CLAIM CHART EXHIBIT 7
"MEDIAVIEW"

INVALIDITY CLAIM CHART FOR U.S. PATENT NO. 5,838,906

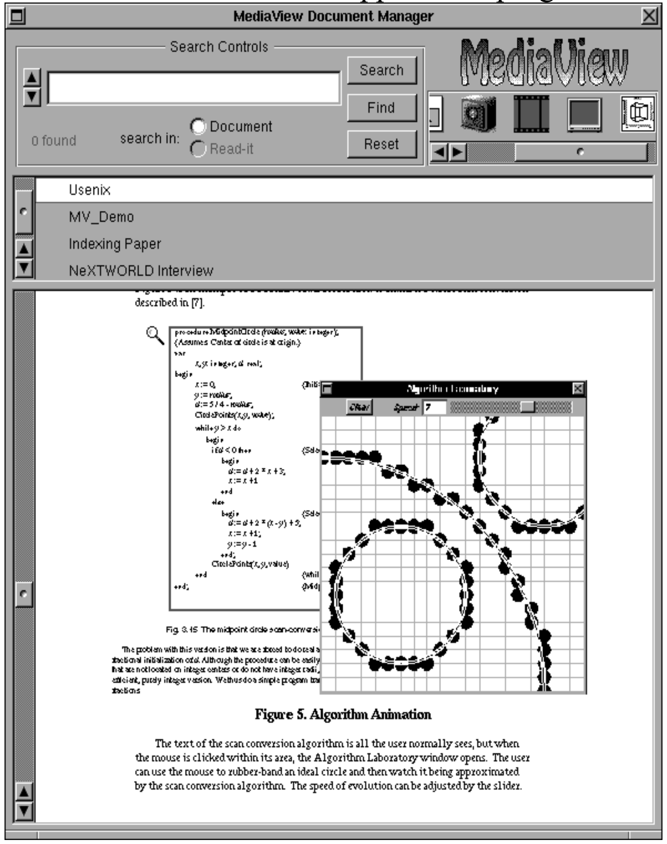
- **"MEDIAVIEW" [1]—AS INTENDED TO BE USED IN A COMPUTER SYSTEM AND DEMONSTRATION OF SAME, INCLUDING INDIVIDUALLY AND COLLECTIVELY:**
    - **An Interpersonal Multimedia Visualization System, IEEE Computer Graphics & Applications, May, 1991 [PA-00273175] [Phillips91a];**
    - **MediaView: An Editable Multimedia Publishing System Developed With An Object-Oriented Toolkit, Proc. USENIX Summer Conf., Nashville, TN, June, 1991 [PA-00327398] [Phillips91b];**
    - **MediaView: A General Multimedia Digital Publication System, Comm. ACM, Vol. 34, No. 7, July, 1991 [PA-00273194] [Phillips91c];**
    - **"Media View, Short and Long Versions" March 1993 [RLP 7];**
    - **Public demonstrations and uses of MediaView including at EDUCOM '89 [RLP 11], SIGGRAPH '90 [RLP 1], SIGGRAPH '92 [DS 1], the Smithsonian Institution in 1991 [RLP 1], NeXTWORLD [RLP 1], and HP's Palo Alto, California (June 1993) and Fort Collins, Colorado (1991) facilities [RLP 10 and RLP 12], and R&D Magazine [Studt94]; and**
    - **Personal experience and knowledge with MediaView and NeXT systems, including my NeXT cube. The body of my report has a narrative description that augments and should be considered part of this chart, and vise-versa for this and all my charts.**

("MEDIAVIEW")

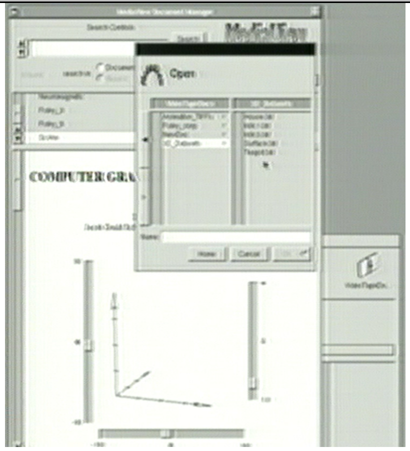
Claim Text from '906 Patent	MediaView

[1] The features and functions of MediaView are identified singularly for all versions of MediaView except where otherwise noted. MediaView 2.X was completed prior to October 1992 and was distributed through a Purdue server, which is no longer available. See Martin Exhibits 14, 15, and 16. A copy of MediaView 2 that was distributed through SIGGRAPH '92 was produced by Sadowski [DS 1], who attended the conference and a demonstration of MediaView. MediaView was widely shown throughout the years of its development including through 1994, see, e.g., [Studt94] (which shows a RenderMan object). The primary difference for purposes of my analysis between MediaView 2.X and 3.X was the inclusion of the RenderMan custom component, which, like the usage of Mathematica in prior versions of MediaView, employed a client-server configuration for the distribution, manipulation, and interaction with objects embedded in a MediaView document. The versions of MediaView 2.X that I have located to date are from June 11, 1992 (on my NeXT cube) and another from SIGGRAPH '92, though on media dated November 29, 1992. The MediaView 3 (the primary application) was completed around October 1992 (which is shortly after the release of NeXTSTEP 3.0), though the last date for RenderMan related code is November 9, 1992, so I will assume that is the date of MediaView 3. I note that I did make a new compilation of this version of MediaView for testing purposes so the tests I performed could be replicated due to a bug in saving certain files. I am not relying on the bug fix itself for my invalidity analysis—my analysis is of the November 9, 1992 version.

Claim Text from ’906 Patent	MediaView

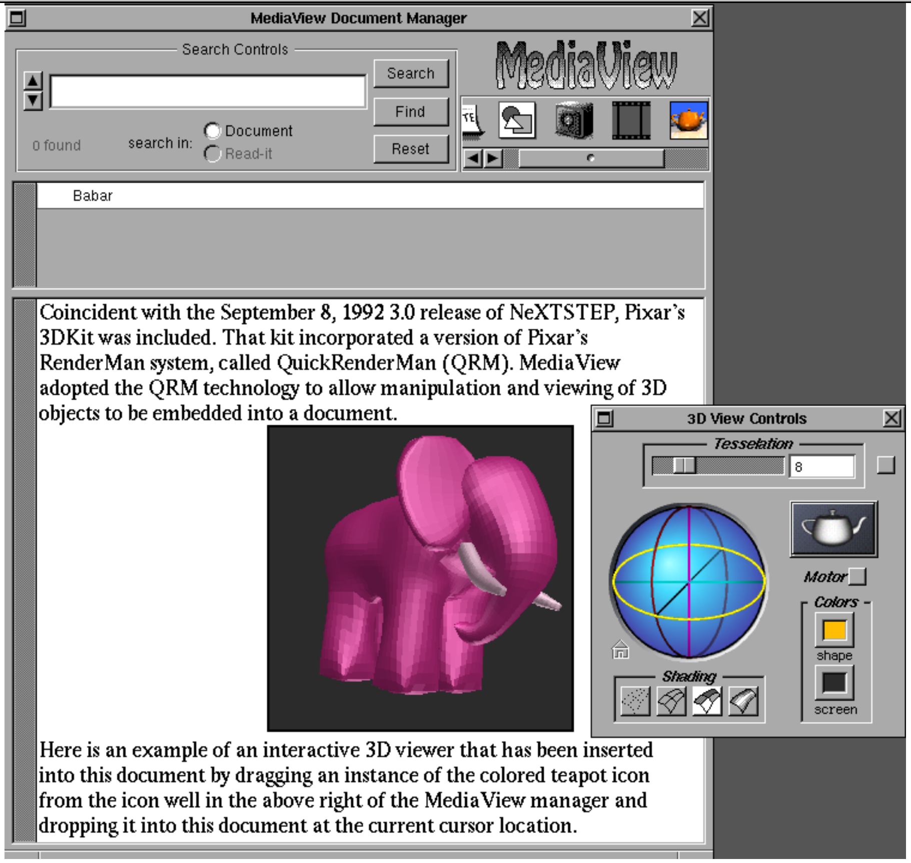
906-1.a:
A method for running an application program in a computer network environment, comprising:
	All instances of MediaView disclose an application program.
	 MediaView Document Manager
	Search Controls
	Search
	Find
	Reset
	0 found
	search in:
	Document
	Read-it
	MediaView
	Usenix
	MV_Demo
	Indexing Paper
	NeXTWORLD Interview
	Algorithm Laboratory
	Clear
	Speed
	7
	described in [7].
	procedure MidpointCircle (radius, value: integer);
	{Assumes Center of circle is at origin.}
	var
	x, y: integer; d: real;
	begin
	x := 0;
	y := radius;
	d := 5 / 4 - radius;
	CirclePoints(x,y, value);
	while y > x do
	begin
	if d < 0 then
	begin
	d := d + 2 * x + 3;
	x := x + 1
	end
	else
	begin
	d := d + 2 * (x - y) + 5;
	x := x + 1;
	y := y - 1
	end;
	CirclePoints(x, y, value)
	end
	end;
	Fig. 3.15 The midpoint circle scan-conversi
	The problem with this version is that we are forced to do real a
	fractional initialization of d. Although the procedure can be easily
	that are not located on integer centers or do not have integer radii,
	efficient, purely integer version. We thus do a simple program tra
	fractions
	Figure 5. Algorithm Animation
	The text of the scan conversion algorithm is all the user normally sees, but when the mouse is clicked within its area, the Algorithm Laboratory window opens. The user can use the mouse to rubber-band an ideal circle and then watch it being approximated by the scan conversion algorithm. The speed of evolution can be adjusted by the slider.
	
	The MediaView program described in this report is a computer program that was compiled from the MV.TAR produced file and whose appearance can be seen in the figures cited in this report above.
	
	MediaView discloses a computer network environment.

Claim Text from ’906 Patent	MediaView
	A workstation running MediaView, being part of a client/server network, operated in a distributed hypermedia environment. MediaView was “designed for maximum communicability” and “allows multimedia documents to be electronically mailed to remote sites. In short, MediaView is a communication tool that offers new and dramatically different ways of interacting with others.” [Phillips91c at 75]
906-1.b: providing at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment;	MediaView discloses a client workstation. MediaView, having been developed under NeXTSTEP, on a NeXT workstation, runs on a NeXT Computer client workstation. The NeXT workstation was designed for a networked environment, using Ethernet connectivity. Each NeXT computer came with a Network File System (NFS) manager, which enabled client/server connectivity. “MediaView fully exploits the platform integration and media richness of NeXT, NeXTstep, and NeXTdimension. Indeed, MediaView was first developed for this environment because it is a precursor for future systems.” [Phillips91c at 75] See also, e.g., [LANL93] (showing the file browser for finding a file, here a dataset that will be processed over the coordinate system):

Claim Text from ’906 Patent	MediaView
	 MediaView discloses a network server. As stated immediately above, MediaView ran in a client/server network environment. Thus, certain computers in the network could, via NFS, be designated as server machines. In this report, referring to Figure 6, et seq., there is a description of MediaView performing mathematical operations by passing coded requests to a Mathematica server. [MediaView 2.X and 3.X] The server could be anywhere on the network and be accessed by a Remote Procedural Call (RPC). Further along, referring to Figure 7, MediaView rendered 3D geometric datasets using a Pixar RenderMan server. [MediaView 3] That server could be anywhere on the network and, indeed, as Figure 8 depicts, there could be multiple servers all working on the same rendering task. MediaView discloses a distributed hypermedia environment. A workstation running MediaView, being part of a client/server network, operated in a distributed hypermedia environment.

Claim Text from ’906 Patent	MediaView
906-1.c: executing, at said client workstation, a browser application, that parses a first distributed hypermedia document to identify text formats included in said distributed hypermedia document and for responding to predetermined text formats to initiate processing specified by said text formats;	MediaView discloses a browser application. The MediaView program described in this is a computer program that was compiled from the MV.TAR produced-file and whose appearance can be seen in the figures cited above. Furthermore, Figure 5 of this report shows a document titled Usenix has been selected for browsing and a portion of it is seen in the browser window. MediaView discloses that the browser application parses a hypermedia document. In this report, under the heading “Parsing” the digital structure of a MediaView document is described. There it states, “MediaView…parses a stream of data in Rich Text Format (RTF) in order to discover the location of multimedia components embedded therein. It does this by looping through a series of “runs” of RTF data and searching for a “character” with an embedded flag that identifies it as a pointer to a multimedia component.” MediaView discloses a hypermedia document with text formats. In this report, under the heading “Parsing” the digital structure of a MediaView document is described. There it states, “MediaView…parses a stream of data in Rich Text Format (RTF) in order to discover the location of multimedia components embedded therein. It does this by looping through a series of “runs” of RTF data and searching for a “character” with an embedded flag that identifies it as a pointer to a multimedia component.”
906-1.d: utilizing said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said	MediaView discloses that a hypermedia document is received from the server. As stated above, using native NFS capabilities of the NeXT, MediaView could receive MediaView documents from a server.

