UCSF CKM's Interactive Remote Visualization Server -- This allows interactive real-time visualization tools to be embedded into Mosaic documents.
Vis:	UCSF CKM's distributed remote volume visualization tool
Midas:	UCSF CGL's molecular visualization package
SVM:	Sequence Visualization Module -- An as-yet unamed tool for graphical display of genetic sequence data.
ITP:	Informal Text Phrase -- A user-entered search term, or a word or phrase of text that the user highlights from within a document.
URN:	Universal Resource Name -- A persistent, location-independent identifier for an object.
URL:	Universal Resource Location -- The address of an object. It contains enough information to identify a communications protocol and retrieve the object.
URC:	Universal Resource Characteristics -- Any combination of one or more URNs or URLs with meta information (e.g. author, format, compression method).

Description:

The system will draw from of a number of fundamental databases including bibliographic data (Medline) in the form of MARC records, journal publication data (Red Sage) in the form of SGML header and Postscript files, encyclopedic reference text data (CMDT) stored in an object-oriented SGML database, volumetric anatomical data (Visible Human Project) stored as NCSA HDF datasets, protein structure data (Protein Data Bank) stored as PDB files, and genetic sequence data (Genbank) stored as compressed ASCII strings (?, I'm guessing about Genbank).

These databases will reside behind a Z39.50 interface layer which yeilds, to the requesting client, the respective dataset in its native form. This data then goes through a translation layer where the data is either translated directly into HTML+ (Medline, Red Sage, CMDT) or loaded into a native-data visualization tool (Visible Human, PDB, Genbank). The HTML+ code is then passed to a set of HTML+ servers, which can be browsed by the Mosaic client. The visualization data is handled differently. The graphical I/O of the relevant visualization tool is passed to an interactive remote visualization (IRV) server, which handles both mapping of the display output from the visualization tool onto embedded live-visualization windows within the Mosaic-browsable HTML+ documents, as well as capture of user-entered mouse and keyboard events within the visualization windows and transmission of those mouse and keyboard events back to the relevant visualization tools. The user, browsing the system with the project's enhanced version of the Mosaic client, is presented with data and visualizations derived from these various databases, yet embedded into coherent, multimedia Mosaic documents.

For multimedia documents that have been explicitly pre-composed, the linking of these various data resources can take the form of universal resource names (URNs) that are encoded as tags into the HTML+ documents. This is passed to the system's semantic domain name server, for resolution of the information object's location and retrieval means. The URNs are used as indices in order to look up the relevant universal resource characteristics (URCs) in a URC database, which yeilds the universal resource location (URL), or physical adress, of the information object in question.

Semi-automatic means will be provided for a user to search for arbitrary information objects on the system by either keying in a search word or phrase, or by highlighting a not-already-hyperlinked section of text that (s)he happens to be viewing within the Mosaic client at the time. This informal text phrase (ITP) is then passed to the semantic domain name server, which passes it on to a universal resource thesaurus (which will incorporate elements of the NLM's UMLS system). The thesaurus compares the ITP to its database of terms and phrases and returns a rank-ordered list of URNs that are likely to match the object in question. These URNs are then passed to the URC database for resolution of URLs that point to information objects on the Internet that are most likely to match the ITP that the user employed to initiate the search. The user is presented with a rank ordered set of textual descriptions of likely matches which are hyperlinked, via their URLs, to the data in question. Clicking upon a selection from this list loads the related data into the relevant visualization server (IRV) or HTML+ server, and a second Mosaic window pops up to allow viewing or interaction with that dataset.

A set of authoring and editing tools will be designed to allow the interactive WYSIWYG creation of HTML+ documents, as well as allowing the embedding of visualizations, etc., which can be created using the interactive remote visualization tools, and which can use data from the various scientific databases mentioned above. Alternatively, the author can use his/her own datasets, which would be uploaded to an Internet-accesible World Wide Web server. The journal editor can use the same set of tools to edit submitted articles and to communicate changes to the text with the author. This, of course, would occur in a private, access-controlled, area of the system, so that confidentiality of the material to be published can be controlled.

Other private, access-controlled HTML+ servers will be used to administer the peer review process. A modification of NCSA's Mosaic-based group annotation server will be developed to allow the jounal editor to exercise precise control and documentation of each reviewer's comments and suggestions.

Contributions:

UCSF CKM:

- Development of Z39.50-compliant experimental (subset) databases for storage of Visible Human data, PDB data, and Genbank sequence data.
- Cooperation with AT&T in the development of an object-oriented SGML-based database for the Handbook of Current Medical Diagnosis and Treatment (CMDT)
- Development of an experimental Z39.50 interface to Medline data (will be unnecessary if UC's DLA can provide such an interface to Melvyl Medline early enough into the project timeline)
- Development of translator servers to translate Medline MARC records, CMDT SGML data and Red Sage SGML/Postscript data into HTML+
- Development of a set of HTML+ documents that act as browsers to Medline, CMDT, and Red Sage
- Refinement and further development of Vis to allow better distribution of computation and better integration with Mosaic.
- Cooperation with CGL to adapt Midas for integration within Mosaic, and to identify and adapt a suitable program for graphical display of genetic sequence data.
- Refinement and further development of the interactive remote visualization server, and its incorporation (with NCSA's help) within the Mosaic environment.
- Development, in cooperation with NCSA, of an enhanced version of the Mosaic client to allow easier integration of external programs within Mosaic-readable documents.
- Development, in cooperation with Springer-Verlag and NCSA, of an interactive WYSIWYG editor for creation of HTML+ documents, and for embedding visualizations created using CKM's IRV tools, as well as development of a modified version of NCSA's group annotation server to support the peer review process.
- Development, in cooperation with AT&T, of an object-oriented SGML-based URC database
- Development, in cooperation with UCSF's CGL, UCSF's Radiology Dept., Washington Univ., and AT&T, of a Semantic domain name server and a URN Thesarus, based upon AT&T's object-oriented SGML database technology.
- Development, in cooperation with UCSF's CGL, UCSF's Radiology Dept., Washington Univ., and Springer-Verlag of a set of sample content for use in evaluating the effectiveness of the system, as well as for demonstration of the results of the project.

UCSF CGL:

- Cooperation with CKM to adapt Midas for integration within Mosaic, and to identify and adapt a suitable program for graphical display of genetic sequence data.
- Contributing to the refinement and further development of the interactive remote visualization server, and its incorporation (with NCSA's help) within the Mosaic environment.

- Development, in cooperation with UCSF's CKM, UCSF's Radiology Dept., Washington Univ., and AT&T, of a Semantic domain name server and a URN Thesarus, based upon AT&T's object-oriented SGML database technology.

- Development, in cooperation with UCSF's CKM, UCSF's Radiology Dept., Washington Univ., and Springer-Verlag of a set of sample content for use in evaluating the effectiveness of the system, as well as for demonstration of the results of the project.

Washington University:

- Development, in cooperation with UCSF's CKM, UCSF's CGL, and AT&T, of a Semantic domain name server and a URN Thesarus, based upon AT&T's object-oriented SGML database technology.

- Development, in cooperation with UCSF's CKM, and UCSF's CGL., and Springer Verlag of a set of sample content for use in evaluating the effectiveness of the system, as well as for demonstration of the results of the project.

AT&T Bell Laboratories:

- Development of Z39.50 interface to the RightPages server..

- Cooperation with CKM in the development of an object-oriented SGML-based database for the Handbook of Current Medical Diagnosis and Treatment (CMDT)

- Development, in cooperation with CKM, of an object-oriented SGML-based URC database

- Development, in cooperation with UCSF's CGL, UCSF's Radiology Dept., Washington Univ., and CKM, of a Semantic domain name server and a URN Thesarus, based upon AT&T's object-oriented SGML database technology.

Springer-Verlag:

- Development, in cooperation with UCSF's CKM and NCSA, of an interactive WYSIWYG editor for creation of HTML+ documents, and for embedding visualizations created using CKM's IRV tools, as well as development of a modified version of NCSA's group annotation server to support the peer review process.

- Development, in cooperation with UCSF's CKM, and UCSF's CGL., and Washington Univ. of a set of sample content for use in evaluating the effectiveness of the system, as well as for demonstration of the results of the project.

NCSA:

- Cooperation with CKM in developing an enhanced version of Mosaic to allow easier integration of a client modeule for CKM's interactive remote visualization server.

- Cooperation with CKM and Springer-Verlag in the modification of NCSA's group annotation server to facilitate the peer-review process.

Personnel:

Co-Investigators:

UCSF:

Library & CKM: Richard Lucier, David Martin, Zoe Stavri, Ph.D., Cheong Ang, Marc Salomon
Radiology: Tom Budinger, Ph.D.
Molecular & developmental Biology: Tom Ferrin, Ph.D., Charles Ordahl, PhD.

Washington University (molecular biology): Toni Kazic, PhD
Bell Laboratories: Ed Szurkowski, Guy Story
Springer Verlag: Bob Badger, PhD
NCSA: Joseph Hardin, PhD, & Mosaic development group
SFSU: Computer Science Dept. MS students

Timetable: 4 years

Budget: 1.2 $M/year

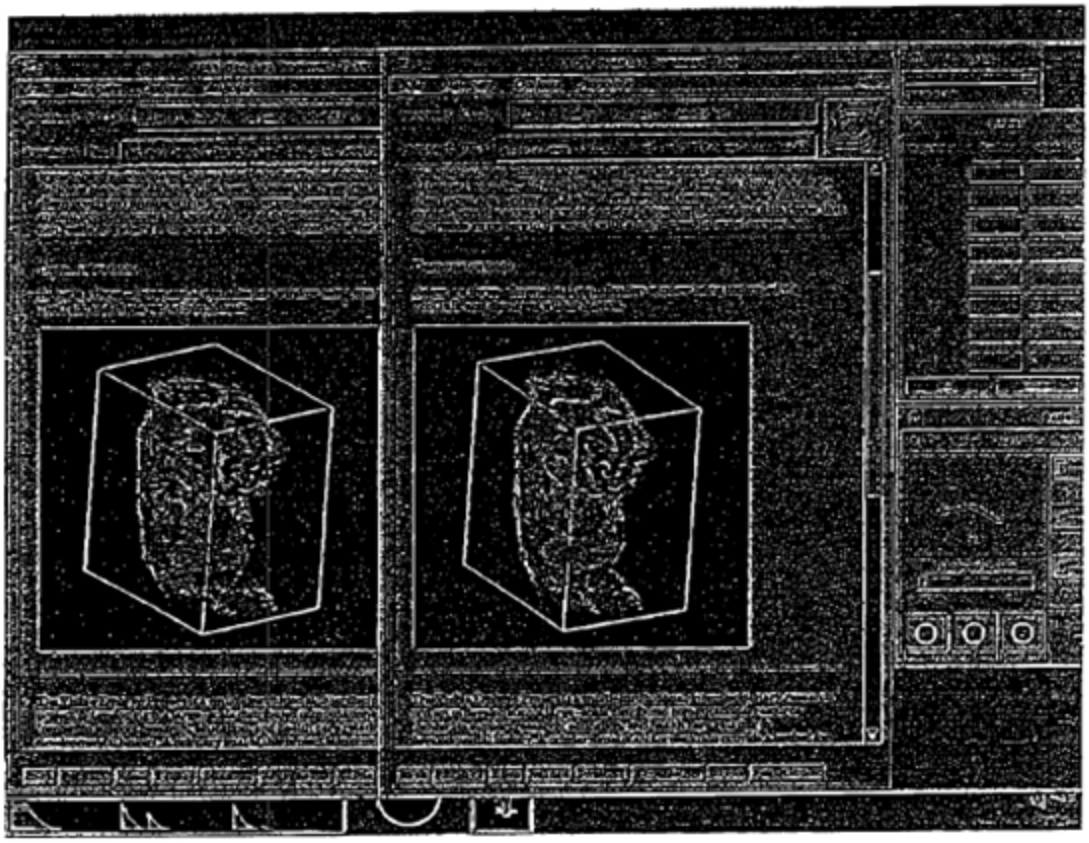


Figure 1: A stereo-pair illustration of interactive real-time 3-dimensional human embryonic volume reconstructions embedded within an NCSA Mosaic document. This technology was developed by the Center for Knowledge Management at the University of California, San Francisco, and was demonstrated there in November, 1993.

JOSEPH HARDIN

National Center for Supercomputing Applications (NCSA)
University of Illinois at Urbana-Champaign
Champaign, IL 61820
217/244-6095
email: hardin@ncsa.uiuc.edu

Major work experience

University of Illinois,
Associate Director, Software Development Group, NCSA, 1992-present

University of Illinois,
Manager, Software Development Group, NCSA, 1988-1992
Coordinator, Academic Affiliates Program, NCSA, 1987 -1988
Visiting Research Associate, NCSA, 1986-1987

University of Georgia at Athens,
Visiting Instructor, Department of Speech Communication, 1985-1986

University of Illinois,
Teaching Assistant, Department of Speech Communication, 1979-1985
Teaching Assistant, Department of English-Rhetoric, 1978
Research Assistant, Department of Sociology, 1977-1978

Other work

Consultant -Business computer communications, database management, and office organization, 1982-1985

Consulting Editor/Contributor-The Champaign-Urbana Weekly news magazine, 1981-1982

Editor-The Champaign-Urbana Weekly news magazine, 1979-1980

Recent Grants and Awards

Exploratory Research and Initial Development of Software for the Analysis of Multiple Hybridization Images, 1990-1992

IBM Second Generation RISL Workstation Graphics Capabilities: Jointly Defined Effort, 1990-1992

NCSA Hierarchical Data File Software Capitalization: National Distribution and Support, 1991-1993

Research Education for Undergraduate Students Supplement, 1990-1993

Supercomputers for Biologists: Macromolecular Sequence Analysis through Distributed Computing, 1991-1993

Visualization in the MS DOS Environment: NCSA/Jackson State Collaborative Project, 1990-1993

x3d Program Development, 1992-1993

Education

Study toward Ph.D., Speech Communication, University of Illinois
Study Abroad, University of Cologne, Cologne, West Germany, 1978
B.A., History, University of Illinois, 1972

Selected Recent Conference Papers/Participation

Sigchi, Session Paper, "Scientific Visualization in a Collaborative Environment," Amsterdam, Holland, April, 1993 (in submission)

Sigchi, Session Paper, "Collaborative Hypermedia for Computational Science," Amsterdam, Holland, April, 1993 (in submission)

Oceanographic Institute, Old Dominion University, Invited Seminar, "Recent Developments in Collaboration Technologies," Norfolk, Va., Feb. 1993

Scientific Computing & Automation Conference, Invited Speaker, "Cross platform Digital Conferencing Software Development," Washington, DC. 1992

New York University, Academic Computing Facility, "Scientific Visualization in Collaborative Technologies," New York City, NY, 1992

International Institute of Ecological Economics Conference, Invited Workshop Participant, Beijer Institute, Stockholm, Sweden, 1992

American Educational Research Association Conference, "Collaborative Tools for Scientific Communication and Understanding," San Francisco, CA, 1992

Mac SciTech National Conference'92, "Interpersonal Computing for Computational Scientists: The NCSA Collage Series," San Francisco, CA, 1992

Office Action, at 5 (March 26, 1997) (906 PH Ex. 10 at PH_001_0000784017):

Claims. 2-6, 10-14, 45-48, 15, 17-23, 24-33, 34-43, 54, 55, and 56 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Applicant disclosed prior art, Khoyi et al, Hansen "Andrew as a Multiparadigm Environment for Visual Languages" and further in view of Moran "Tele-Nicer-Dicer: A new tool for the visualization of large volumetric data".

As per claim 2, the disclosed prior art does not disclose interactively controlling via communication sent over the distributed environment. Moran discloses a distributed application (TNSD) for interactive control and visualization of graphical object through communication over network. Moran

application allow usage of remote system resources for visualization of large data set at a client station. It would have been obvious for one of ordinary skill in the art to utilize Moran application as an external application ("Viewer") in the prior art system as modified because it would have improved the system by enabling the client station access to resources on higher performance servers and to have interactive visualization of large data set capability.

Amendment B, at 23-24 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784051-52):

The rejection of Claim 2 on Mosaic, Khoyi, Hansen and Moran is overcome

Applicants' Claim 2 recites the additional step over Claim 1 of interactively controlling the controllable application on the client workstation via inter-process communications between the browser and said controllable application. The disclosure of Mosaic, Khoyi, and Hansen has been described above. The reference Moran discloses a tool for interactive visualization of large, rectilinear volumentric data called Tele-Nicer-Slice-Dicer (TNSD). TNSD is based on clientserver design where the client-side process is an extended version of a stand-alone visualization tool and the server process runs on a high-performance system where the data are located.

The client-side process describes data sets by text fields. Each data set description is used as a command which is sent to the server when a volume from the corresponding data set is requested. The use of a remote server is transparent. The Examiner states that it would have been obvious to utilize the Moran application as an external application ("Viewer") in the prior art system as modified because it would have improved the system by enabling the client station access to resources on higher performance serves to have interactive visualization of large data set capability.

Neither Mosaic, nor Khoyi, nor Hansen shows an executable application which is external to a document being displayed and interactively processed within that document's display window, nor do they show such an application where said executable application is interactively controlled on said client workstation by interprocess communications between the external application and the browser. This feature produces surprising and unexpected results over the prior art, since it allows the reader to perform all necessary interactive functions with external applications without directing his or her attention away from the hypermedia document. Additional surprising and unexpected results are yielded by the fact that the hypermedia browser application can have its functionality extended without making any changes to the hypermedia browser's object code. Further, surprising and unexpected results come from the ability of the document author to design interactive hypermedia document content that displays a similar look and feel to the reader, regardless of what the underlying operating system or computer platform the browser program is being executed upon.

The amendments to these claims have made the Moran reference irrelevant to the case, since Moran teaches a remotely networked application being controlled via communications over a network, not an embedded (in a hypermedia document) interactive external application on the client workstation being controlled via inter-process communications between the document browser application and the external application.

Even if Moran was still in some way relevant, and even if the proposed combination was possible, was suggested by the prior art, and showed the features of the invention, all of which the above arguments for Claim 1 clearly show is not the case, the fact that a large number of references (more than 3) must be combined to meet the invention is further evidence of unobviousness

Office Action, at 4-5 (Aug. 25, 1997) (906 PH Ex. 12 at PH_001_0000784094-95):

Claims 3-4 and 46-47 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Applicant disclosed prior art and Koppolu et al. US patent 5,581,686, and further in views of Moran "Tele-Nicer-Dicer: A new tool for the visualization of large volumetric data".

As per claim 3, the disclosed prior art does not disclose interactively controlling via commands sent over the distributed environment. Moran discloses a distributed application (TNSD) for interactive control and visualization of graphical object through communication over network. Moran application allow usage of remote system resources for visualization of large data set at a client station. Moran discloses sending command to remote server, executing on the server, and sending result to the client to process and display [p.3 col.2-3 specifically col.1 3rd paragraph] . It would have been obvious for one of ordinary skill in the art to utilize Moran application as an external application ("Viewer") in the prior art system as modified because it would have improved the system by enabling the client station access to resources on higher performance servers and to have interactive visualization of large data set capability. As per claim 4, it is apparent that the system as modified would have instructions residing on the client workstation In order to provide the resulting graphic representation [NSD visualization tool [p.1 col.2 last paragraph].

Applicants' Response, at 30 (Dec. 23, 1997) (906 PH Ex. 16 at PH_001_0000784160):

The reference Moran discloses a tool for interactive visualization of large, rectilinear volumetric data called TeleNicer-Slice-Dicer (TNSD). TNSD is based on client-server design where the client-side process is an extended version of a standalone visualization tool and the server process runs on a high performance system where the data are located.

The client-side process describes data sets by text fields. Each data set description is used as a command which is sent to the server when a volume from the corresponding data set is requested. The use of a remote server is transparent.

The only relevance of Moran to the subject matter of claim 3 is the use of a network. There is no disclosure relating to HTML, the WWW, or embedding controllable objects' in a document displayed in a browser-controlled window.

The teachings of Mosaic-Koppolu (OLE), and Moran would not make the combination of claim 3·obvious. Such a combination is in violation of the requirement of 35 U.S.C. §103 because the combination requires taking isolated features from the references, utilizing applicant's disclosure as a roadmap, while ignoring the operation and purposes of the references.

ii. Second reexam (90/007,858)

Declaration of Michael D. Doyle, at Attachment A, page 12 (Sept. 22, 2007) (accompanying Applicants' Response (Sept. 27, 2007)) (858 PH Ex. 07 at PH_001_0000787088):

2.2 Distributed VIS

The VIS implementation has great potentials to run in parallel. We distributed the loads of volume rendering among workstations with a greedy algorithm that chooses to give bigger portions of the picture to faster machines. The client that VIS is running on will break the load into a relatively small portions, which are still big enough such that the overhead for network transmission of the data is substantially smaller than the time to transmit the actual data. Each of the computation servers will then fetch a portion for rendering via Remote Procedure Calls (RPC). A fast server will return the results earlier, and fetch another portion. The servers will compete with each other for data, and this ensures that most servers are busy most of the time. A report of analysis of this algorithm from Giertsen and Petersen (Giertsen, 93) shows that the performance improvement is a function of the number of load portions, and the number of computation servers. The test results suggest that the performance improvement max out when the number of sections of 512x512 and 1024x1024 pictures are around 10-20 on several volume datasets of various sizes on up to four servers. They also show that a larger number of workstations servers will subsequently enlarge the optimal number of sections. This approach may not be the best distributing algorithm, but it achieves the close-to-optimal results most of the time.

iii. '985 prosecution history (10/217,955)

Second Supplemental Amendment, at 68-69 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784639,81):

Wherein the executable application is part of a distributed application,

EXAMPLE SUPPORT:

10:33 "Another embodiment of the present invention uses an application server process executing on server computer 204 to assist in processing that may need to be performed by an external program. For example, in FIG. 5, application server 220 resides on server computer 204. Application server 220 works in communication with application client 210 residing on client computer 200."