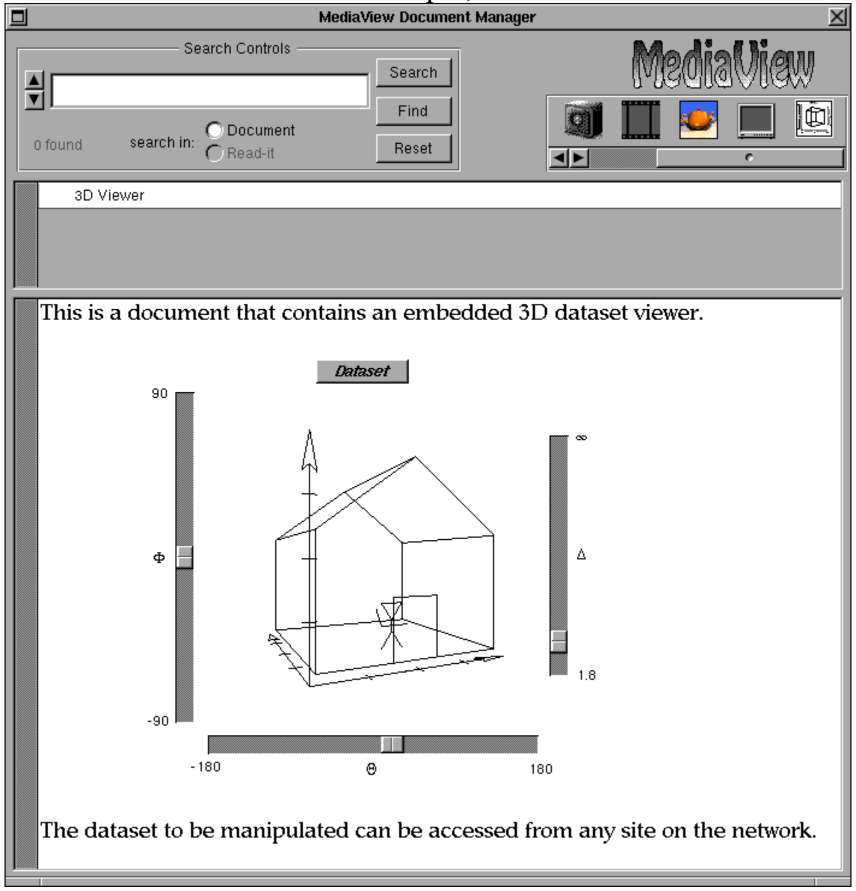
Claim Text from ’906 Patent	MediaView
server,	MediaView discloses that the browser displays a hypermedia document. The MediaView program displays hypermedia documents, documents that contain not only text but a wide variety of hypermedia components. From [Phillips91a], pp. 20-21, “Armed with modern workstation technology, we can provide an electronic reading environment. ‘Documents’ read in this environment can include not only text, line art, and still images, but also sound, video sequences, and computer-produced animations. And, when cast in digital form, the mathematical content of a document can be symbolically and numerically manipulated. Thus, we can experiment with the mathematics, derive new results, and simulate different situations with different parameters. We can explore computer generated images by moving the eye point, changing the lighting conditions, or using the underlying model with our own algorithms. Best of all, we can make it possible for people to extract useful material and incorporate it with their work.” Also, from [Phillips91b], pg. 75, “In addition to the expected multi-media components such as graphics, audio and video, MediaView supports several nontraditional components. These include full-color images; object based animations; image-based animations; mathematics; and custom, dynamically loadable components.”
906-1.e: wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation,	MediaView discloses that a hypermedia document is displayed in a browser window.

Claim Text from '906 Patent	MediaView
	 MediaView Document Manager. Search Controls. 0 found. search in: Document, Read-it. Babar. Coincident with the September 8, 1992 3.0 release of NeXTSTEP, Pixar's 3DKit was included. That kit incorporated a version of Pixar's RenderMan system, called QuickRenderMan (QRM). MediaView adopted the QRM technology to allow manipulation and viewing of 3D objects to be embedded into a document. Here is an example of an interactive 3D viewer that has been inserted into this document by dragging an instance of the colored teapot icon from the icon well in the above right of the MediaView manager and dropping it into this document at the current cursor location. Search Find Reset MediaView 3D View Controls Tesselation 8 Motor Colors shape screen Shading The MediaView program described in this is a computer program that was compiled from the MV.TAR produced-file and whose appearance can be seen in the figures cited above. Furthermore, Figure 5 of this report shows a document titled *Usenix* has been selected for browsing and a portion of it is seen in the browser window. What is being displayed there is described in [Phillips91c], pp 7-8, "The user is made aware of the presence of a MediaViewButton by a small magnifying glass icon composited into its active area. This means it is no ordinary

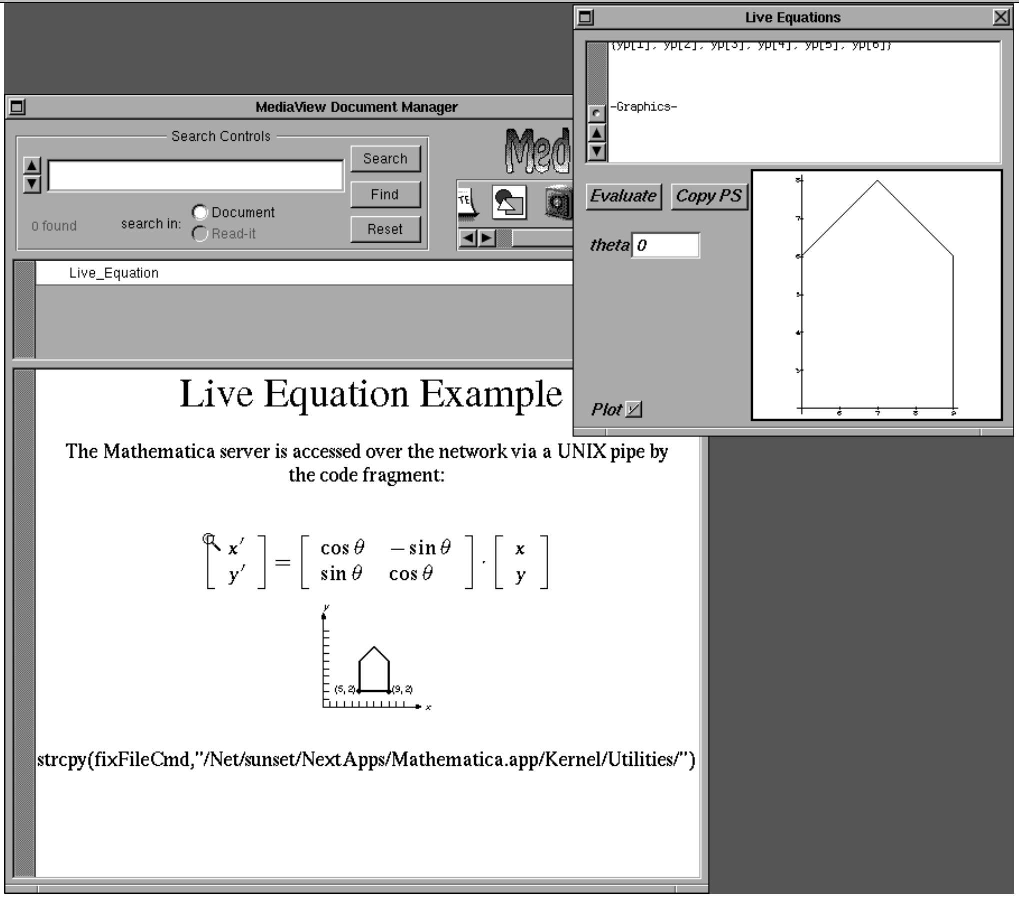
Claim Text from ’906 Patent	MediaView
	image, but is inspectable by clicking the mouse within it. Figure 5 is an example of a MediaViewButton used to animate a circle scan conversion described in [7]. The text of the scan conversion algorithm is all the user normally sees, but when the mouse is clicked within its area, the Algorithm Laboratory window opens. The user can use the mouse to rubber-band an ideal circle and then watch it being approximated by the scan conversion algorithm. The speed of evolution can be adjusted by the slider.”
906-1.f: 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,	MediaView discloses an embed text format at a first location in a hypermedia document. A portion of an embed text format corresponding to the figure shown in the 906-1.e entry above follows: \rtf0\ansi{\fonttbl\f0\fnil Times-Roman;\f1\fswiss Helvetica;} \margl40\margr40\pard\tx520\tx1060\tx1600\tx2120\tx2660\tx3200\tx3720\tx4260\tx4800\tx5320\f0\b0\i0\ulnone\fs36\fc0\cf0 Coincident with the September 8, 1992 3.0 release of NeXTSTEP, Pixar's 3DKit was included. That kit incorporated a version of Pixar's RenderMan system, called QuickRenderMan (QRM). MediaView adopted the QRM technology to allow manipulation and viewing of 3D objects to be incorporated into a document.\n \t\t\t\t\t\t\t ¬ \n Here is an example of an interactive 3D viewer that has been inserted into this document by dragging an instance of the colored teapot icon from the icon well in the above right of the MediaView manager and dropping it into this document at the current cursor location. The above text was captured by running MediaView 3.X with a debugger which made it possible to access the internal representation of the contents of the associated MediaView file. The same document representation techniques were employed in MediaView 2.X and with

Claim Text from ’906 Patent	MediaView

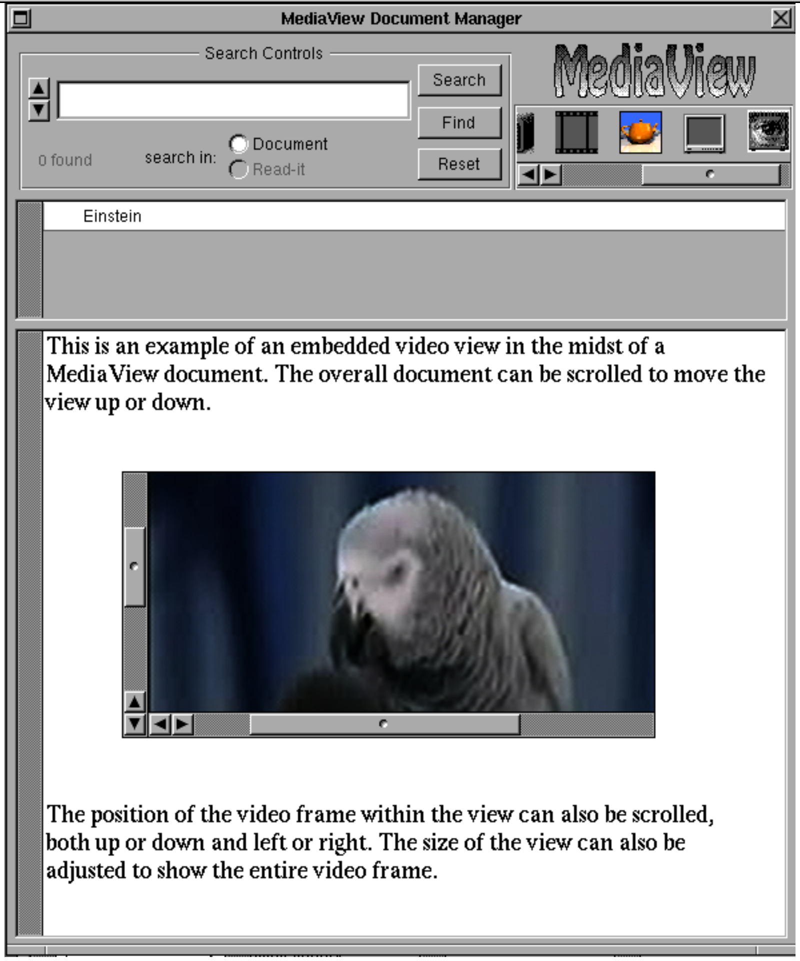
	all the embedded objects found in MediaView files. There is a newline character (\n) at the end of the first paragraph, followed by a line with with 7 tab characters (\t), a space, the not sign (¬), another space, and a newline (\n) character. The ¬ sign, ASCII 172, is a representation of the ViewCell object, which in this case, is an instance of the _3DButton class.
	
	From [Phillips91c], pg 5, “The hash table entries consist of the id of the MediaViewCell and its current integer ordinal position in the text stream.” The code for the parser, represented by the subroutine takeInventory follows on that page and appears in this report. Also, from that same page, “The data structure of theRuns can be found in the description of the Text class [6]. Through this infrastructure MediaView objects can determine their current position in the document —
	`- (int)cellPosition:(MediaViewCell *)cell`
	`{`
	`    return (int)[inventory valueForKey:cell];`
	`}`
	This cellPosition is the MediaView equivalent to a “first location”.
	
	MediaView discloses that the embed text format specifies the location of an object.
	
	As stated above in this report, from [NeXT89], “A Text object or any subclass allows a “graphic character” to be embedded in the text stream. In MediaView “graphic characters” are subclasses of a ViewCell, whose address is the data contained in the info field described in my report under the subsection entitled Parsing. The ViewCell character specifies the processing to be done for the hypermedia component. Thus, if the ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton

Claim Text from ’906 Patent	MediaView
	instance will be rendered by a RenderMan server. Other MediaView embedded interactive hypermedia components exhibit a comparable embed text format behavior. For the following embedded 3D Dataset Viewer example,  MediaView Document Manager Search Controls Search Find Reset 0 found search in: Document Read-it MediaView 3D Viewer Dataset 90 Φ -90 ∞ Δ 1.8 -180 Θ 180 This is a document that contains an embedded 3D dataset viewer. The dataset to be manipulated can be accessed from any site on the network. the relevant embed text format is:

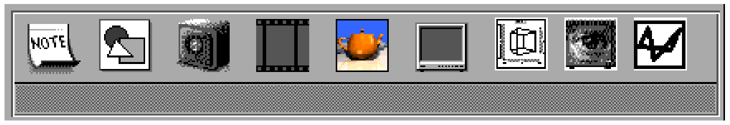
Claim Text from ’906 Patent	MediaView
	\rtf0\\ansi{\\fonttbl\\f1\\fnil Palatino-Roman;\\f0\\fswiss Helvetica;}\n\\margl40\n\\margr40\n\\pard\\tx520\\tx1060\\tx1600\\tx2120\\tx2660\\tx3200\\tx3720\\tx4260\\tx4800\\tx5320\\f1\\b0\\i0\\ulnone\\fs36\\fc0\\cf0 This is a document that contains an embedded 3D dataset viewer.\n\n\t\t\t ¬ \n\t\t\t\t\t\t\t\nThe dataset to be manipulated can be accessed from any site on the network.\n\n\. As for the RenderMan example above, the ¬ sign represents the ViewCell object for the 3D Dataset Viewer. Also, for the following embedded Live Equation example, which accesses the Mathematica server,

Claim Text from ’906 Patent	MediaView
	 MediaView Document Manager Search Controls Search Find Reset 0 found search in: Document Read-it Live_Equation Live Equations Evaluate Copy PS theta 0 Plot {yp[1], yp[2], yp[3], yp[4], yp[5], yp[6]} -Graphics- Live Equation Example The Mathematica server is accessed over the network via a UNIX pipe by the code fragment: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$ strcpy(fixFileCmd,"/Net/sunset/NextApps/Mathematica.app/Kernel/Utilities/") the relevant embed text format is: \rtf0\\ansi{\\fonttbl\\f0\\fnil Times-Roman;\\f1\\fswiss Helvetica;}\n\\margl120\n\\margr120\n{\\colortbl;\\red0\\green0\\blue0;}\n\\pard\\tx1260\\tx3900\\tx5040\\tx6180\\tx7320\\tx8460\\tx9600\\f0\\b0\\i0\\ulnone\\qc\\fs72\\fc1\\cf1 Live Equation Example\n\nThe Mathematica server is accessed over the network via a

Claim Text from ’906 Patent	MediaView
	UNIX pipe by the code fragment:\n\n\n ¬ \n ¬ \n\nstrcpy(fixFileCmd,\"/Net/sunset/NextApps/Mathematica.app/Kernel/Utilities/". Here, the first ¬ sign represents the ViewCell object for the Live Equation and the second ¬ sign represents the non-interactive graph figure. In addition, a MediaView custom component, the EmbeddedVideo viewer displays a live video stream in a sub-window embedded in a MediaView document. An example of this is shown below.

Claim Text from ’906 Patent	MediaView
	 MediaView Document Manager Search Controls Search Find Reset 0 found search in: Document / Read-it MediaView Einstein This is an example of an embedded video view in the midst of a MediaView document. The overall document can be scrolled to move the view up or down. The position of the video frame within the view can also be scrolled, both up or down and left or right. The size of the view can also be adjusted to show the entire video frame. The relevant embed text format for this component is: \rtf0\\ansi{\\fonttbl\\f0\\fnil Times-Roman;\\f1\\fswiss Helvetica;}\n\\margl40\n\\margr40\n\\pard\\tx520\\tx1060\\tx1600\\tx2120\\tx2660\\tx3200\\tx3720\\tx4260\\tx4800\\tx5320\\f0\\b0\\i0

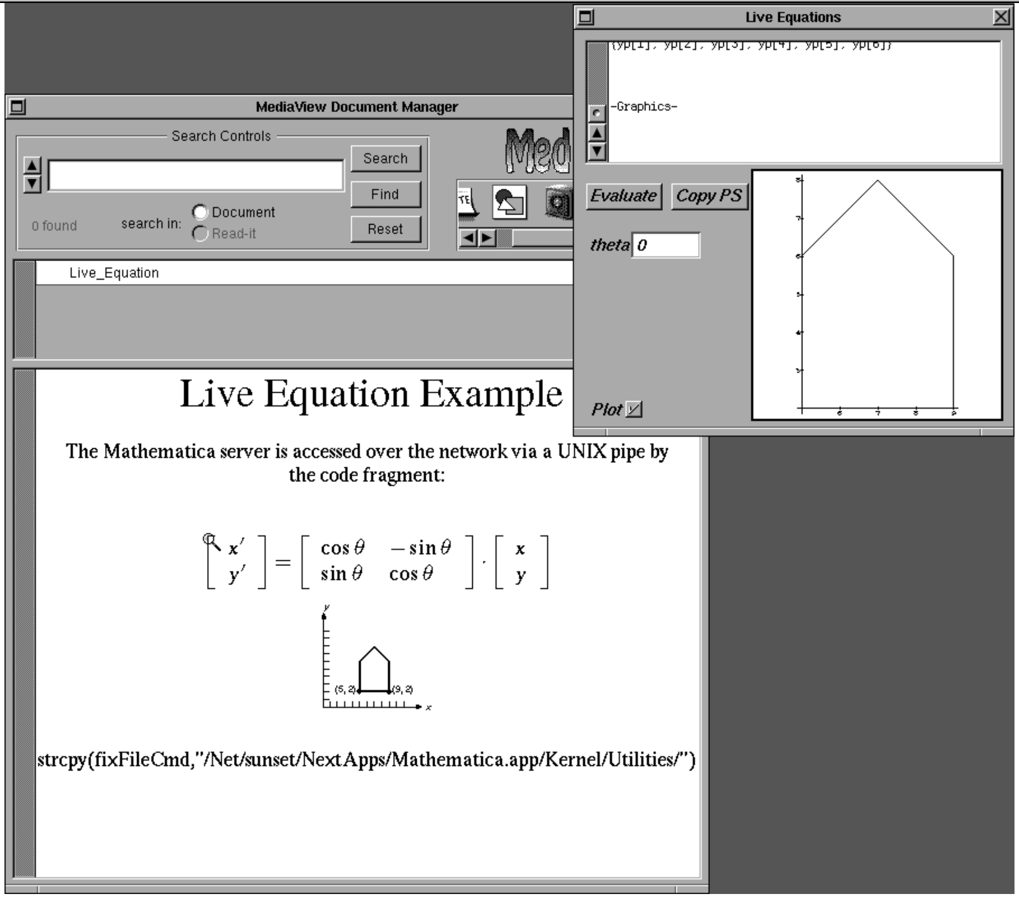
Claim Text from ’906 Patent	MediaView
	\\ulnone\\fs36\\fc0\\cf0 This is an example of an embedded video view in the midst of a MediaView document. The overall document can be scrolled to move the view up or down.\n\n\n\t\t ¬ \n\n\nThe position of the video frame within the view can also be scrolled, both up or down and left or right. The size of the view can also be adjusted to show the entire video frame. As with other embedded component examples, the ¬ sign represents the position of the ViewCell object for the EmbeddedVideo viewer. MediaView discloses an object that is external to a hypermedia document. If, for example, the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. When a document containing a _3DButton class object is read by MediaView from a network location, the information contained in the associated ViewCell object points to a *ribShape* object, which is yet to be read in and activated during the image rendering process. That *ribShape* object contains all the 3D coordinate data that was initially read from a RIB file at the time the _3DButton class object was created.
906-1.g: 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	MediaView discloses that the object has associated type information. MediaView has several types of hypermedia objects that can be embedded in a document. They are represented by the following figure that shows the contents of a fully-populated icon well. The first five represent “standard” hypermedia objects while the remaining four represent custom components. Each object has associated type information, which is contained in the ViewCell object that represents the hypermedia object in the document body.

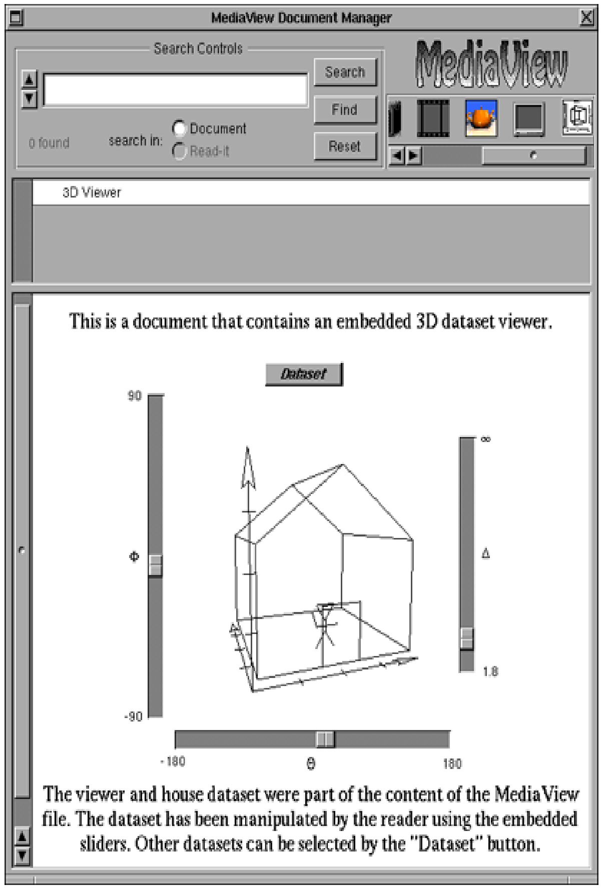
Claim Text from '906 Patent	MediaView
	 In MediaView, the ViewCell character which is parsed out of the text stream contains information about its hypermedia object type and the computer code necessary to perform its function. Thus, if the ViewCell represents a Read-it note, its type is an instance of the Post_itButton class and its function is performed by the post-itAction method (subroutine). Likewise, if the ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. If the ViewCell object represents a Live Equation component its type is an instance of the tiffButton class and its function is peformed by the mathAction method. Also, if the ViewCell object represents a 3D Dataset Viewer component its type is an instance of a a3DViewerView class and its function is performed by any of the setPhi, setTheta or setInvDist methods. Furthermore, if the ViewCell object represents a EmbeddedVideo Viewer component, its type is is an instance of the EmbeddedVideoView class and its function performed is performed by the toggleRun method. MediaView discloses that the browser uses type information to identify and locate an executable application. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is

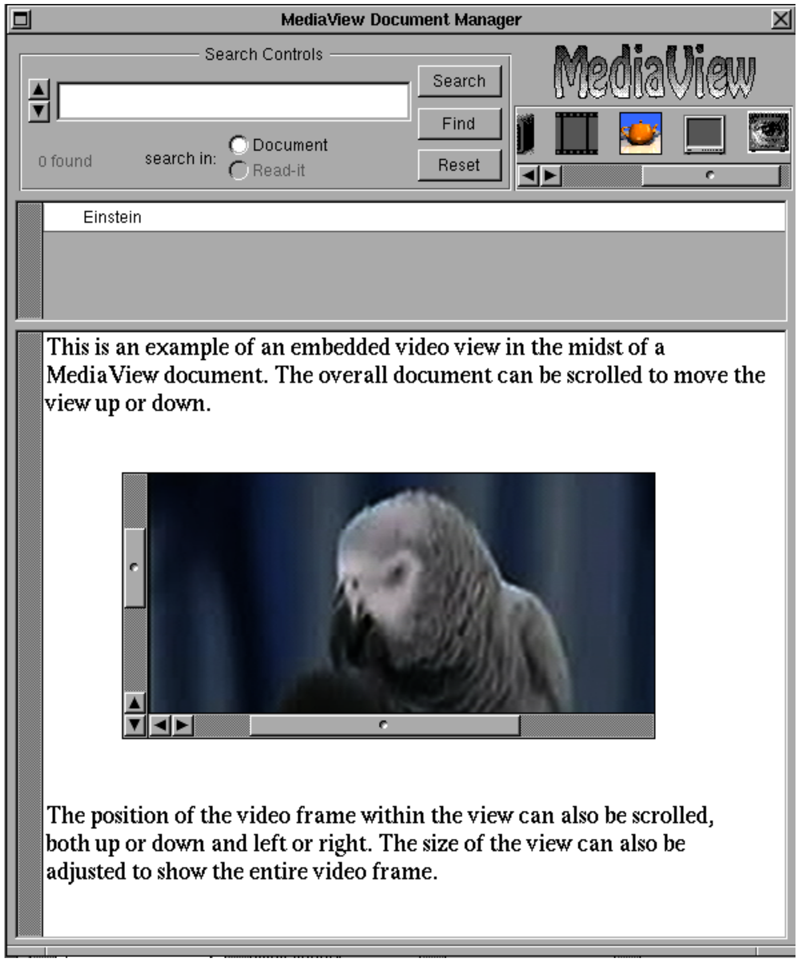
Claim Text from ’906 Patent	MediaView
	performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an executable application. If the ViewCell object represents a Live Equation component its type is an instance of the tiffButton class and its function is peformed by the mathAction method. Moreover, the Live Equation instance will be rendered by a Mathematica server, an executable application. MediaView discloses that the executable application is external to the hypermedia document. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an executable application that is always external to a MediaView (hypermedia) document. If the ViewCell object represents a Live Equation component its type is an instance of the tiffButton class and its function is peformed by the mathAction method. Moreover, the Live Equation instance will be rendered by a Mathematica server, an executable application that is always external to a MediaView (hypermedia) document.
906-1.h: 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	MediaView discloses that the browser parses the embed text format. MediaView parses a stream of data in Rich Text Format (RTF) in order to discover the location of multimedia components embedded therein. It does this by looping through a series of “runs” of RTF data and searching for a “character” with an embedded flag that identifies it as a

Claim Text from ’906 Patent	MediaView
created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.	pointer to a multimedia component. MediaView discloses automatic invocation of the executable application. In MediaView, the ViewCell character parsed out of the text stream contains information about its hypermedia object type and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an executable application, which is automatically invoked, ready for interaction with the client. In MediaView, the ViewCell character parsed out of the text stream contains information about its hypermedia object type and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded Live Equation component, its type is an instance of the tiffButton class and its function is performed by the mathAction method. Moreover, the tiffButton instance will be rendered by a Mathematica server, an executable application, which is automatically invoked, ready for interaction with the client. MediaView discloses that the executable application displays the object. In MediaView, if the ViewCell character is parsed out of the text stream it contains information about its type of hypermedia object and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an executable application, which transmits