11:18 "FIG. 6 shows yet another embodiment of the present invention. FIG 6 is similar to FIG. 5, except that additional computers 222 and 224 are illustrated. Each additional computer includes a process labeled "Application (Distributed)." The distributed application performs a portion of the **task** that an application, such as application server 220 or application client 210, perform. In **the** present, example, tasks such as volume rendering may be broken up and easily performed among two or more computers. These computers can be remote from each other on network 206. Thus, several computers, such as server computer 204 and additional computers 222 and 224 can all work together"

d. Cited prior art

Ang et al., "Integrated Control of Distributed Volume Visualization Through the World-Wide-Web." Proceedings of Visualization 1994, IEEE Press, Washington, D.C., October 1994, § 2 & §§ 5.3-5.4 including figs. 4-5 (Ex. O at [PH_001_0000759332] – [PH_001_0000759351]):

2. VIS: A Distributed Volume Visualization Tool

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 heterogenous pool of general purpose workstations and perform volume rendering at the request of the VIS process . The three modules are integrated as shown in figure 3 when cooperating with another process. A simple output window is displayed when no cooperating process is specified.

2.1 Distributed Volume Rendering

Volume rendering algorithms require a significant amount of computational resources. However, these algorithms are excellent candidates for parallelization. VIS distributes the volume rendering among workstations with a "greedy" algorithm that allocates larger portions of the work to faster machines [Bloomer]. VIS segments the task of volume rendering based on scan-lines, with segments sized to balance computational effort versus network transmission time. Each of the

user-selected computation servers fetches a segment for rendering via remote procedure calls (RPC), returns results and fetch another segment. The servers effectively compete for segments, with faster servers processing more segments per unit time, ensuring relatively equal load balancing across the pool. Analysis of this distribution algorithm [Giertsen, 93] shows that the performance improvement is a function of both the number of segments and the number of computational servers, with the optimal number of sections increasing directly with the number of available servers. Test results indictate that performance improvement flattens out between 10 to 20 segments distributed across an available pool of four servers. Although this algorithm may not be perfect, it achieves acceptable results.

2.2 Cooperative Visualizaton

The VIS client, together with its volume rendering servers, may be lauched by another application collectively as a visualization server. The two requirements of cooperation are a shared window system buffer for the rendered image and support for a limited number of inter-process messages. VIS and the initiating application communicate via the ToolTalk service, passing messages specifying the data object to visualize as well as options for visualization, and maintaining state regarding image display. The VIS Panel application appears as a new top-level window and allows the user control of the visualization tool.

5.3 Multiple Users

With multiple users, the VIS/Mosaic distributed visualization system will need to manage the server resources, since multiple users utilizing the same computational servers will slow the servers down significantly. The proposed solution is depicted in Fig 5. The server resource manager will allocate servers per VIS client request only if those servers are not

overloaded. Otherwise, negotiation between the resource manager and the VIS client will be necessary, and, perhaps the resource manager will allocate less busy alternatives to the client.

5.4 Load Distributing Algorithm

Since the load distributing algorithm in the current VIS implementation is not the most optimal load distribution solution, we expect to see some improvement in the future implementation, which will be using sender-initiated algorithms, described in [Shivaratri].

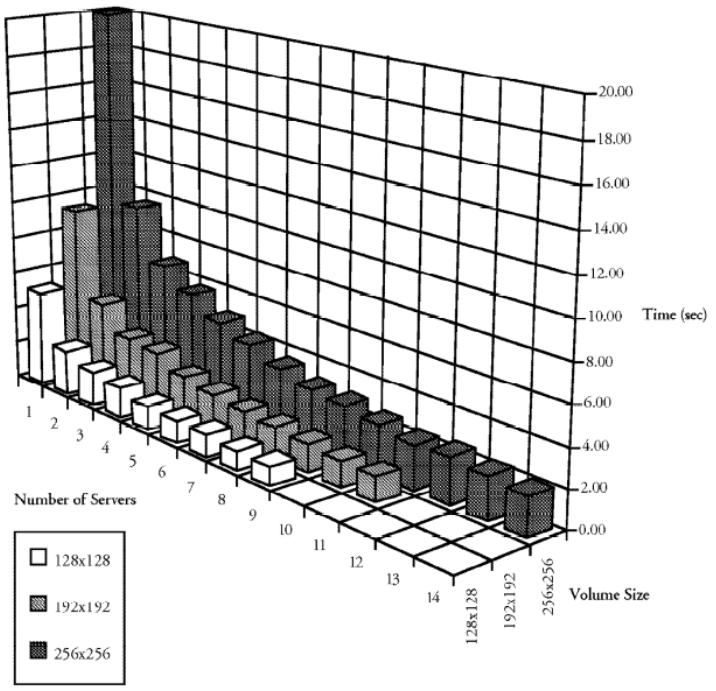


Figure 4: Volume rendering performance for 128^2, 192^2, and 256^2 data sets .

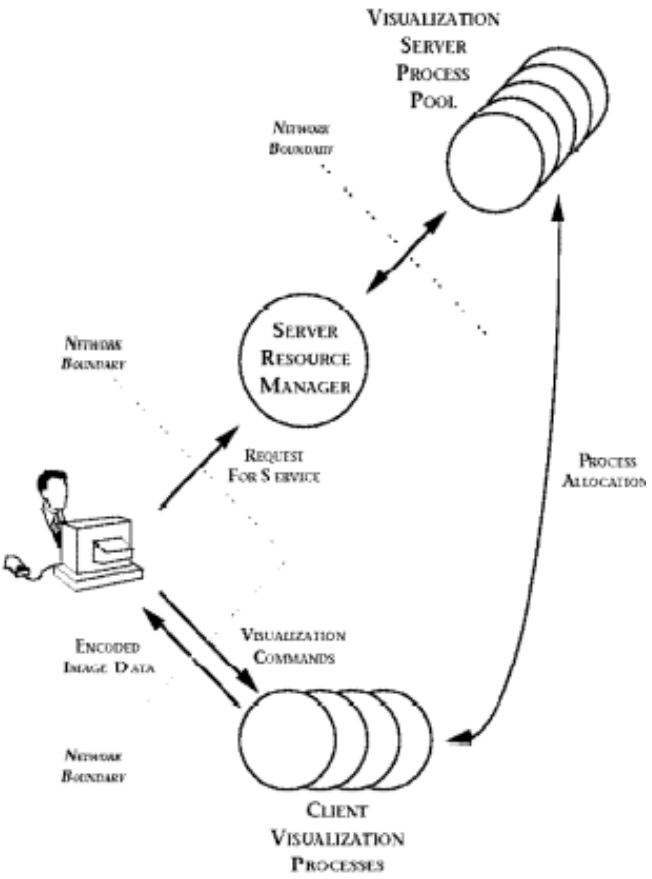


Figure 5: Server Resource Management

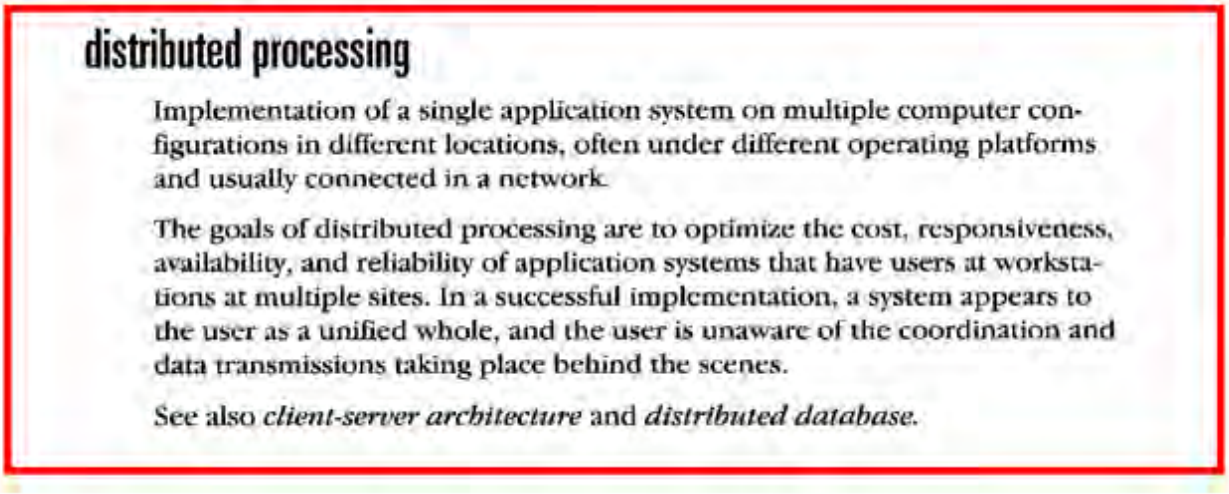
Doyle et al., "Processing Cross-sectional Image Data for Reconstruction of Human Developmental Anatomy from Museum Specimens," Newsletter of the Association for Computing Machinery Special Interest Group on Biomedical Computing, vol. 13, No. 1, ACM Press, coverage page, table of contents, pp. 9-15 (Feb. 1993), at 10 (Ex. M at PH_001_0000041533)): "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***."

2. Defendants' extrinsic evidence

21st Centry Dictionary of Computer Terms 112 (1994) ("distributed processing") [Ex. X at PA-0000333435]:

distributed processing An approach to information management in which data are stored and processed on more than one computer.

Que's Computer Programmer's Dictionary 137 (1993) ("distributed processing") [Ex. V at PA-0000333390]:



distributed processing

Implementation of a single application system on multiple computer configurations in different locations, often under different operating platforms and usually connected in a network.

The goals of distributed processing are to optimize the cost, responsiveness, availability, and reliability of application systems that have users at workstations at multiple sites. In a successful implementation, a system appears to the user as a unified whole, and the user is unaware of the coordination and data transmissions taking place behind the scenes.

See also *client-server architecture* and *distributed database*.

J. "workstation"

Claim Term(s)	Defendants' Proposed Construction	Eolas's Proposed Construction
workstation	a desktop or deskside computer with an operating system and hardware that provides higher performance than a personal computer	a computer system connected to a network that serves the role of an information requester

1. Defendants' intrinsic evidence

a. Claims

In the following chart, the term(s) for construction appear in all the claims marked with an "x." The letter "m" indicates a method claim, and "¶6" indicates a claim that the Defendants contend includes limitations subject to § 112, ¶ 6.

	'906 patent						'985 patent								
	m 1	m 4	m 5	¶6 6	¶6 9	¶6 10	m 1	¶6 16	m ¶6 20	m ¶6 24	¶6 28	m ¶6 32	m 36	m ¶6 40	m 44
workstation	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

b. Specification (all cites to '906 patent)

1:31–:38 (Background of the Invention): Computer systems connected to a network such as the Internet may be of varying types, e.g., mainframes, ***workstations***, personal computers, etc. The computers are manufactured by different companies using proprietary hardware and operating systems and thus have incompatibilities in their instruction sets, busses, software, file formats and other aspects of their architecture and operating systems.