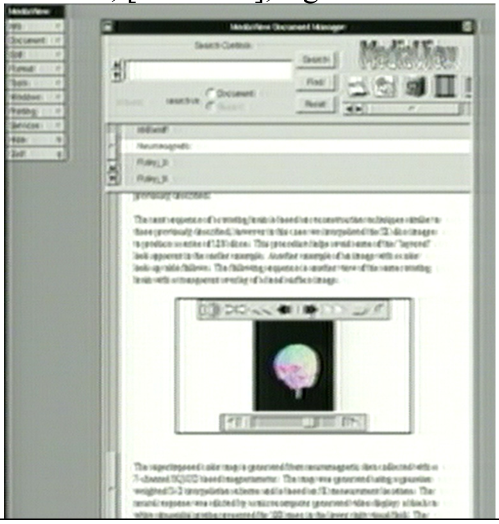
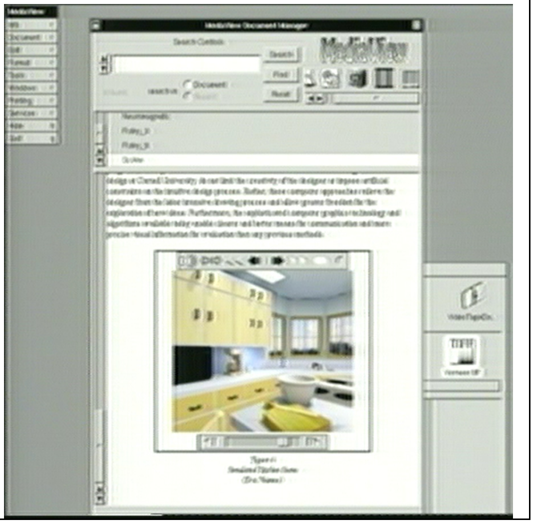
Claim Text from ’906 Patent	MediaView
	graphical data to the object for display. In MediaView, the ViewCell character parsed out of the text stream contains information about its hypermedia object type and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded Live Equation component, its type is an instance of the tiffButton class and its function is performed by the mathAction method. Moreover, the tiffButton instance will be rendered by a Mathematica server, an executable application, which transmits graphical data to the object for display. MediaView discloses that the executable application enables direct interaction with the object. Among the hypermedia objects that interact directly with an executable application is the Live Equation object, depicted below. The equation inspection panel permits direct interaction with a Mathematica server, an application external to the MediaView document.

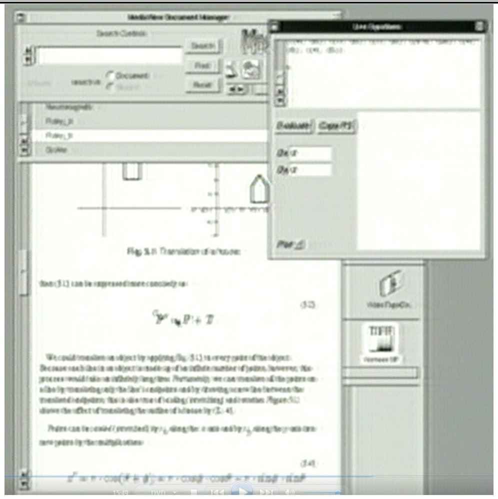
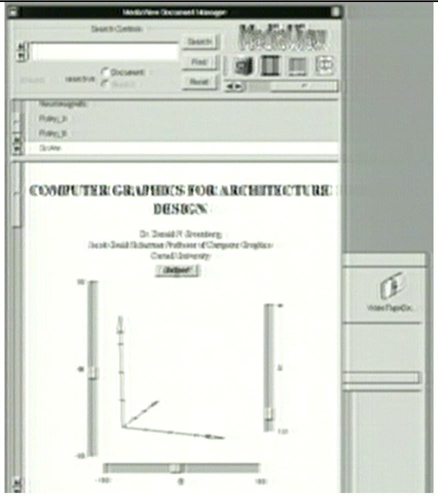
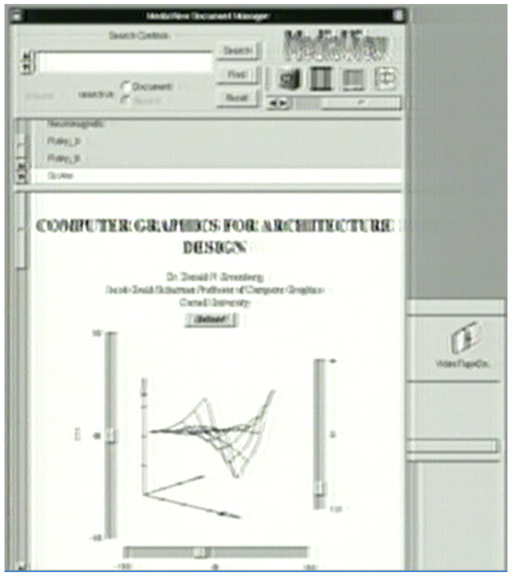
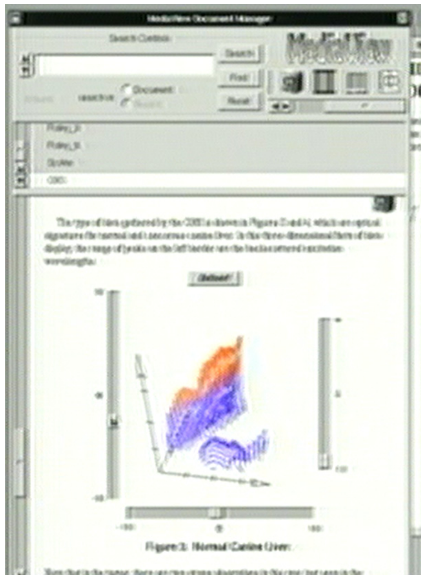
Claim Text from ’906 Patent	MediaView
	 Live Equations {yp[1], yp[2], yp[3], yp[4], yp[5], yp[6]} -Graphics- theta 0 MediaView Document Manager Search Controls 0 found search in: Document Read-it Live_Equation **Live Equation Example** The Mathematica server is accessed over the network via a UNIX pipe by the code fragment: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$ strcpy(fixFileCmd,"/Net/sunset/NextApps/Mathematica.app/Kernel/Utilities/") Evaluate Copy PS Plot Search Find Reset (5, 2) (9, 2) In addition, the 3D Line Dataset Viewer object directly interacts with a geometric transformation application running in conjunction with MediaView and provides access to external 3D datasets located anywhere on a network, as seen below.

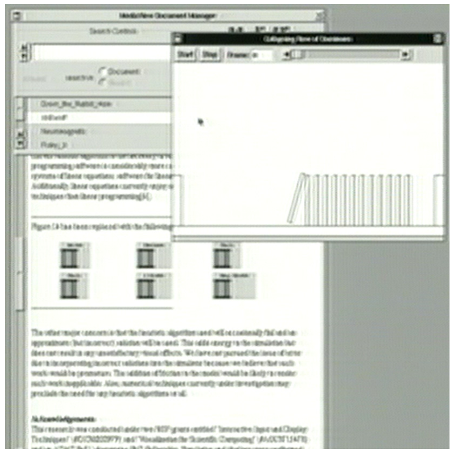
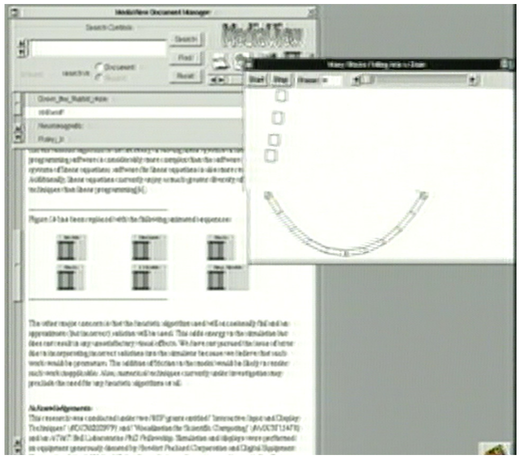
Claim Text from ’906 Patent	MediaView
	 MediaView Document Manager Search Controls Search Find Reset MediaView 0 found search in: Document Read-it 3D Viewer Dataset 90 Φ -90 ∞ Δ 1.8 -180 Θ 180 This is a document that contains an embedded 3D dataset viewer. The viewer and house dataset were part of the content of the MediaView file. The dataset has been manipulated by the reader using the embedded sliders. Other datasets can be selected by the "Dataset" button.

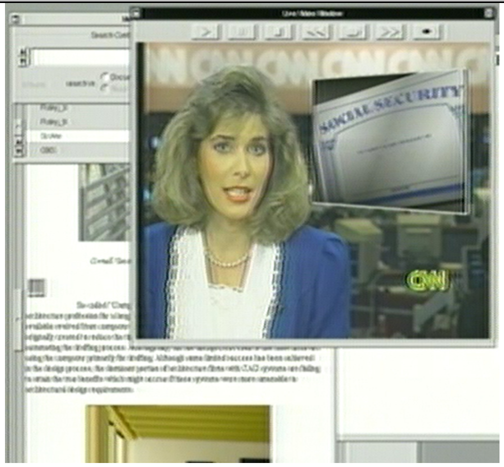
Claim Text from ’906 Patent	MediaView
	As another example, a custom component, the EmbeddedVideo viewer displays a live video data stream in a sub-window embedded in a MediaView document. An example of this component is shown here.  This is an example of an embedded video view in the midst of a MediaView document. The overall document can be scrolled to move the view up or down. The position of the video frame within the view can also be scrolled, both up or down and left or right. The size of the view can also be adjusted to show the entire video frame.

Claim Text from ’906 Patent	MediaView
	In MediaView, the ViewCell character is parsed out of the text stream contains information about its hypermedia object type and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an executable application. The server application is automatically invoked, enabling direct interaction with the 3D button object. In MediaView, the ViewCell character is parsed out of the text stream contains information about its hypermedia object type and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded Live Equation viewer, its type is an instance of the tiffButton class and its function is performed by the mathAction method. Moreover, the tiffButton instance will be rendered by a Mathematica server, an executable application. The server application is automatically invoked, enabling direct interaction with the Live Equation object. MediaView discloses that interaction with the object is at a first location in the hypermedia document. In MediaView, if the ViewCell character is parsed out of the text stream it contains information about its type of hypermedia object and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered

Claim Text from ’906 Patent	MediaView
	by a RenderMan server, an executable application. The server application is automatically invoked, enabling interaction with the 3D button object at its first location in the hypermedia document, i.e. the location in the text stream where it was determined by parsing. In MediaView, the ViewCell character is parsed out of the text stream contains information about its hypermedia object type and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded Live Equation viewer, its type is an instance of the tiffButton class and its function is performed by the mathAction method. Moreover, the tiffButton instance will be rendered by a Mathematica server, an executable application. The server application is automatically invoked, enabling interaction with the Live Equation object at its first location in the hypermedia document, i.e. the location in the text stream where it was determined by parsing. See also, [LANL93], e.g.:  MediaView Document Manager Search Controls Search Find Reset MediaView MediaView Document Manager Search Controls Search Find Reset MediaView TIFF

Claim Text from ’906 Patent	MediaView
	  COMPUTER GRAPHICS FOR ARCHITECTURE DESIGN Dr. Donald P. Greenberg Jacob Gould Schurman Professor of Computer Graphics Cornell University  COMPUTER GRAPHICS FOR ARCHITECTURE DESIGN Dr. Donald P. Greenberg Jacob Gould Schurman Professor of Computer Graphics Cornell University  Figure 3: Normal Canine Liver

Claim Text from ’906 Patent	MediaView
	Also note [LANL93] (while not literally satisfying the limitation, since the window is place separate from the MediaView document in a location that corresponds to the location of the component – here the upper right edge):   [LANL93] also shows video functionality and while it would not satisfy the limitation “within a display area created at said first location within the portion of said first distributed hypermedia document,” as is shown in other embodiments the components could be moved within the document.

Claim Text from '906 Patent	MediaView
	
906-2.a: The method of claim 1, wherein said executable application is a controllable application and further comprising the step of: interactively controlling said controllable application on said client workstation via inter-process communications between said browser and said controllable application.	MediaView discloses interactive control via inter-process communications between a browser and an application. In the case of the objects shown in my report at Figures 6 and 7, the MediaView browser uses inter-process communications between the Live Equation object and the Mathematica server using UNIX pipes. Communication between the 3D button object and RenderMan servers is done using RPC.
906-3.a: The method of claim 2, wherein the communications to interactively control said controllable application continue to be exchanged between the controllable application and the browser even after the controllable application program has been launched.	MediaView discloses ongoing inter-process communications. Once either of the objects shown in Figures 6 and 7 is launched, the MediaView browser permits ongoing inter-process communications between the Live Equation object and the Mathematica server or between the 3D button object and RenderMan servers.
906-6.a: A computer program product for use in a system	MediaView discloses an application program in a computer network environment. *See* evidence recited for 906-1.a.

Claim Text from ’906 Patent	MediaView
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:	MediaView also discloses a client workstation and a network server in a distributed hypermedia environment. *See* evidence recited for 906-1.b.
906-6.b: a computer usable medium having computer readable program code physically embodied therein, said computer program product further comprising:	MediaView discloses computer code physically embodied on a medium. Complete computer source code for MediaView in MV.TAR on PA-NAT-71, which when compiled on a NeXT computer, under NeXTSTEP Release 3.2, produces the executable MediaView browser. Another version is found on [DS 1]. Both versions are substantially as seen in Fig.2 of [Phillips91b], Fig.1 of [Phillips91a and Fig. 5 of this report]. From pg. 75 of [Phillips91b], “MediaView is a multimedia digital publication system that was designed to be flexible and free from restrictions. It was also designed to take maximum advantage of the media-rich hardware and software capabilities of the NeXT [5] computer, especially the features of the NeXTdimension[17] subsystem.” From pg. 78 of [Phillips91b], “The Application Kit is a collection of about 50 classes of commonly used graphics user-interface objects. The application programmer's interface to the Application Kit is based on Objective-C. The style of programming in NeXTstep is to subclass objects in the Application Kit and then to override default behavior or add new behavior. While one's application-specific code can be written in ordinary C, MediaView has benefitted greatly from being programmed primarily in Objective-C.”
906-6.c: computer readable program code for causing said client workstation to execute a browser application to parse a first distributed hypermedia document to	MediaView discloses a browser application that parses a hypermedia document with text formats. *See* evidence recited for 906-1.c.