3:63–4:6 & fig.2 (Background of the Invention): In FIG. 2, a user 102 operates a small computer 104, such as a personal computer or a ***work station***. The user's computer is equipped with various components, such as user input devices (mouse, trackball, keyboard, etc.), a display device (monitor, liquid crystal display (LCD), etc.), local storage (hard disk drive, etc.), and other components. Typically, small computer 104 is connected to a larger computer, such as server A at 106. The larger computer may have additional users and computer systems connected to it, such as computer 108 operated by user 110.

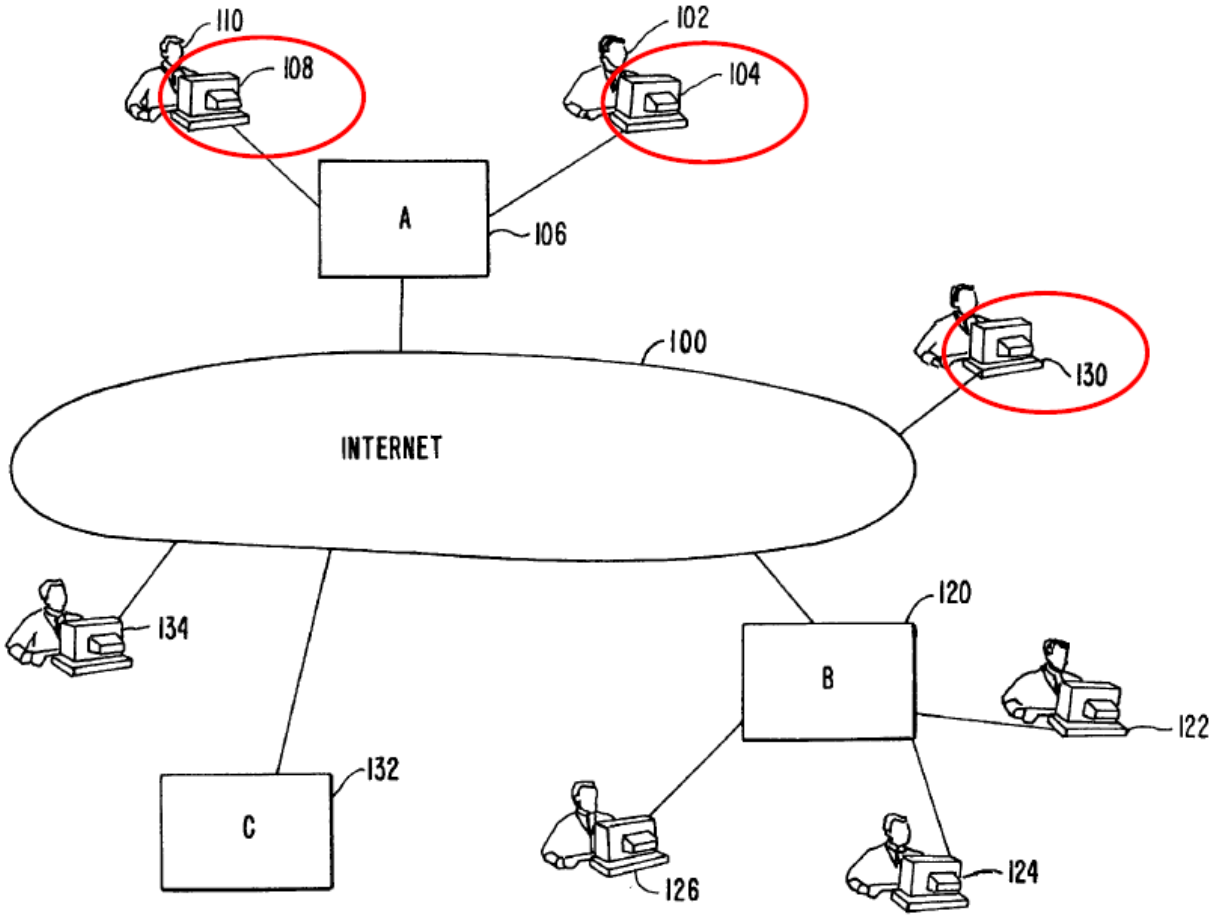


FIG. 2. PRIOR ART

4:24–:31 & fig.2 (Background of the Invention): A user at a ***workstation*** or personal computer need not connect to the Internet via a larger computer, such as server A or server B. This is shown, for example, by small computer 130 connected directly to Internet 100 as by a telephone modem or other link. Also, a server need not have users connected to it locally, as is shown by server C at 132 of FIG. 2. Many configurations of large and small computers are possible.

5:39–:56 (Background of the Invention): The open distributed hypermedia system provided by the Internet allows users to easily access and retrieve different data objects located in remote geographic locations on the Internet. However, this open distributed hypermedia system as it currently exists has shortcomings in that today's large data objects are limited largely by bandwidth constraints in the various communication links in the Internet and localized networks, and by the limited processing power, or computing constraints, of small computer systems normally provided to most users. Large data objects are difficult to update at frame rates fast enough (e.g., 30 frames per second) to achieve smooth animation. Moreover, the processing power needed to perform the calculations to animate such images in real time does not exist on most ***workstations***, not to mention personal computers. Today's browsers and viewers are not capable of performing the computation necessary to generate and render new views of these large data objects in real time.

6:17–:21 (Background of the Invention): [S]mall client computers in the form of personal computers or ***workstations*** such as client computer 108 of FIG. 2 are generally available to a much larger number of researchers. Further, it is common for these smaller computers to be connected to the Internet.

c. Prosecution history

i. '906 prosecution history (08/324,443)

Proposed claims to a "workstation"	**Proposed claims to a "computer"**
Original Application, at 29 (Oct. 17, 1994) (906 PH Ex. 1 at PH_001_0000783829): "1. A method for running an application program in a computer network environment, comprising: providing at least one client ***workstation*** . . . ." Original Application, at 31 (Oct. 17, 1994) (906 PH Ex. 1 at PH_001_0000783831): "15. A method for running an application program in a computer network environment, comprising: providing at least one client ***workstation*** . . . ."	Original Application, at 32 (Oct. 17, 1994) (906 PH Ex. 1 at PH_001_0000783832): "24. A method for interactively controlling an embedded object in a document displayed on a client ***computer*** . . . ." Original Application, at 34 (Oct. 17, 1994) (906 PH Ex. 1 at PH_001_0000783834): "34. A method for displaying a three dimensional image object on a client ***computer*** . . . ."
Amendment A, at 7 (Aug. 6, 1996) (906 PH Ex. 3 at PH_001_0000783885): "44. (New) A computer program product for use in a system having at least one client ***workstation*** . . . ." Amendment A, at 9 (Aug. 6, 1996) (906 PH Ex. 3 at PH_001_0000783887): "54. (New) A computer program product for use in a system having at least one client ***workstation*** . . . ."	Amendment A, at 10 (Aug. 6, 1996) (906 PH Ex. 3 at PH_001_0000783888 – 90): "55. (New) A computer program product for use in a system including a client ***computer*** . . . ." Amendment A, at 12 (Aug. 6, 1996) (906 PH Ex. 3 at PH_001_0000783890 – 91): "56. (New) A computer program product for use in a system including a client ***computer*** . . . ."
Amendment B, at 1 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784029): "Please cancel claims 6-15, 17-43, and 49-56."	
Amendment B, at 1–2 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784029 – 30): "1. (Twice Amended) A method for running an application program in a computer network environment, comprising: providing at least one client ***workstation*** . . . said executable application to execute on said client ***workstation*** in order to display said object and enable interactive processing of said object . . . ." Amendment B, at 3–4 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784031 – 32): "44. (Amended) A computer program product for use in a system having at least one client ***workstation*** . . . said executable application to execute on said client ***workstation*** in order to display said object and enable interactive processing of said object . . . ."	
Notice of Allowability, at 1 (Mar. 30, 1998) (906 PH Ex. 17 at PH_001_0000784167): "The allowed claims are 1-5, 44-48."	

Amendment A, at 18–19 (Aug. 6, 1996) (906 PH Ex. 03 at PH_001_0000783896-97):

> The Ness disclosure does not teach using an application on a first additional computer to manipulate an object external to the hypermedia document within document window. In Ness both the object and application are included within the document displayed. While this feature enhances interactivity with documents, the application is executed on the computer displaying the document.
>
> In contrast, in the combination of claim 24 the first computer could be much more powerful than the client computer to run an application performing, for example, 3-D visualization or CAD/CAM programs, and the results could be displayed in the document window of client computer in the form of a PC or network computer.
>
> * * * * *
>
> The subject matter of claim 34 differs from the cited references for reasons similar to those discussed above with reference to claim 24. Again, the claimed system allows for a much more powerful first additional computer to perform calculations to reorient an object with the results communicated to the client computer to redisplay the object in a new orientation.

ii. <u>Abandoned application (09/075,359)</u>

Preliminary Amendment, at 2 (May 8, 1998) (359 PH Ex. 02 at PH_001_0000787771): "44. (New) A method for running an application program in a ***computer system*** . . . ."

Preliminary Amendment, at 3 (May 8, 1998) (359 PH Ex. 02 at PH_001_0000787772): "48. (New) The method of claim 4, wherein said client program and said server program reside on the same ***computer system***."

Preliminary Amendment, at 2 (May 8, 1998) (359 PH Ex. 02 at PH_001_0000787775): "55. (New) A computer program product for use in a system having at least one client ***workstation*** . . . ."

Preliminary Amendment, at 7 (May 8, 1998) (359 PH Ex. 02 at PH_001_0000787776): "57. (New) The method of claim 13, wherein said client program and said server program reside on the same ***computer system***."

Office Action, at 8 (Sept. 6, 2000) (359 PH Ex. 03 at PH_001_0000787800):

> As per claims 48 and 57, the location of the client and server program would have been a matter of design choice. It would have been obvious for one of ordinary skill in the art to have the client and server program on the same computer if the computer is powerful enough to handle the server program because it would have reduced delays and communications over the network.

iii. Second reexam (90/007,858)

Declaration of Michael D. Doyle, ¶¶ 4, 6, 8 (Sept. 22, 2007) (accompanying Applicants' Response (Sept. 27, 2007)) (858 PH Ex. 07 at PH_001_0000787070,72):

> 4. The earliest demonstrations of reductions to practice of the invention described in claims 1 and 6 of the 5,838,906 patent (referred to hereafter as 906) to individuals outside of UCSF Center for Knowledge Management staff were given during November of 1993. These demonstrations included private presentations to influential experts in the field, as well as public demonstrations to large technical and scientific audiences. The software used for these demonstrations incorporated all of the elements of claims 1 and 6 of the 906 patent. A detailed chronology these software demonstrations follows.
>
> * * * * *
>
> 6. Dr. Lindberg was visiting UCSF on November 17, 1993, for a conference regarding the Red Sage Project, which I was directing at the time. We pulled him aside during the conference to come down to my Center in order to view and interact with a demonstration of the 3-dimensional visualization of Visible Embryo Project data via our 906-enhanced web browser. This demonstration involved using our 906-enhanced Web browser, from a workstation in one of the Library's conference rooms, to access the Center for Knowledge Management's (CKM) Web server in order to demonstrate the features of the 906 invention. This demonstration involved showing him the Vis embedded 3-D visualization application, as well as an embedded video player and an embedded molecular modeling application, all embedded in Web pages served up by the Center's Web server over a TCP/IP Internet connection.
>
> * * * * *
>
> 8. During that same trip, I also visited with Dr. Adrianne Noe, currently Director of the National Museum of Health and Medicine, at the Armed Forces Institute of Technology (AFIP), to show her a similar demonstration of the 906 technology. Using a Silicon Graphics workstation in her Human Developmental Anatomy Center at AFIP, I installed and demonstrated our 906-enhanced Web browser, accessing the UCSF CKM Web server to show her the embedded visualization examples, as discussed above.

d. Cited prior art

Douglas Young, The X Window System, Programming and Applications with Xt, Prentice Hall, title page, copyright page, pp. i-x, 1-13, 123-166, 280-332, 520-533 (1990), at 1 (Ex. K):

AN INTRODUCTION TO THE X WINDOW SYSTEM

The X Window System is an industry-standard software system that allows programmers to develop portable graphical user interfaces. One of the most important features of X is its unique device-independent architecture. X allows programs to display windows containing text and graphics on any hardware that supports the X protocol without modifying, recompiling, or relinking the application. This device independence, along with X's position as an industry standard, allows X-based applications to function in a heterogeneous environment consisting of mainframes, workstations, and personal computers.

Doyle et al., "Processing Cross-sectional Image Data for Reconstruction of Human Developmental Anatomy from Museum Specimens," Newsletter of the Association for Computing Machinery Special Interest Group on Biomedical Computing, vol. 13, No. 1, ACM Press, coverage page, table of contents, pp. 9-15 (Feb. 1993), at 13 (Ex. M at PH_001_0000041533):

3-D Visualization Tools

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.

Ang et al., "Integrated Control of Distributed Volume Visualization Through the World-Wide-Web." Proceedings of Visualization 1994, IEEE Press, Washington, D.C., October 1994, § 4 (Ex. O at PH_001_0000759348)):

4. Results

The results of the above implementation are very encouraging. The Mosaic/VIS sucessfully allows users to visualize HDF volume datasets from various HTTP server sites. Fig 2 shows a snapshot of the WWW visualizer. Distributing the volume rendering loads results in a remarkable speedup in image computations. Our performance ananlysis with a homogeneous pool of Sun SPARCstation 2's on a relatively calm network produced reasonable results (Figures 4a, 4b, and 4c. Three trials per plot). The time-versus-number-of-workstations curve decreases as more servers participate, and plateaus when the number of SPARCstations is 11 in the case of 256x256 image (9 for 192x192 image, and 7 for 128x128 image). The speed increases at the plateaus are very significant: about 10 times for the 256x256 image, 8 times for the 192x192 image, and 5 times for the 128x128 image. The outcomes suggest that performance improvement is a function of the number of volume rendering servers. Furthermore, the optimal number of workstations and the speed increase are larger when the image size is bigger. This is in complete agreement with Giertsen's analysis. We have also successfully tested the software system in an environment consisting of heterogenous workstations: a SGI Indigo2 R4400/150MHz, two SGI Indy R4000PC/100MHz, a DEC Alpha 3000/500 with a 133MHz Alpha processor, two Sun SparcStations 10, and two Sun SparcStations 2, which were located arbitrarily on an Ethernet network. To our knowledge this is the first demonstration of the embedding of interactive control of a client/server visualization application within a multimedia document in a distributed hypermedia environment, such as the World Wide Web.

2. Defendants' extrinsic evidence

a. Dictionaries

21st Centry Dictionary of Computer Terms 380–81 (1994) [Ex. X at PA-000033341]:

workstation Broadly, applies to any computer available for use by only one individual at a time, and as such can refer to a personal computer; however, generally assumed to refer to high-power, full-featured desktop computers used for scientific and engineering applications. These are often based on the UNIX operating system with high-resolution screens, fast processing power, and large storage capacities.

Microsoft Press Computer Dictionary 369 (1991) [Ex. R at PA-00333496]:

> **workstation** In general, a combination of input, output, and computing hardware that can be used for work by an individual. More often, however, the term refers to a powerful stand-alone computer of the sort used in computer-aided design and other applications requiring a high-end, usually expensive, machine ($10,000 to $100,000) with considerable calculating or graphics capability. Sometimes, *workstation* is also used to refer to a microcomputer or terminal connected to a network.

b. <u>SGI's 10-K report on Sept. 28, 1994</u>

(Ex. Z at PA-0000333295):

BUSINESS

GENERAL

Silicon Graphics, Inc. (the "Company") designs and supplies a family of ***workstation***, server and supercomputer systems, incorporating interactive three-dimensional ("3D") graphics, digital media and multiprocessing supercomputing technologies. The ***workstation*** products are available in ***desktop*** and ***deskside*** configurations, and are used primarily by ***technical, scientific and creative professionals to simulate, analyze, develop and display complex 3D objects and phenomena***. The Company has, over the last ten years, been a pioneer in the 3D graphics field, and continues to be a leader in ***workstation*** graphics technology. The Company's marketing and development efforts have, in the past, focused largely on the ***technical computing*** community, including engineers, scientists, designers, simulation specialists, animators and others who deal with complex visualization problems.

. . . .

PRINCIPAL PRODUCTS

The Company's graphics computer systems range from the Indigo-R- family of ***desktop workstations***, including the Indy-TM- and Indigo(2)-TM-, to the Onyx-TM- and POWER Onyx-TM- systems, a family of advanced graphics supercomputers. In addition, the Company's Challenge-TM- and POWER Challenge-TM- family ranges from entry-level single processor servers to enterprise-wide symmetric multiprocessing supercomputers. The Company's products all use the MIPS RISC microprocessors developed by MTI, and generally are binary-compatible, meaning that software applications

run without modification across the entire product line. The Company's ***workstations*** include display, graphics and computational capabilities. Server models are general purpose computers with the same computational performance of their workstation counterparts, but without the graphics capabilities. Depending upon their application, servers may also have higher levels of data storage and/or communications capabilities than comparable ***workstations***. The high-end multiprocessor supercomputer systems are meant to replace or augment aging ***mainframe*** computers in compute intensive engineering, animation and scientific environments.

. . . .

THE INDY FAMILY The Indy ***desktop workstation***, originally introduced in July 1993, features advanced 3D graphics and imaging and the Indy Cam-TM-, its own digital color video camera. The Indy was developed as a ***low-price***, high-performance ***workstation*** with real-time video capability, interactive and professional quality graphics, audio and imaging capabilities. The Indy has significant appeal in markets such as mechanical CAD, chemistry, color publishing, film and video, software development, education and media authoring. The Indy systems are available with either the R4600-TM- or 150mhz R4400-TM- microprocessor and ***range in price from approximately $6,000 to $28,500.***[*]

THE IRIS INDIGO FAMILY The IRIS Indigo-R- ***workstation***, originally introduced in July 1991, was the first RISC PC with integrated digital media, combining the power of ***workstations*** with the ease-of-use, standards and ***affordability of personal computers***. The IRIS Indigo ***workstations*** are expandable and upgradable and were enhanced in January 1992 by the addition of three models, including the high-end Indigo Elan-TM-, which provide higher levels of graphics performance. Among the primary markets addressed by the IRIS Indigo are the mechanical CAD and computer-aided engineering, electronic design automation, computer-aided software engineering (CASE), geo-science, life science, management support and publishing markets. The IRIS Indigo systems incorporate either an R4000-R- or R4400 microprocessor at ***prices ranging from approximately $14,500 to $40,000***.

Silicon Graphics, Inc., Annual Report, Securities and Exchange Commission, Fiscal Year ended June 30, 1994 ("Form 10-K") (available at <http://www.sec.gov/Archives/edgar/data/802301/0000912057-94-003243.txt>, last visited Sept. 16, 2010) (Ex. Z.)

[*] These and all other prices quoted are September 1994 list prices, which are subject to discount based on volume and other factors.

K. "network server"

Claim Term(s)	Defendants' Proposed Construction	Eolas's Proposed Construction
network server	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

1. Defendants' intrinsic evidence

a. Claims

In the following chart, the term(s) for construction appear in all the claims marked with an "x." The letter "m" indicates a method claim, and "¶6" indicates a claim that the Defendants contend includes limitations subject to § 112, ¶ 6.

	'906 patent						'985 patent								
	m 1	m 4	m 5	¶6 6	¶6 9	¶6 10	m 1	¶6 16	m ¶6 20	m ¶6 24	¶6 28	m ¶6 32	m 36	m ¶6 40	m 44
network server	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

b. Specification (all cites to '906 patent)

Abstract: The invention allows a program to execute on a remote ***server*** or other computers to calculate the viewing transformations and send frame data to the client computer thus providing the user of the client computer with interactive features and allowing the user to have access to greater computing power than may be available at the user's client computer.

1:31–:38 (Background of the Invention): ***Computer systems*** connected to a network such as the Internet may be of varying types, e.g., mainframes, workstations, personal computers, etc. The computers are manufactured by different companies using proprietary hardware and operating systems and thus have incompatibilities in their instruction sets, busses, ***software***, file formats and other aspects of their architecture and operating systems.

4:16–:23 & fig. 2 (Background of the Invention): Internet 100 is made up of many interconnected computer systems and communication links. Communication links may be by hardwire, fiber optic cable, satellite or other radio wave propagation, etc. Data may move from ***server*** A to ***server*** B through any number of ***intermediate servers*** and communication links or other computers and data processing equipment not shown in FIG. 2 but symbolically represented by Internet 100.