Claim Text from ’906 Patent	MediaView
identify text formats included in said distributed hypermedia document and to respond to predetermined text formats to initiate processes specified by said text formats;	
906-6.d: 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,	MediaView discloses a hypermedia document received from a server and a browser that displays the hypermedia document. *See* evidence recited for 906-1.d.
906-6.e: wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation,	MediaView discloses that the hypermedia document is displayed in a browser window. *See* evidence recited for 906-1.e.
906-6.f: 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,	MediaView discloses an embed text format at a first location in a hypermedia document; that the embed text format specifies the location of an object; and that the object is external to the hypermedia document. *See* evidence recited for 906-1.f.
906-6.g: 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	MediaView discloses that the object has associated type information, that the browser uses the type information to identify and locate an executable application, and that the executable application is external to the hypermedia document. *See* evidence recited for 906-1.g.
906-6.h: 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	MediaView discloses that the browser parses the embed text format; that the browser automatically invokes the executable application; that the executable application displays the object and enables an end-user to directly interact with it; and that interaction with the object is at a first location in the hypermedia document. *See* evidence recited for 906-1.h.

Claim Text from ’906 Patent	MediaView
created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.	
906-7.a: The computer program product of claim 6, wherein said executable application is a controllable application and further comprising: computer readable program code for causing said client workstation to interactively control said controllable application on said client workstation via inter-process communications between said browser and said controllable application.	MediaView discloses interactive control via inter-process communications between a browser and an application. *See* evidence recited for 906-2.a.
906-8.a: The computer program product of claim 7, wherein the communications to interactively control said controllable application continue to be exchanged between the controllable application and the browser even after the controllable application program has been launched.	MediaView discloses ongoing inter-process communications. *See* evidence recited for 906-3.a.
906-11.a: The method of claim 3, wherein additional instructions for controlling said controllable application reside on said network server, wherein said step of interactively controlling said controllable application includes the following sub steps:	MediaView discloses additional instructions on the server The applications depicted by Figures 6 and 7 in my report are distributed applications where the work done prior to dispatching a task to a server, is done on a client workstation. The task thus dispatched is performed on the appropriate server. Additional instructions on the server control the Mathematica engine (MediaView 2.X and 3.X) or the RenderMan engine (MediaView 3.X).
906-11.b: issuing, from the client workstation, one or more	MediaView discloses that the client issues commands to the server.

Claim Text from ’906 Patent	MediaView
commands to the network server;	The applications depicted by Figures 6 and 7 in my report are distributed applications where the work done prior to dispatching a task to a server, is done on a client workstation. The task thus dispatched is performed on the appropriate server. The control panels seen in Figures 6 and 7 send commands from the client to the server.
906-11.c: executing, on the network server, one or more instructions in response to said commands;	MediaView discloses that the server executes instructions in response to client commands. The applications depicted by Figures 6 and 7 in my report are distributed applications where the work done prior to dispatching a task to a server, is done on a client workstation. The task thus dispatched is performed on the appropriate server. The control panels seen in Figures 6 and 7 send commands from the client to the server. The server executes in response to these commands.
906-11.d: sending information from said network server to said client workstation in response to said executed instructions; and	MediaView discloses that the server responds with information to the client. The applications depicted by Figures 6 and 7 in my report are distributed applications where the work done prior to dispatching a task to a server, is done on a client workstation. The task thus dispatched is performed on the appropriate server. The control panels seen in Figures 6 and 7 send commands from the client to the server. The server executes in response to these commands. Once finished, the server sends resulting information, to the client.
906-11.e: processing said information at the client workstation to interactively control said controllable application.	MediaView discloses that the client uses information from the server to interactively control the application. The applications depicted by Figures 6 and 7 in my report are distributed applications where the work done prior to dispatching a task to a server, is done on a client workstation. The task thus dispatched is performed on the appropriate server. The control panels seen in Figures 6 and 7 send commands from the client to the server. The server executes in response to these commands. Once finished, the server sends resulting information, to

Claim Text from ’906 Patent	MediaView
	the client. Client uses that information to display text and/or graphics.
906-13.a: The computer program product of claim 8, wherein additional instructions for controlling said controllable application reside on said network server, wherein said computer readable program code for causing said client workstation to interactively control said controllable application on said client workstation includes:	MediaView discloses additional instructions on the server *See* evidence recited for 906-11.a.
906-13.b: computer readable program code for causing said client workstation to issue from the client workstation, one or more commands to the network server;	MediaView discloses that the client issues commands to the server. *See* evidence recited for 906-11.b.
906-13.c: computer readable program code for causing said network server to execute one or more instructions in response to said commands;	MediaView discloses that the server executes instructions in response to client commands. *See* evidence recited for 906-11.c.
906-13.d: computer readable program code for causing said network sever to send information to said client workstation in response to said executed instructions; and	MediaView discloses that the server responds with information to the client. *See* evidence recited for 906-11.d.
906-13.e: computer readable program code for causing said client workstation to process said information at the client workstation to interactively control said controllable application.	MediaView discloses that the client uses information from the server to interactively control the application. *See* evidence recited for 906-11.e.

INVALIDITY CLAIM CHART FOR U.S. PATENT NO. 7,599,985

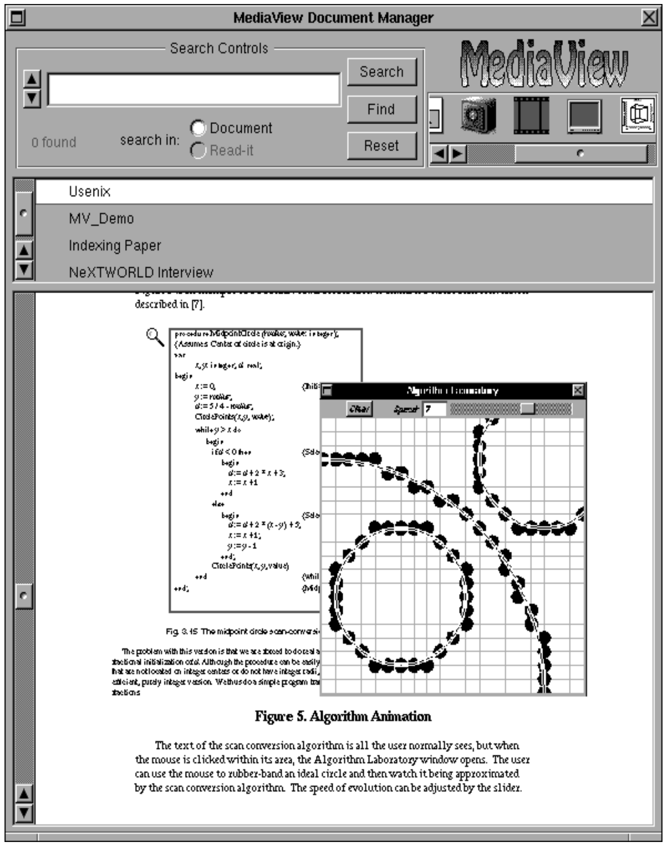
- **"MEDIAVIEW" [2]—AS INTENDED TO BE USED IN A COMPUTER SYSTEM AND DEMONSTRATION OF SAME, INCLUDING INDIVIDUALLY AND COLLECTIVELY:**
  - **An Interpersonal Multimedia Visualization System, IEEE Computer Graphics & Applications, May, 1991 [PA-00273175] [Phillips91a];**
  - **MediaView: An Editable Multimedia Publishing System Developed With An Object-Oriented Toolkit, Proc. USENIX Summer Conf., Nashville, TN, June, 1991 [PA-00327398] [Phillips91b];**
  - **MediaView: A General Multimedia Digital Publication System, Comm. ACM, Vol. 34, No. 7, July, 1991 [PA-00273194] [Phillips91c];**
  - **"Media View, Short and Long Versions" March 1993 [RLP 7];**
  - **Public demonstrations and uses of MediaView including at EDUCOM '89 [RLP 11], SIGGRAPH '90 [RLP 1], SIGGRAPH '92 [DS 1], the Smithsonian Institution in 1991 [RLP 1], NeXTWORLD [RLP 1], and HP's Palo Alto, California (June 1993) and Fort Collins, Colorado (1991) facilities [RLP 10 and RLP 12], and R&D Magazine [Studt94]; and**
  - **Personal experience and knowledge with MediaView and NeXT systems, including my NeXT cube. The body of my report has a narrative description that augments and should be considered part of this chart, and vise-versa, for this an all my charts.**

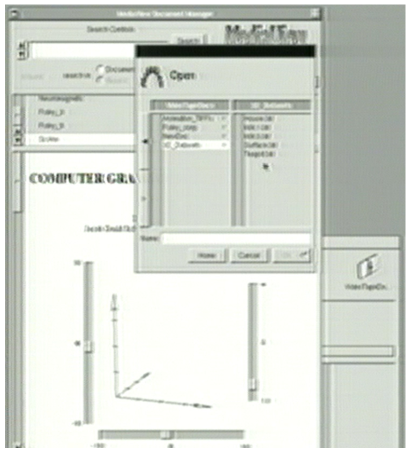
("MEDIAVIEW")

Claim Text from '985 Patent	MediaView
985-1.a:	All instances of MediaView disclose an application program. See also 906-1 for

[2] The features and functions of MediaView are identified singularly for all versions of MediaView except where otherwise noted. MediaView 2.X was completed prior to October 1992 and was distributed through a Purdue server, which is no longer available. See Martin Exhibits 14, 15, and 16. A copy of MediaView 2 that was distributed through SIGGRAPH '92 was produced by Sadowski [DS 1], who attended the conference and a demonstration of MediaView. MediaView was widely shown throughout the years of its development including through 1994, see, e.g., [Studt94] (which shows a RenderMan object). The primary difference for purposes of my analysis between MediaView 2.X and 3.X was the inclusion of the RenderMan custom component, which, like the usage of Mathematica in prior versions of MediaView, employed a client-server configuration for the distribution, manipulation, and interaction with objects embedded in a MediaView document. The versions of MediaView 2.X that I have located to date are from June 11, 1992 (on my NeXT cube) and another from SIGGRAPH '92, though on media dated November 29, 1992. The MediaView 3 (the primary application) was completed around October 1992 (which is shortly after the release of NeXTSTEP 3.0), though the last date for RenderMan related code is November 9, 1992, so I will assume that is the date of MediaView 3.

Claim Text from '985 Patent	MediaView

A method for running an application program in a distributed hypermedia network environment, wherein the network environment comprises at least one client workstation and one network server coupled to the network environment, the method comprising:	discussion of similar limitations.
	
	 MediaView Document Manager
	Search Controls
	Search
	Find
	Reset
	0 found
	search in:
	Document
	Read-it
	MediaView
	Usenix
	MV_Demo
	Indexing Paper
	NeXTWORLD Interview
	Algorithm Laboratory
	Clear
	Speed 7
	described in [7].
	procedure MidpointCircle (radius, value: integer);
	(Assumes Center of circle is at origin.)
	var
	x, y: integer; d: real;
	begin
	x := 0;
	y := radius;
	d := 5 / 4 - radius;
	CirclePoints(x, y, value);
	while y > x do
	begin
	if d < 0 then
	begin
	d := d + 2 * x + 3;
	x := x + 1
	end
	else
	begin
	d := d + 2 * (x - y) + 5;
	x := x + 1;
	y := y - 1
	end;
	CirclePoints(x, y, value)
	end
	end;
	Fig. 3.15 The midpoint circle scan-conversi
	Figure 5. Algorithm Animation
	The text of the scan conversion algorithm is all the user normally sees, but when the mouse is clicked within its area, the Algorithm Laboratory window opens. The user can use the mouse to rubber-band an ideal circle and then watch it being approximated by the scan conversion algorithm. The speed of evolution can be adjusted by the slider.
	
	The MediaView browser application program described in concept a. above is a computer program that was compiled from the MV.TAR produced file and whose appearance can be seen in the figures cited above.
	
	MediaView discloses a computer network environment.
	