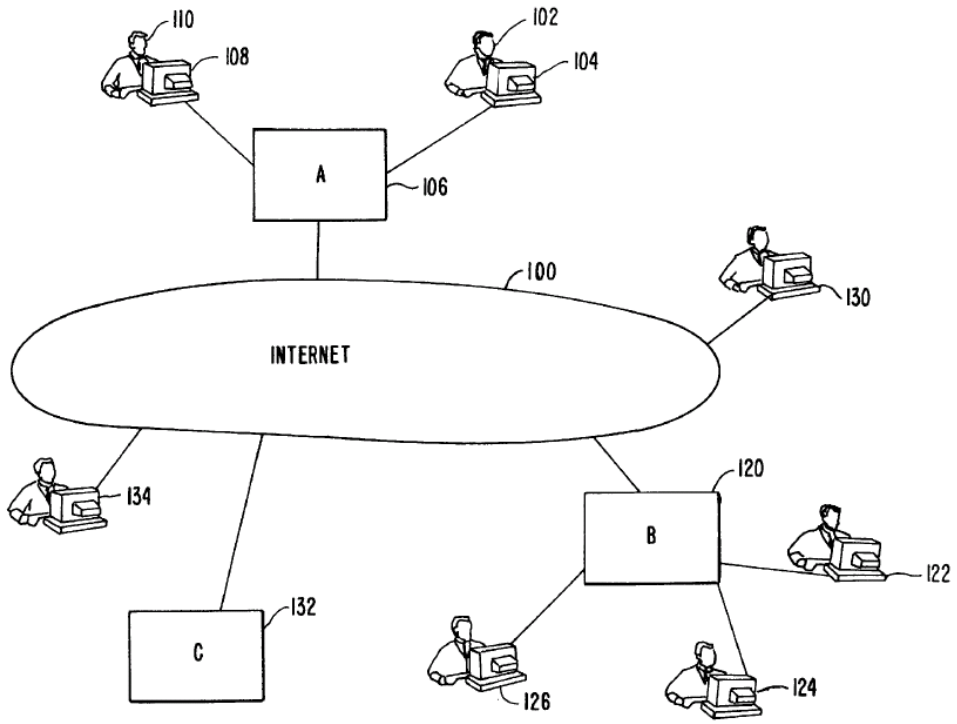


FIG. 2. PRIOR ART

4:32–:50 & fig. 2 (Background of the Invention): Typically, a computer on the Internet is characterized as either a "client" or "server" depending on the role that the computer is playing with respect to requesting information or providing information. Client computers are computers that typically request information from a server computer which provides the information. ***For this reason, servers are usually larger and faster machines that have access to many data files, programs, etc., in a large storage associated with the server.*** However, the role of a server may also be adopted by a smaller machine depending on the transaction. That is, user 110 may request information via their computer 108 from server A. At a later time, server A may make a request for information from computer 108. In the first case, where computer 108 issues a request for information from server A, computer 108 is a "client" making a request of information from server A. ***Server A may have the information in a storage device that is local to Server A or server A may have to make requests of other computer systems to obtain the information***.

4:66–5:21 & figs. 1-2 (Background of the Invention): For example, hypertext document 10 of FIG. 1 may be located at user 110's client computer 108. When user 110 makes a request by, for example, clicking on hypertext 20 (i.e., the phrase "hypermedia"), user 110's small client computer 108 processes links within hypertext document 10 to retrieve document 14. ***In this example, we assume that document 14 is stored at a remote location on server B.*** Thus, in this example, computer 108 ***issues a command*** that includes the address of document 14. This command is routed through server A and Internet 100 and eventually is received by server B. ***Server B processes the command and locates document 14 on its local storage.*** Server 14 [sic] then ***transfers a copy*** of the document back to client 108 via Internet 100 and server A. After client computer 108 receives document 14, it is displayed so that user 110 may view it.

Similarly, image object 16 and sound data file 40 may reside at any of the computers shown in FIG. 2. Assuming image object 16 resides on server C when user 110 clicks on image icon 22, client computer 108 ***generates a command*** to retrieve image object 16 to server C. ***Server C receives the command*** and ***transfers a copy*** of image object 16 to client computer 108.

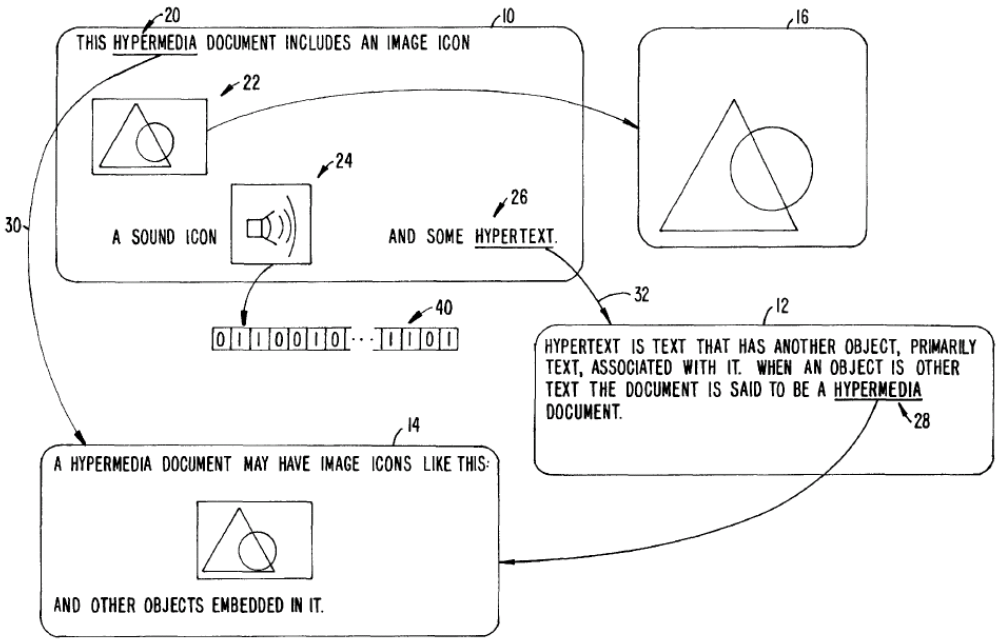


FIG. 1. PRIOR ART

9:45–:63 & fig. 5 (Detailed Description of a Preferred Embodiment): An example of the type of processing that application client 210 may perform is multidimensional image visualization. Note that application client 210 is in communication with network 206 via the network protocol layer of client computer 200. This means that application client 210 can make requests over network 206 for data objects, such as multidimensional image objects. For example, application client 210 may request an object, such as object 1 at 216, located in ***server computer*** 204. Application client 210 may make the request by any suitable means. Assuming network 206 is the Internet, such a request would typically be made by using HTTP in response to a HTML-style link definition for embedded program link 214.

Assuming application client 210 has made a request for the data object at 216, ***server process*** 218 ultimately receives the request. ***Server process*** 218 then ***retrieves data object*** 216 and ***transfers*** it over network 206 back to application client 210.

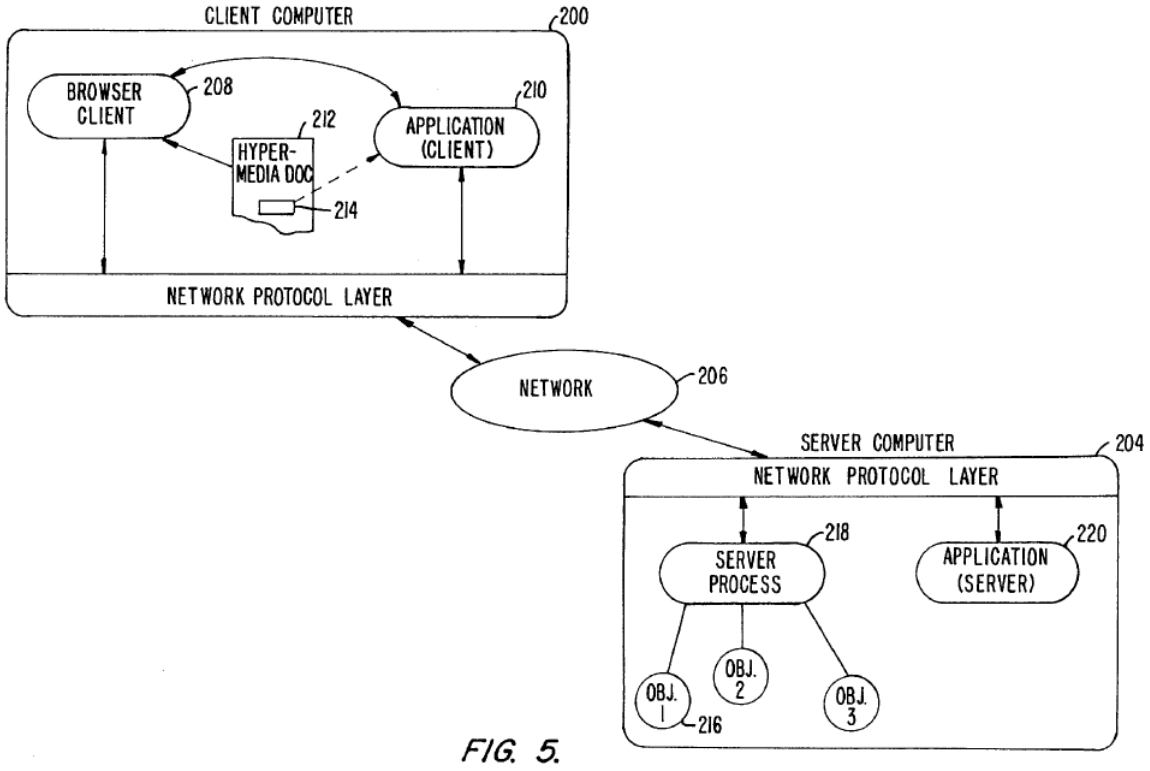


FIG. 5.

10:33–:46 & fig. 5 (Detailed Description of a Preferred Embodiment): Another embodiment of the present invention uses an application ***server process*** executing on ***server computer*** 204 to assist in processing that may need to be performed by an external program. For example, in FIG. 5, application server 220 resides on ***server computer*** 204. ***Application server*** 220 works in communication with application client 210 residing on client computer 200. In a preferred embodiment, application server 220 is called VRServer, also a part of Doyle Group's approach. Since ***server computer*** 204 is typically a larger computer having more data processing capabilities and larger storage capacity, ***application server*** 220 can operate more efficiently, and much faster, than application client 210 in executing complicated and numerous instructions.

11:28–:32 & fig. 6 (Detailed Description of a Preferred Embodiment): Thus, several computers, such as ***server computer*** 204 and additional computers 222 and 224 can all work together to perform the task of computing a new viewpoint and frame buffer for the embryo for the new orientation of the embryo image in the present example.