	A workstation running MediaView, being part of a client/server network,

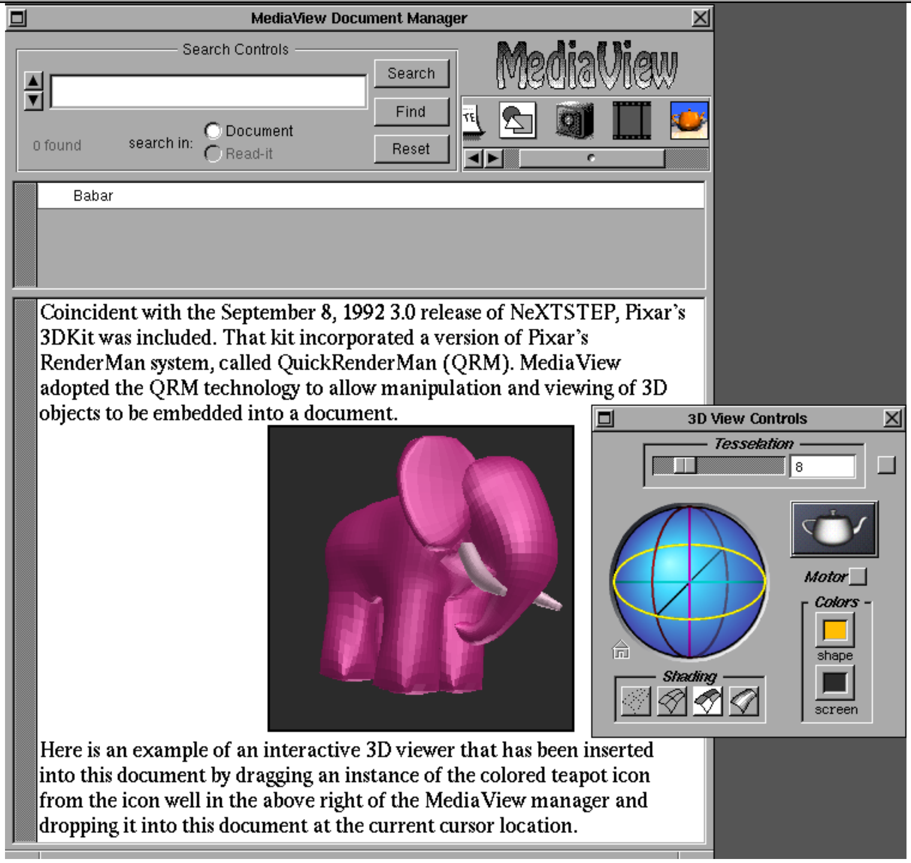
Claim Text from ’985 Patent	MediaView
	operated in a distributed hypermedia environment. MediaView discloses a client workstation. MediaView, having been developed under NeXTSTEP, on a NeXT workstation, runs on a NeXT Computer client workstation. The NeXT workstation was designed for a networked environment, using Ethernet connectivity. Each NeXT computer came with a Network File System (NFS) manager, which enabled client/server connectivity. See also, e.g., [LANL93] (showing the file browser for finding a file, here a dataset that will be processed over the coordinate system):  MediaView discloses a network server. As stated immediately above, MediaView ran in a client/server network environment. Thus, certain computers in the network could, via NFS, be designated as server machines. In this report, referring to Figure 6, et seq.,

Claim Text from ’985 Patent	MediaView
	there is a description of MediaView performing mathematical operations by passing coded requests to a Mathematica server. [MediaView 2.X and 3.X] The server could be anywhere on the network and be accessed by a Remote Procedural Call (RPC). Further along, referring to Figure 7, MediaView rendered 3D geometric datasets using a Pixar RenderMan server. [MediaView 3.X] That server could be anywhere on the network and, indeed, as Figure 8 depicts, there could be multiple servers all working on the same rendering task. MediaView discloses a distributed hypermedia environment. A workstation running MediaView, being part of a client/server network, operated in a distributed hypermedia environment.
985-1.b: 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;	MediaView discloses a browser application. The MediaView browser application program described in concept a. above is a computer program that was compiled from the MV.TAR produced-file and whose appearance can be seen in the figures cited above. Furthermore, Figure 5 of this report shows a document titled Usenix has been selected for browsing and a portion of it is seen in the browser window. MediaView discloses a file containing enabling information. The MediaView browser application program described in concept a. above is a computer program that was compiled from the MV.TAR produced-file and whose appearance can be seen in the figures cited above. Furthermore, Figure 5 of this report shows a document titled Usenix has been selected for browsing and a portion of it is seen in the browser window. What is being displayed there is described in [Phillips91c], pp 7-8, “The user is made aware of the presence of a MediaViewButton by a small magnifying glass icon composited into its

Claim Text from ’985 Patent	MediaView
	active area. This means it is no ordinary image, but is inspectable by clicking the mouse within it. Figure 5 is an example of a MediaViewButton used to animate a circle scan conversion described in [7]. The text of the scan conversion algorithm is all the user normally sees, but when the mouse is clicked within its area, the Algorithm Laboratory window opens. The user can use the mouse to rubber-band an ideal circle and then watch it being approximated by the scan conversion algorithm. The speed of evolution can be adjusted by the slider.” MediaView discloses that the file is received at the client workstation from the network server. As stated above, using NFS, MediaView could receive saved and stored MediaView-specific documents from any server permitting that operation. MediaView documents are files with a suffix *mdvw. MediaView discloses that the browser displays at least a portion of a distributed hypermedia document. The MediaView browser application program displays hypermedia documents, documents that contain not only text but a wide variety of hypermedia components. From [Phillips91a], pp. 20-21, “Armed with modern workstation technology, we can provide an electronic reading environment. ‘Documents’ read in this environment can include not only text, line art, and still images, but also sound, video sequences, and computer-produced animations. And, when cast in digital form, the mathematical content of a document can be symbolically and numerically manipulated. Thus, we can experiment with the mathematics, derive new results, and simulate different situations with different parameters. We can explore computer generated images by moving the eye point, changing the lighting conditions, or using the underlying model with our own algorithms. Best

Claim Text from '985 Patent	MediaView
	of all, we can make it possible for people to extract useful material and incorporate it with their work." Also, from [Phillips91b], pg. 75, "In addition to the expected multi-media components such as graphics, audio and video, MediaView supports several nontraditional components. These include full-color images; object based animations; image-based animations; mathematics; and custom, dynamically loadable components." MediaView discloses that at least a portion of a hypermedia document is displayed in a browser-controlled window. The MediaView browser application program described in concept a. above is a computer program that was compiled from the MV.TAR produced-file and whose appearance can be seen in the figures cited above. Furthermore, Figure 5 of this report shows a document titled Usenix has been selected for browsing and a portion of it is seen in the browser window. What is being displayed there is described in [Phillips91c], pp 7-8, "The user is made aware of the presence of a MediaViewButton by a small magnifying glass icon composited into its active area. This means it is no ordinary image, but is inspectable by clicking the mouse within it. Figure 5 is an example of a MediaViewButton used to animate a circle scan conversion described in [7]. The text of the scan conversion algorithm is all the user normally sees, but when the mouse is clicked within its area, the Algorithm Laboratory window opens. The user can use the mouse to rubber-band an ideal circle and then watch it being approximated by the scan conversion algorithm. The speed of evolution can be adjusted by the slider."
985-1.c: executing the browser application on the client workstation, with the browser application:	MediaView discloses a browser application executing on the client workstation.

Claim Text from ’985 Patent	MediaView

Claim Text from ’985 Patent	MediaView
	 MediaView Document Manager Search Controls Search Find Reset 0 found search in: Document Read-it MediaView Babar 3D View Controls Tesselation 8 Motor Colors shape screen Shading Coincident with the September 8, 1992 3.0 release of NeXTSTEP, Pixar’s 3DKit was included. That kit incorporated a version of Pixar’s RenderMan system, called QuickRenderMan (QRM). MediaView adopted the QRM technology to allow manipulation and viewing of 3D objects to be embedded into a document. Here is an example of an interactive 3D viewer that has been inserted into this document by dragging an instance of the colored teapot icon from the icon well in the above right of the MediaView manager and dropping it into this document at the current cursor location. The MediaView browser application program described in concept a. above is a computer program that was compiled from the MV.TAR produced-file and whose appearance can be seen in the figures cited above. Furthermore, Figure 5 of this report shows a document titled Usenix has been selected for browsing and a portion of it is seen in the browser window.
985-1.d: responding to text formats to initiate processing	MediaView discloses responding to text formats to initiate processing specified by the text formats, i.e., parsing text formats.

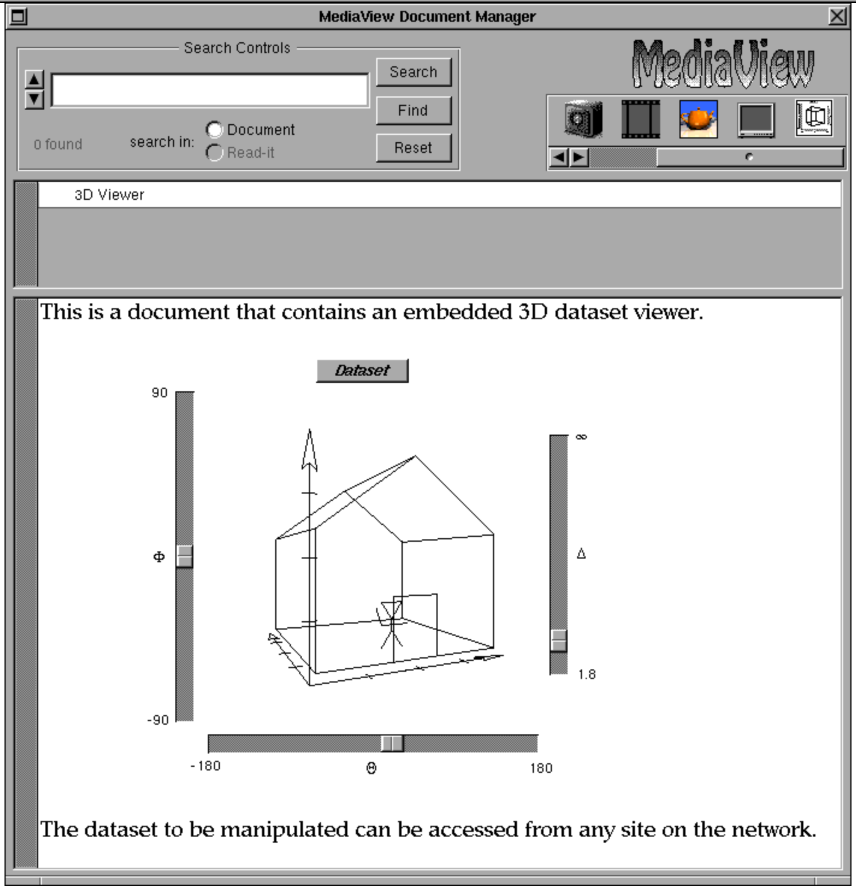
Claim Text from ’985 Patent	MediaView
specified by the text formats;	In this report, under the heading “Parsing” the digital structure of a MediaView document is described. There it states, “MediaView…parses a stream of data in Rich Text Format (RTF) in order to discover the location of multimedia components embedded therein. It does this by looping through a series of “runs” of RTF data and searching for a “character” with an embedded flag that identifies it as a pointer to a multimedia component.”
985-1.e: displaying at least a portion of the document within the browser-controlled window;	MediaView discloses that the browser displays a hypermedia document. The MediaView browser application program displays hypermedia documents, documents that contain not only text but a wide variety of hypermedia components. From [Phillips91a], pp. 20-21, “Armed with modern workstation technology, we can provide an electronic reading environment. ‘Documents’ read in this environment can include not only text, line art, and still images, but also sound, video sequences, and computer-produced animations. And, when cast in digital form, the mathematical content of a document can be symbolically and numerically manipulated. Thus, we can experiment with the mathematics, derive new results, and simulate different situations with different parameters. We can explore computer generated images by moving the eye point, changing the lighting conditions, or using the underlying model with our own algorithms. Best of all, we can make it possible for people to extract useful material and incorporate it with their work.” Also, from [Phillips91b], pg. 75, “In addition to the expected multi-media components such as graphics, audio and video, MediaView supports several nontraditional components. These include full-color images; object based animations; image-based animations; mathematics; and custom, dynamically loadable components.”

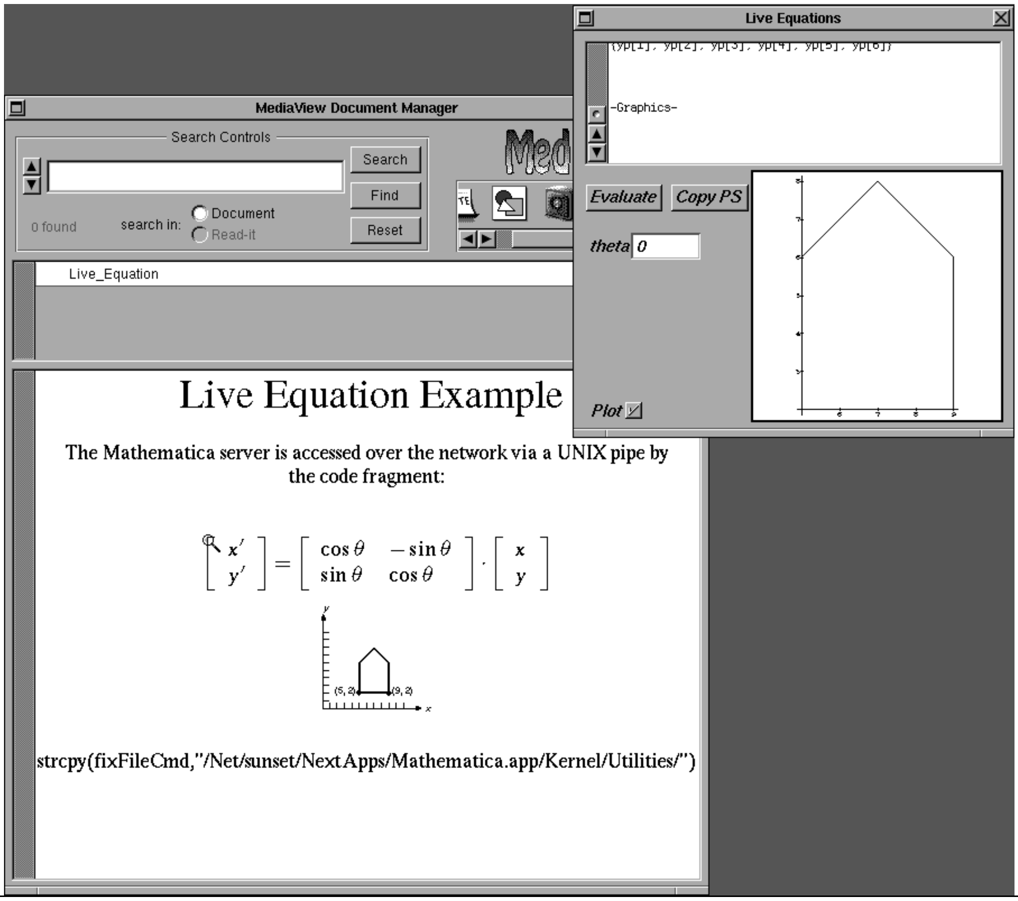
Claim Text from ’985 Patent	MediaView
	MediaView discloses that a hypermedia document is displayed in a browser window. The MediaView browser application program described in concept a. above is a computer program that was compiled from the MV.TAR produced-file and whose appearance can be seen in the figures cited above. Furthermore, Figure 5 of this report shows a document titled Usenix has been selected for browsing and a portion of it is seen in the browser window. What is being displayed there is described in [Phillips91c], pp 7-8, “The user is made aware of the presence of a MediaViewButton by a small magnifying glass icon composited into its active area. This means it is no ordinary image, but is inspectable by clicking the mouse within it. Figure 5 is an example of a MediaViewButton used to animate a circle scan conversion described in [7]. The text of the scan conversion algorithm is all the user normally sees, but when the mouse is clicked within its area, the Algorithm Laboratory window opens. The user can use the mouse to rubber-band an ideal circle and then watch it being approximated by the scan conversion algorithm. The speed of evolution can be adjusted by the slider.”
985-1.f: identifying an embed text format which corresponds to a first location in the document, where the embed text format specifies the location of at least a portion of an object external to the file, where the object has type information associated with it;	MediaView discloses identifying an embed text format. MediaView parses a stream of data in Rich Text Format (RTF) in order to discover the location of multimedia components embedded therein. It does this by looping through a series of “runs” of RTF data and searching for a “character” with an embedded flag that identifies it as a pointer to a multimedia component. MediaView discloses that the embed text format corresponds to a first location in the hypermedia document. A portion of an embed text format corresponding to the figure shown in the 985-1.c entry above follows: `\rtf0\ansi{\fonttbl\f0\fnil Times-Roman;\f1\fswiss Helvetica;}`