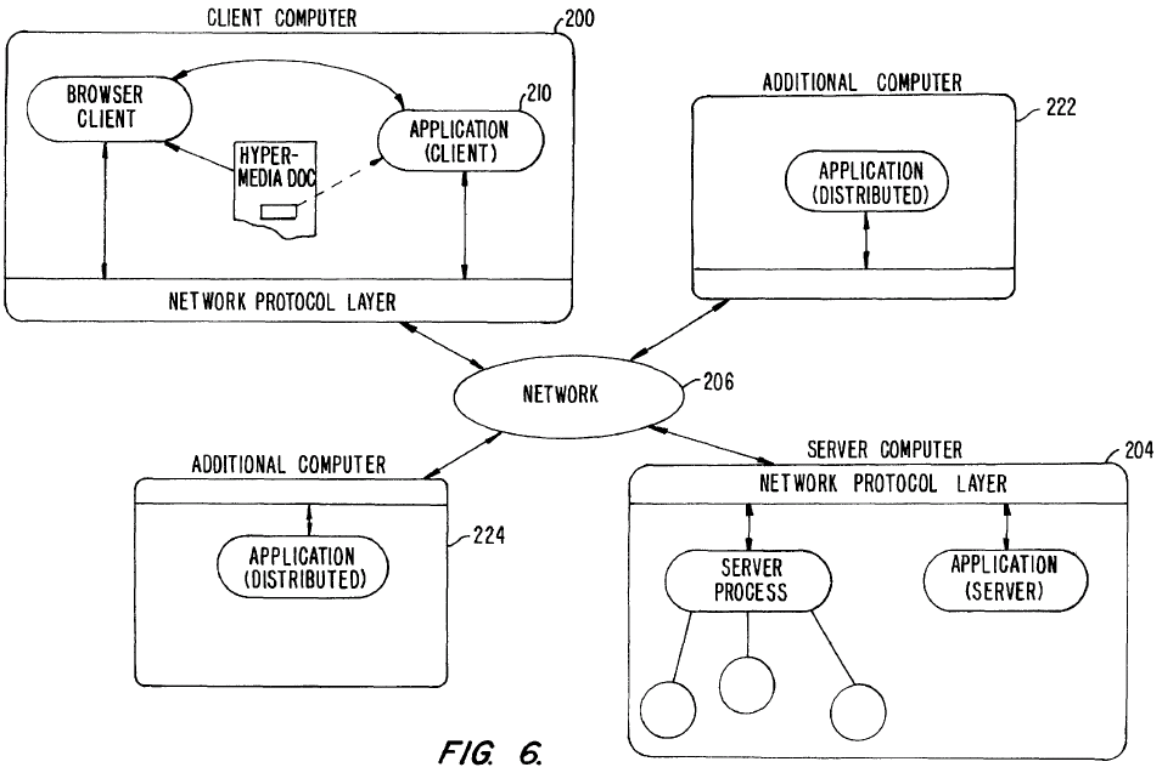


FIG. 6.

12:9–:13 & table I (Detailed Description of a Preferred Embodiment): ***The various processes in the system of the present invention*** communicate through a custom API called Mosaic/External Application Program Interface MEAPI. The MEAPI set of predefined messages includes those shown in Table I.

TABLE I

Message Function	Message Name
Messages from server to client:	
1. Server Update Done	XtNrefreshNotify
2. Server Ready	XtNpanelStartNotify
3. Server Exiting	XtNpanelExitNotify
Messages from client to server:	
4. Area Shown	XtNmapNotify
5. Area Hidden	XtNunmapNotify
6. Area Destroyed	XtNexitNotify

12:30–:37 & table I (Detailed Description of a Preferred Embodiment): Thus, by using MEAPI a ***server process*** communicates to a client application program to let the client application know when the ***server*** has finished updating information, such as an image frame buffer, or pixmap (Message 1); when the ***server*** is ready to start processing messages (Message 2) and when the ***server*** is exiting or stopping computation related to the ***server application program***.

12:38–:45 & table I (Detailed Description of a Preferred Embodiment): For client to ***server*** communications, MEAPI provides for the client informing the ***server*** when the image display window area is visible, when the area is hidden and when the area is destroyed. Such information

allows the ***server*** to decide whether to allocate computing resources for, e.g., rendering and viewing transformation tasks where the ***server*** is running an application program to generate new views of a multi dimensional object.

15:58–:67 (Detailed Description of a Preferred Embodiment): The ***present invention*** allows a user to have interactive control over application objects such as three dimensional image objects and video objects. In a preferred embodiment, controls are provided on the external applications' user interface. In the case of a VIS/panel application, a process, "panel" creates a graphical user interface (GUI) thru which the user interacts with the data. The application program, VIS, can be executing locally with the user's computer or remotely on a ***server***, or on one or more different computers, on the network.

c. Prosecution history

i. '906 prosecution history (08/324,443)

Amendment B, at 25-26 (June 2, 1997) (906 PH Ex. 11 at PH_001_0000784053-54): None of the cited references show this feature. This feature leads to the additional surprising an[d] unexpected results over the prior art of allowing the user to employ the hypermedia document as an interface to control and/or edit ***data objects which reside on the network server***, remotely, from the client workstation. One of many possible uses of this feature is to allow the user to make modifications to the ***original data object***, which may remain in place on the ***network server***, and which is referenced in the hypermedia document, so that others viewing the hypermedia document in the future from other client workstations will see those modifications.

ii. First reexam (90/006,831)

Response, at 3-4 & 13-14 (May 11, 2004) (831 PH Ex. 07 at PH_001_0000785361-62,71-72):

> The specification of the '906 patent (Applicants' Admitted Prior Art) describes a browser application, e.g., Mosaic, that functions as a viewer to view HTML documents. There are several ways to retrieve an HTML document from a network server, all of which require user interaction with the browser. [Felten, paragraph 8] When the browser is launched on a client workstation, a home page may be retrieved, a URL saved in a favorites list may be selected, or a link in a displayed page may be selected. 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 stores a temporary local copy of the HTML source document in a cache on the client workstation.
>
> The browser application then parses the local copy of the HTML document, renders the temporary local copy of the HTML document into a Web page , and displays the rendered Web page in a browser-controlled window. [Felten, paragraph 21] During the rendering step, the browser may retrieve information external to the local copy of the HTML document, such as source files referenced by IMG tags, render the images from the retrieved files

as static graphic images, and insert the images into the Web page of the HTML document, for display to the user.

There is no further interaction with the source HTML document or the local copy of the source HTML document subsequent to its being rendered and displayed. If a user believes the source HTML document has changed (s)he can click a refresh button in the browser GUI which causes the browser application to retrieve the source HTML document from the network server again, store a local copy again, parse and render again the newly retrieved local copy of the source HTML document, and replace the display of the previous version of the retrieved source HTML document with the subsequently retrieved version in the browser-controlled window or another window. For example, if the source HTML document were a price list of goods the user might refresh the document to determine if the prices had changed.

* * * *

As described above, in the write-once-publish-many paradigm of Mosaic and Berners-Lee only a temporary copy of the source HTML document is stored on the client machine. If an end-user utilized an editor executed on the client-workstation only a locally-cached data file could be edited. [Felten, paragraph 48] In the client-server model of Mosaic and Berners-Lee, the end user can't upload changes back to the server, only the Web page author can do that. If the end-user were to refresh the page the changes made to the locally-cached data would be overwritten and the display in the browser-controlled window would not change. [Felten, paragraph 23,24]

Declaration of Edward W. Felton, ¶¶ 20-23 (May 7, 2004) (accompanying Applicants' Response (May 11, 2004)) (831 PH Ex. 10 at PH_001_0000785440):

20. Mosaic, and other prior art browsers, executed on a client computer, and operated by downloading copies of web pages (and other files, such as embedded static images) over a network from web servers. After downloading a copy of a file, Mosaic would sometimes keep a copy of that file in a local cache, on the user's client computer. Caching allowed the file to be referenced more quickly if it was needed again later.

21. After downloading a file, Mosaic would parse that file (i.e., analyze its structure) to determine how the file should be displayed on the screen. Mosaic would then paint the contents of the file into a browser window.

22. When Mosaic, or another prior art browser, was used to view web pages, several steps stood between the author of the web page and the user who was viewing it. First, the file would be copied, at least once and perhaps more times, while in transit between the web server and the user's browser. Second, the file would be written in one format (typically, HTML) but displayed in another form, by rendering the HTML into a visual representation that would actually be presented to the user.

23. Because these steps stood between the author and the user, there was no realistic way for the user to edit the web page on the client workstation. The user did not have access to the version of the page that was distributed – that version lived on the server, and it wouldn't make sense to let an arbitrary user edit the contents of somebody else's web page.

iii. Interference 105,563 McK

BPAI decision, at 19 (May 24, 2007) (563 PH Ex. 01 at PH_001_0000787502):

document.[12] While the term "network" does not appear in the '701 patent, there is no dispute regarding the meanings of the terms "network" and "network server," which are defined as follows in *Microsoft Press Computer Dictionary* (3d ed. 1997) (hereinafter *1997 Microsoft Dictionary*) at 327, 329, 430:[13]

> **network** . . . *n.* A group of computers and associated devices that are connected by communications facilities. . . .
>
> **network server** . . . *n.* *See* server.
>
> **server** . . . *n.* 1. On a local area network (LAN), a computer running administrative software that controls access to the network and its resources, such as printers and disk drives, and provides resources to computers functioning as workstations on the network. 2. On the Internet or other network, a computer or program that responds to commands from a client. . . .

Doyle Annotated Copy of Claims, at 2 (July 3, 2007) (563 PH Ex. 02 at PH_001_0000787571):

providing at least one client workstation { **Fig. 5, item 200** } and one network server { **Fig. 5, item 204** } coupled to said network environment { **Fig. 5, item 206** }, wherein

* * * *

hypermedia document { **Fig. 5, item 212** } received over said network { **Fig. 5, item 206** } from said server { **Fig. 5, item 204** }, wherein the portion of said first

Doyle Annotated Copy of Claims, at 3 (July 3, 2007) (563 PH Ex. 02 at PH_001_0000787572):

controllable application { **Fig. 5, item 210** } reside on said network server { **Fig. 5, item 204** }, wherein said step of interactively controlling said controllable application { **Fig. 5, item 210** }

* * * *

issuing, from the client workstation { **Fig. 5, item 200** }, one or more commands to the network server { **Fig. 5, item 204** };

executing, on the network server { **Fig. 5, item 204** }, one or more instructions in response to said commands;

Doyle Annotated Copy of Claims, at 4 (July 3, 2007) (563 PH Ex. 02 at PH_001_0000787573):

sending information from said network server { **Fig. 5, item 204** } to said client

* * * *

workstation { **Fig. 5, item 200** } and one network server { **Fig. 5, item 204** } coupled to said

* * * *

document { **Fig. 5, item 212** } received over said network { **Fig. 5, item 206** } from said server { **Fig. 5, item 204** }, wherein the portion of said first

Doyle Annotated Copy of Claims, at 6 (July 3, 2007) (563 PH Ex. 02 at PH_001_0000787575):

9. The computer program product of claim 8, wherein additional instructions for controlling said controllable application { **Fig. 5, item 210** } reside on said network server { **Fig. 5, item 204** }, wherein said step of interactively controlling said controllable application { **Fig. 5, item 210** } includes:

computer readable program code for causing said client workstation { **Fig. 5, item 200** } to issue, from the client workstation { **Fig. 5, item 200** }, one or more commands to the network server { **Fig. 5, item 204** };

computer readable program code for causing said network server { **Fig. 5, item 204** } to

iv. Second reexam (90/007,858)

Applicants' Response, at 7 (Sept. 27, 2007) (858 PH Ex. 05 at PH_001_0000787036):

Mosaic functions as a viewer to view HTML documents. There are several ways to retrieve an HTML document from a network server, all of which require user interaction with the browser. [Felten at paragraphs 15, 29: NCSA Mosaic page 463, col. 1, 3rd paragraph]. The browser then retrieves a selected published source HTML document from a network server by utilizing a uniform resource locator (URL) that locates the HTML document on the network and stores a temporary local copy of the HTML source document in a cache on the client workstation.