Claim Text from ’985 Patent	MediaView

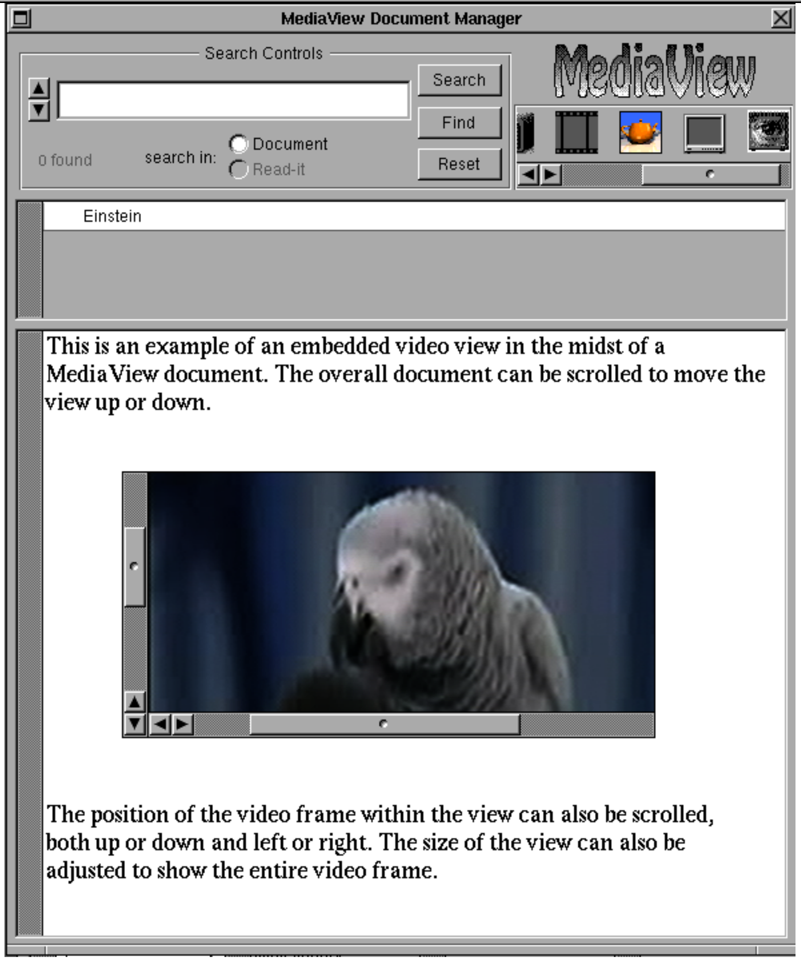
	\margl40\margr40\pard\tx520\tx1060\tx1600\tx2120\tx2660\tx3200\tx3720\tx4260\tx4800\tx5320\f0\b0\i0\ulnone\fs36\fc0\cf0 Coincident with the September 8, 1992 3.0 release of NeXTSTEP, Pixar's 3DKit was included. That kit incorporated a version of Pixar's RenderMan system, called QuickRenderMan (QRM). MediaView adopted the QRM technology to allow manipulation and viewing of 3D objects to be incorporated into a document.\n
	\t\t\t\t\t\t\t ¬ \n
	Here is an example of an interactive 3D viewer that has been inserted into this document by dragging an instance of the colored teapot icon from the icon well in the above right of the MediaView manager and dropping it into this document at the current cursor location.
	
	The above text was captured by running MediaView 3.X with a debugger which made it possible to access the internal representation of the contents of the associated MediaView file. The same document representation techniques were employed in MediaView 2.X and with all of the embedded objects found in MediaView files. There is a newline character (\n) at the end of the first paragraph, followed by a line with with 7 tab characters (\t), a space, the not sign (¬), another space, and a newline (\n) character. The ¬ sign, ASCII 172, is a representation of the ViewCell object, which in this case, is an instance of the _3DButton class.
	
	From [Phillips91c], pg 5, “The hash table entries consist of the id of the MediaViewCell and its current integer ordinal position in the text stream.” Through this infrastructure MediaView objects can determine their current position in the document —
	- (int)cellPosition:(MediaViewCell *)cell
	{
	return (int)[inventory valueForKey:cell];
	}

Claim Text from ’985 Patent	MediaView
	This cellPosition is the MediaView equivalent to a “first location”. MediaView discloses that the embed text format specifies the location of an object. As stated above in this report, from [NeXT89], “A Text object or any subclass allows a “graphic character” to be embedded in the text stream. In MediaView “graphic characters” are subclasses of a ViewCell, whose address is the data contained in the info field described in my report under the subsection entitled Parsing. The ViewCell character directly specifies the processing to be done for the hypermedia component. Thus, if the ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server. Other MediaView embedded interactive hypermedia components exhibit a comparable embed text format behavior. For the following embedded 3D Dataset Viewer example,

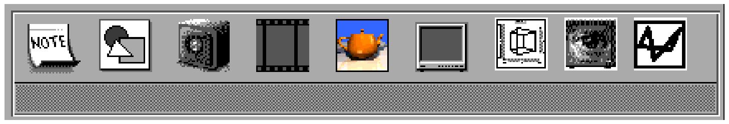
Claim Text from '985 Patent	MediaView
	 MediaView Document Manager Search Controls Search Find Reset 0 found search in: Document Read-it MediaView 3D Viewer Dataset 90 Φ -90 -180 Θ 180 ∞ Δ 1.8 This is a document that contains an embedded 3D dataset viewer. The dataset to be manipulated can be accessed from any site on the network. the relevant embed text format is: \rtf0\\ansi{\\fonttbl\\f1\\fnil Palatino-Roman;\\f0\\fswiss Helvetica;}\n\\margl40\n\\margr40\n\\pard\\tx520\\tx1060\\tx1600\\tx2120\\tx2660\\tx3200\\tx3720\\tx4260\\tx4800\\tx5320\\f1\\b0\\i0\\ulnone\\fs36\\fc0\\cf0 This is a document that contains an embedded 3D dataset viewer.\n\n\t\t\t ¬ \n\t\t\t\t\t\t\t\nThe dataset to be

Claim Text from ’985 Patent	MediaView
	manipulated can be accessed from any site on the network.\n\n\. As for the RenderMan example above, the ¬ sign represents the ViewCell object for the 3D Dataset Viewer. Also, for the following embedded Live Equation example, which accesses the Mathematica server,  MediaView Document Manager Search Controls Search Find Reset 0 found search in: Document Read-it Live_Equation Live Equations Evaluate Copy PS theta 0 Plot (5, 2) (9, 2) {yp[1], yp[2], yp[3], yp[4], yp[5], yp[6]} -Graphics- Live Equation Example The Mathematica server is accessed over the network via a UNIX pipe by the code fragment: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$ strcpy(fixFileCmd,"/Net/sunset/NextApps/Mathematica.app/Kernel/Utilities/")

Claim Text from ’985 Patent	MediaView
	the relevant embed text format is: \rtf0\\ansi{\\fonttbl\\f0\\fnil Times-Roman;\\f1\\fswiss Helvetica;}\n\\margl120\n\\margr120\n{\\colortbl;\\red0\\green0\\blue 0;}\n\\pard\\tx1260\\tx3900\\tx5040\\tx6180\\tx7320\\tx8460\\tx9600\ \f0\\b0\\i0\\ulnone\\qc\\fs72\\fc1\\cf1 Live Equation Example\n \nThe Mathematica server is accessed over the network via a UNIX pipe by the code fragment:\n\n\n ¬ \n ¬ \n\nstrcpy(fixFileCmd,\"/Net/sunset/NextApps/Mathematica.app/Kernel/Utilities/". Here, the first ¬ sign represents the ViewCell object for the Live Equation and the second ¬ sign represents the non-interactive graph figure. In addition, a MediaView custom component, the EmbeddedVideo viewer displays a live video stream in a sub-window embedded in a MediaView document. An example of this is shown below.

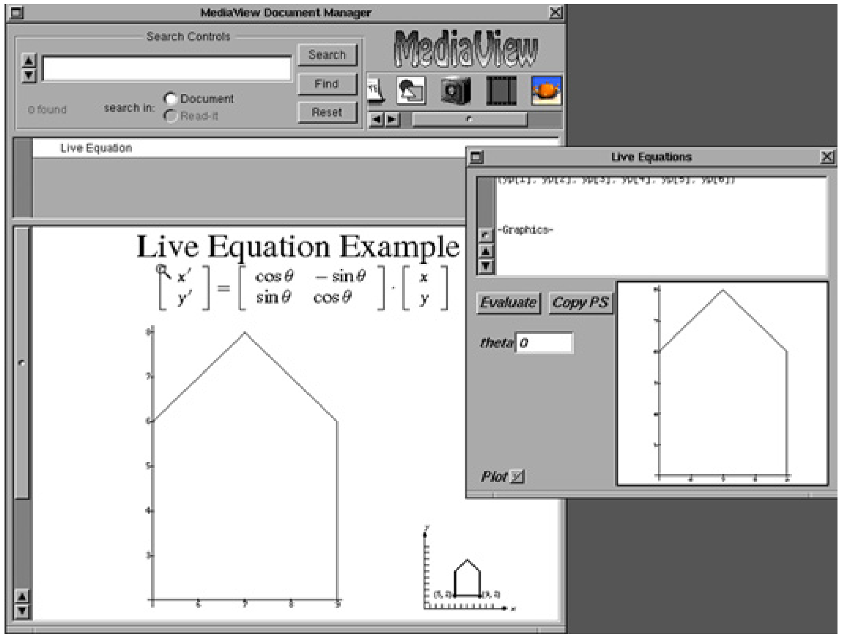
Claim Text from '985 Patent	MediaView
	 MediaView Document Manager Search Controls Search Find Reset 0 found search in: Document Read-it MediaView Einstein This is an example of an embedded video view in the midst of a MediaView document. The overall document can be scrolled to move the view up or down. The position of the video frame within the view can also be scrolled, both up or down and left or right. The size of the view can also be adjusted to show the entire video frame. The relevant embed text format for this component is: \rtf0\\ansi{\\fonttbl\\f0\\fnil Times-Roman;\\f1\\fswiss Helvetica;}\n\\margl40\n\\margr40\n\\pard\\tx520\\tx1060\\tx1600\\tx2120\\tx2660\\tx3200\\tx3720\\tx4260\\tx4800\\tx5320\\f0\\b0\\i0\\uln

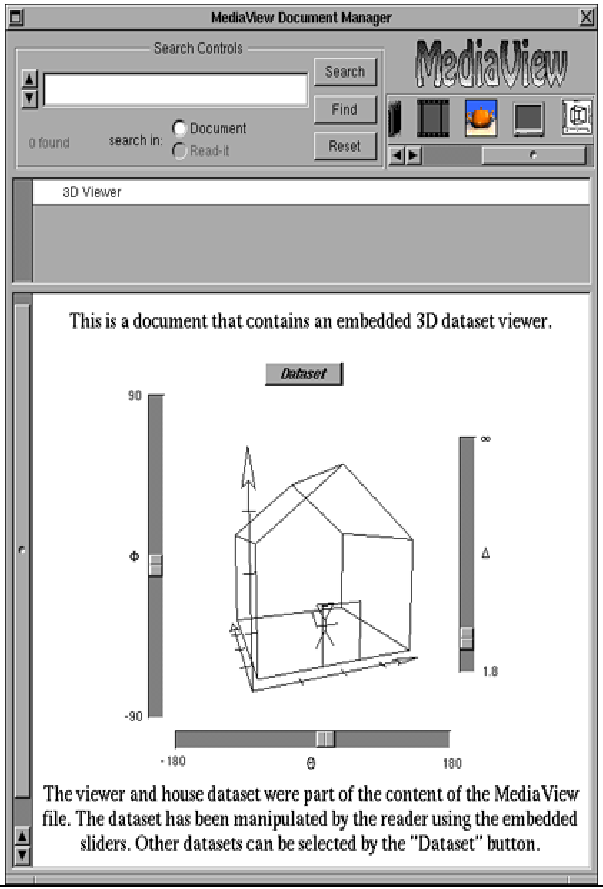
Claim Text from ’985 Patent	MediaView
	one\\fs36\\fc0\\cf0 This is an example of an embedded video view in the midst of a MediaView document. The overall document can be scrolled to move the view up or down.\n\n\n\t\t ¬ \n\n\nThe position of the video frame within the view can also be scrolled, both up or down and left or right. The size of the view can also be adjusted to show the entire video frame. As with other embedded component examples, the ¬ sign represents the position of the ViewCell object for the EmbeddedVideo viewer. MediaView discloses that the object is external to the file containing enabling information. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. When a document containing a _3DButton class object is read by MediaView from a network location, the information contained in the associated ViewCell object points to a *ribShape* object, which is yet to be read in and activated during the image rendering process. That *ribShape* object contains all the 3D coordinate data that was initially read from a RIB file at the time the _3DButton class object was created. MediaView discloses that the object has associated type information. MediaView has several types of hypermedia objects that can be embedded in a document. They are represented by the following figure that shows the contents of a fully-populated icon well. The first five represent “standard” hypermedia objects while the remaining four represent custom components. Each object has associated type information, which is contained in the ViewCell object that represents the hypermedia object in the document body.