The browser application then parses the local copy of the HTML document, renders the temporary local copy of the HTML document into a Web page , and displays the rendered Web page in a browser-controlled window. [Felten at paragraph 15; NCSA Mosaic page 463, Fig. 1 and caption]. During the rendering step, the browser may retrieve information external to the local copy of the HTML document, such as source files referenced by IMG tags, render the images from the retrieved files as static graphic images, and insert the images into the Web page of the HTML document, for display to the user.

v. '985 prosecution history (10/217,955)

Second Supplemental Amendment, at 19 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784631):

> *wherein the network environment comprises at least one* ***client workstation*** *and one* ***network server*** *coupled to the network environment,*
>
> **EXAMPLE SUPPORT:**
>
> **8:58** "In FIG. 5, client computer 200 communicates with server computer 204 via network 206. Both client computer 200 and server computer 204 use a network protocol layer to communicate with network 206. In a preferred embodiment, network 206 is the Internet and the network protocol layers are TCP/IP. Other networks and network protocols may be used."

Second Supplemental Amendment, at 29 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784641):

> *wherein the network environment comprises at least one* ***client workstation*** *and one* ***network server*** *coupled to the network environment,*
>
> **EXAMPLE SUPPORT:**
>
> 8:58 "In FIG. 5, client computer 200 communicates with server computer 204 via network 206. Both client computer 200 and server computer 204 use a network protocol layer to communicate with network 206. In a preferred embodiment, network 206 is the Internet and the network protocol layers are TCP/IP. Other networks and network protocols may be used."

Second Supplemental Amendment, at 36-37 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784648-49):

communicating via the network server with at least one client workstation *over said network*

EXAMPLE SUPPORT:

4:44 "In the first case, where computer 108 issues a request for information from server A, computer 108 is a "client" making a request of information from server A. Server A may have the information in a storage device that is local to Server A or server A may have to make requests of other computer systems to obtain the information. User 110 may also request information via their computer 108 from a server, such as server B located at a remote geographical location on the Internet."

5:6 "Thus, in this example, computer 108 issues a command that includes the address of document 14. This command is routed through server A and Internet 100 and eventually is received by server B. Server B processes the command and locates document 14 on its local storage. Server 14 then transfers a copy of the document back to client 108 via Internet 100 and server A. After client computer 108 receives document 14, it is displayed so that user 110 may view it. "

8:58 "In FIG. 5, client computer 200 communicates with server computer 204 via network 206. Both client computer 200 and server computer 204 use a network protocol layer to communicate with network 206."

9:15 "Browser client 208 is a process that a user of client computer 200 invokes in order to access various data objects, such as hypermedia documents, on network 206. Hypermedia document 212 shown within client computer 200 is an example of a hypermedia document, or object, that a user has requested access to."

Second Supplemental Amendment, at 37 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784649):

in order to cause said client workstation to:

receive, over said network environment from said server,

EXAMPLE SUPPORT:

5:10 "Server 14 then transfers a copy of the document back to client 108 via Internet 100 and server A. After client computer 108 receives document 14, it is displayed so that user 110 may view it."

9:20 "In this example, hypermedia document 212 has been retrieved from a server connected to network 206 and has been loaded into, e.g., client computer 200's RAM or other storage device. "

Second Supplemental Amendment, at 43 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784655):

> *wherein said network environment has at least* ***one client workstation and one network server*** *coupled to a network environment, the method comprising:*
>
> **EXAMPLE SUPPORT:**
>
> **8:58** "In FIG. 5, client computer 200 communicates with server computer 204 via network 206. Both client computer 200 and server computer 204 use a network protocol layer to communicate with network 206. In a preferred embodiment, network 206 is the Internet and the network protocol layers are TCP/IP. Other networks and network protocols may be used."

Second Supplemental Amendment, at 49 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784661):

> **CLAIM 31**. *One or more computer readable media encoded with software comprising an executable application for use in a system having at least* ***one client workstation and one network server***
>
> **EXAMPLE SUPPORT:**
>
> **8:58** "In FIG. 5, client computer 200 communicates with server computer 204 via network 206. Both client computer 200 and server computer 204 use a network protocol layer to communicate with network 206. In a preferred embodiment, network 206 is the Internet and the network protocol layers are TCP/IP. Other networks and network protocols may be used."

Second Supplemental Amendment, at 55-57 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784667-68):

CLAIM 35. *A method for serving digital information in a computer network environment, said method comprising:*

communicating via a network server with at least one client workstation over said computer network environment *in order to cause said client workstation to:*

EXAMPLE SUPPORT:

4:44 "In the first case, where computer 108 issues a request for information from server A, computer 108 is a "client" making a request of information from server A. Server A may have the information in a storage device that is local to Server A or server A may have to make requests of other computer systems to obtain the information. User 110 may also request information via

their computer 108 from a server, such as server B located at a remote geographical location on the Internet."

5:6 "Thus, in this example, computer 108 issues a command that includes the address of document 14. This command is routed through server A and Internet 100 and eventually is received by server B. Server B processes the command and locates document 14 on its local storage. Server 14 then transfers a copy of the document back to client 108 via Internet 100 and server A. After client computer 108 receives document 14, it is displayed so that user 110 may view it. "

8:58 "In FIG. 5, client computer 200 communicates with server computer 204 via network 206. Both client computer 200 and server computer 204 use a network protocol layer to communicate with network 206."

9:15 "Browser client 208 is a process that a user of client computer 200 invokes in order to access various data objects, such as hypermedia documents, on network 206. Hypermedia document 212 shown within client computer 200 is an example of a hypermedia document, or object, that a user has requested access to."

Second Supplemental Amendment, at 59 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784671):

wherein said computer network environment has ***at least said client workstation and said network server*** *coupled to the computer network environment,*

EXAMPLE SUPPORT:

8:58 "In FIG. 5, client computer 200 communicates with server computer 204 via network 206. Both client computer 200 and server computer 204 use a network protocol layer to communicate with network 206. In a preferred embodiment, network 206 is the Internet and the network protocol layers are TCP/IP. Other networks and network protocols may be used."

Second Supplemental Amendment, at 63 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784675):

comprises at least one client workstation and one ***remote network server***

EXAMPLE SUPPORT:

7:7 "In one application, high resolution three dimensional images are processed in a distributed manner by several computers located remotely from the user's client computer."

8:58 "In FIG. 5, client computer 200 communicates with server computer 204 via network 206. Both client computer 200 and server computer 204 use a network protocol layer to communicate with network 206. In a preferred embodiment, network 206 is the Internet and the network protocol layers are TCP/IP. Other networks and network protocols may be used. For ease of illustration, additional hardware and software layers are not shown in FIG. 5. "

10:61 "application server 220 performs the mathematical calculations to compute a new view for the embryo image. Once the new view has been computed, the image data for the new view is sent over network 206 to application client 210 so that application client 210 can update the viewing window currently displaying the embryo image."

Second Supplemental Amendment, at 70 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784682):

> *having* ***a network server*** *coupled to said computer network environment,*
>
> **EXAMPLE SUPPORT:**
>
> **8:58** "In FIG. 5, client computer 200 communicates with server computer 204 via network 206. Both client computer 200 and server computer 204 use a network protocol layer to communicate with network 206. In a preferred embodiment, network 206 is the Internet and the network protocol layers are TCP/IP. Other networks and network protocols may be used."

Second Supplemental Amendment, at 77-78 (Feb. 5, 2009) (985 PH Ex. 14 at PH_001_0000784689-90):

> *with the network server coupled to said computer network environment,*
>
> *wherein said computer network environment has* ***at least said client workstation*** *and said* ***network server*** *coupled to the computer network environment,*
>
> **EXAMPLE SUPPORT:**
>
> **8:58** "In FIG. 5, client computer 200 communicates with server computer 204 via network 206. Both client computer 200 and server computer 204 use a network protocol layer to communicate with network 206. In a preferred embodiment, network 206 is the Internet and the network protocol layers are TCP/IP. Other networks and network protocols may be used."

d. Cited prior art

Ang et al., "Integrated Control of Distributed Volume Visualization Through the World-Wide-Web." Proceedings of Visualization 1994, IEEE Press, Washington, D.C., October 1994, § 3 (Ex. O at PH_001_0000759346): HTTP ***servers*** respond to requests from clients, e.g. Mosaic, and ***transfer*** hypertext documents. Those documents may contain text and images as intrinsic elements and may also contain external links to any arbitrary data object (e.g. audio, video, etc...). Mosaic may also communicate with other Internet servers, e.g. FTP, either directly - translating request results into HTML on demand - or via a gateway that provides translation services. As a W3 client, Mosaic communicates with the ***server(s)*** of interest in response to user actions (e.g. selecting a hyperlink), initiating a connection and requesting the document specified by the URL. The ***server*** delivers the file specified in the URL, which may be a HTML document or a variety of multimedia data files (for example, images, audio files, and MPEG movies) and Mosaic uses the predefined SGML DTD for HTML to parse and present the information.

2. Defendants' extrinsic evidence

Microsoft Press Computer Dictionary 75 (2d ed.1994) [Ex. Y at PA-0000333411]:

client/server architecture An arrangement used on local area networks that makes use of "distributed intelligence" to treat both the server and the individual workstations as intelligent, programmable devices, thus exploiting the full computing power of each. ***This is done by splitting the processing of an application between two distinct components: a "front-end" client and a "backend" server.*** The client component, itself a complete stand-alone personal computer (vs. the "dumb" terminal found in older architectures such as the time-sharing used on mainframe), offers the user its full range of power and features for running applications. The server component, which can be another personal computer, a minicomputer or a mainframe enhances the client component by providing the traditional strengths offered by minicomputers and mainframes in time-sharing environment: data management information sharing between clients and sophisticated network administration and security features. The advantage of the client/server architecture over older architectures is that the client and server machines work together to accomplish the processing of the application being used. Not only does this increase the processing power available, but it also uses that power more efficiently. The client portion of the application is typically optimized for user interaction, whereas the server portion provides the centralized multiuser functionality.

Microsoft Press Computer Dictionary 268 (2d ed.1994) [Ex. Y at PA-0000333422]:

network A group of computers and associated devices that are connected by communications facilities. A network can involve permanent connections, such as cables, or temporary connections made through telephone or other communications links. A network can be as small as a local area network consisting of a few computers, printers, and other devices, or it can consist of many small and large computers distributed over a vast geographic area. Small or large, a computer network exists to provide computer users with the means of communicating and transferring information electronically. Some types of communication are simple user-to-user messages; others, of the type known as distributed processes, can involve several computers and the sharing of workloads or cooperative efforts in performing a task.

Microsoft Press Computer Dictionary 269 (2d ed.1994) [Ex. Y at PA-0000333423]:

network server *See* server

Microsoft Press Computer Dictionary 355 (2d ed.1994) [Ex. Y at PA-0000333426]:

server On a local area network, a ***computer running administrative software*** that controls access to all or part of the network and its resources (such as disk drives or printers). A computer acting as a ***server*** makes resources available to computers acting as workstations on the network. *Compare* client; *see also* client/server architecture.