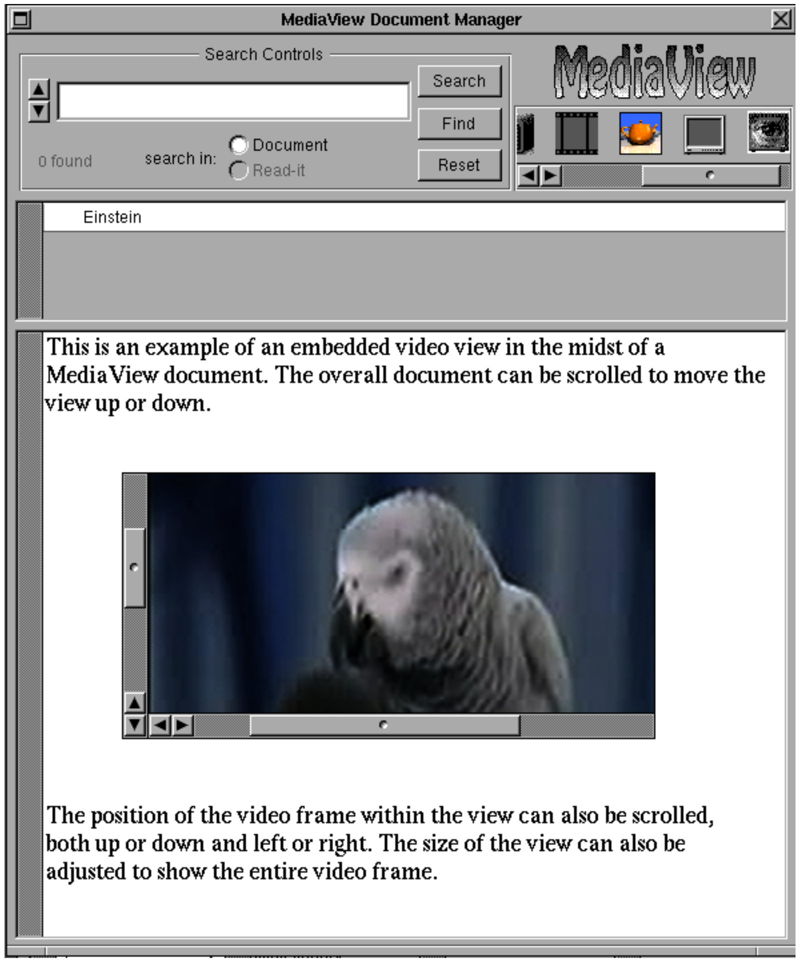
Claim Text from ’985 Patent	MediaView
	 In MediaView, the ViewCell character parsed out of the text stream contains information about its type of hypermedia object and the computer code necessary to perform its function. Thus, if the ViewCell represents a Read-it note, its type is an instance of the Post_itButton class and its function is performed by the post-itAction method (subroutine). Likewise, if the ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. If the ViewCell object represents a Live Equation component its type is an instance of the tiffButton class and its function is peformed by the mathAction method. Also, if the ViewCell object represents a 3D Dataset Viewer component its type is an instance of a a3DViewerView class and its function is performed by any of the setPhi, setTheta or setInvDist methods. Furthermore, if the ViewCell object represents a EmbeddedVideo Viewer component, its type is is an instance of the EmbeddedVideoView class and its function performed is performed by the toggleRun method.
985-1.g: utilizing the type information to identify and locate an executable application external to the file; and	MediaView discloses that the browser uses type information to identify and locate an executable application. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an executable application.

Claim Text from '985 Patent	MediaView
	If the ViewCell object represents a Live Equation component its type is an instance of the tiffButton class and its function is peformed by the mathAction method. Moreover, the Live Equation instance will be rendered by a Mathematica server, an executable application. MediaView discloses that the executable application is external to the file containing enabling information. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an executable application that is always external to a MediaView file containing enabling information. If the ViewCell object represents a Live Equation component its type is an instance of the tiffButton class and its function is peformed by the mathAction method. Moreover, the Live Equation instance will be rendered by a Mathematica server, an executable application that is always external to a MediaView (hypermedia) document.
985-1.h: 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.	MediaView discloses that the browser parses the embed text format. MediaView parses a stream of data in Rich Text Format (RTF) in order to discover the location of multimedia components embedded therein. It does this by looping through a series of "runs" of RTF data and searching for a "character" with an embedded flag that identifies it as a pointer to a multimedia component. MediaView discloses automatic invocation of the executable application. In MediaView, if the ViewCell character is parsed out of the text stream it contains information about its type of hypermedia object and the computer

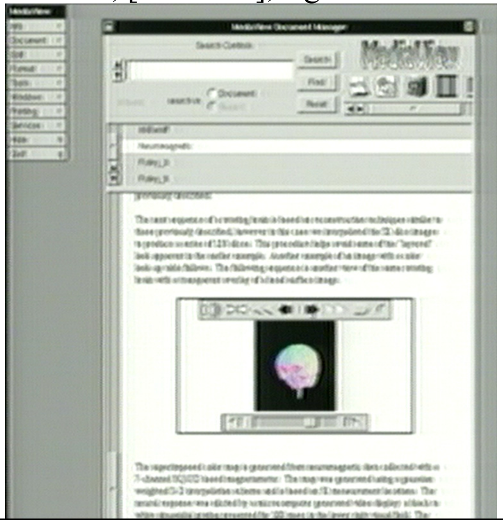
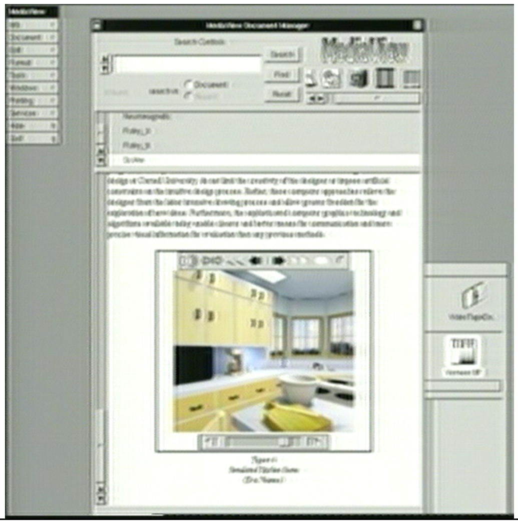
Claim Text from ’985 Patent	MediaView
	code necessary to perform its function. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an executable application, which is automatically invoked, ready for interaction with the client. In MediaView, the ViewCell character parsed out of the text stream contains information about its hypermedia object type and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded Live Equation component, its type is an instance of the tiffButton class and its function is performed by the mathAction method. Moreover, the tiffButton instance will be rendered by a Mathematica server, an executable application, which is automatically invoked, ready for interaction with the client. MediaView discloses that the executable application displays the object. In MediaView, if the ViewCell character is parsed out of the text stream it contains information about its type of hypermedia object and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an executable application, which transmits graphical data to the object for display. In MediaView, the ViewCell character parsed out of the text stream contains information about its hypermedia object type and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded Live Equation component, its type is an instance of the tiffButton class and its function is performed by the mathAction

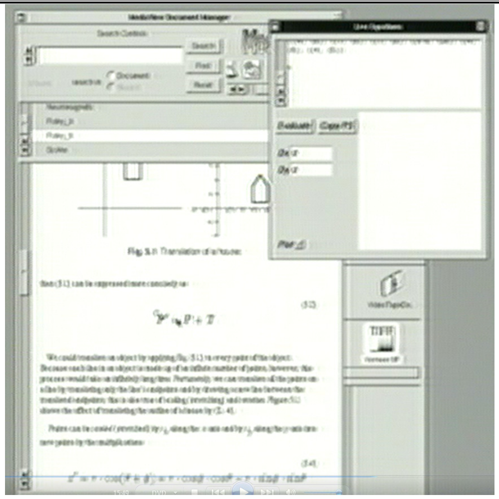
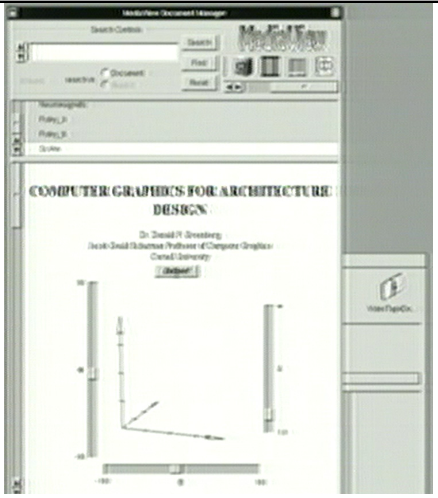
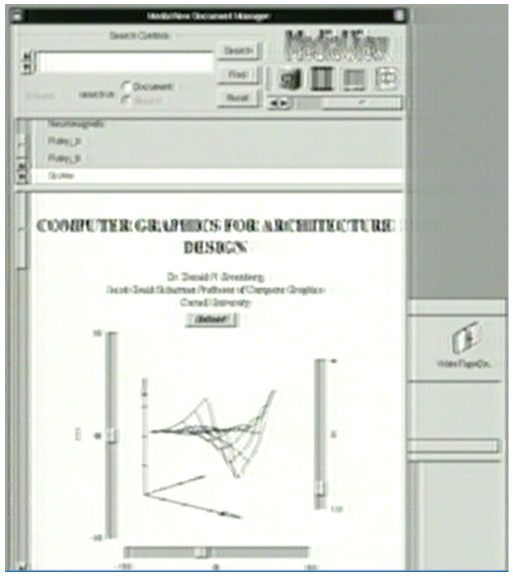
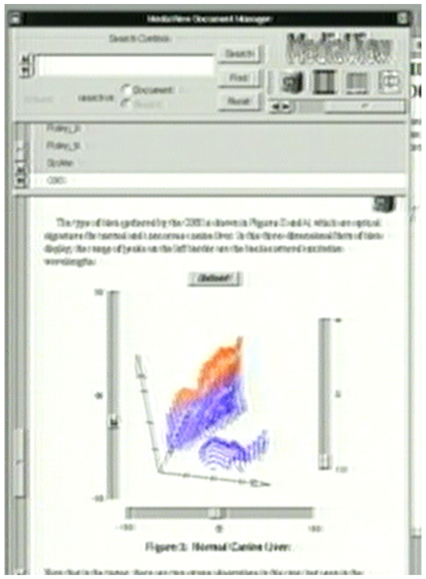
Claim Text from '985 Patent	MediaView
	method. Moreover, the tiffButton instance will be rendered by a Mathematica server, an executable application, which transmits graphical data to the object for display. MediaView discloses that the executable application enables direct interaction with the object. Among the hypermedia objects that interact directly with an executable application is the Live Equation object, depicted below. The equation inspection panel permits direct interaction with a Mathematica server, an application external to the MediaView document.  MediaView Document Manager Search Controls Search Find Reset 0 found search in: Document Read-it MediaView Live Equation Live Equations -Graphics- Evaluate Copy PS theta 0 Plot Live Equation Example $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$ In addition, the 3D Line Dataset Viewer object directly interacts with a geometric transformation application running in conjunction with

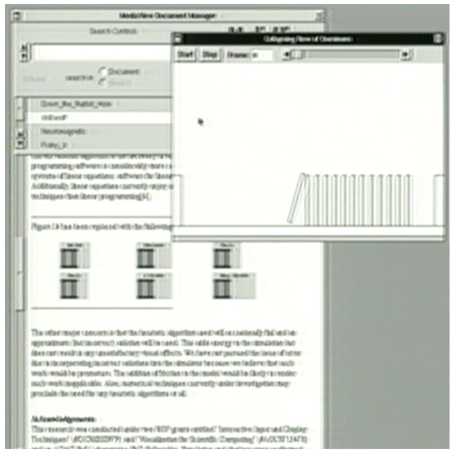
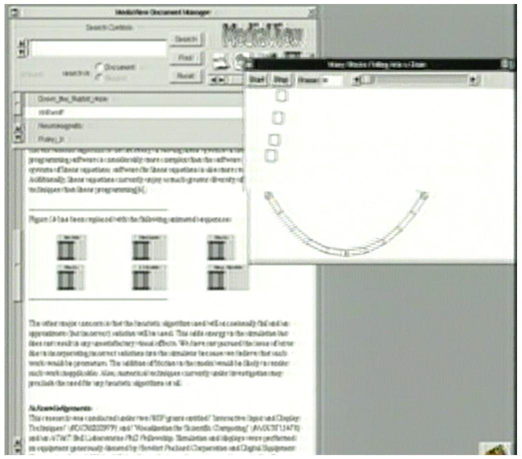
Claim Text from '985 Patent	MediaView
	MediaView and provides access to external 3D datasets located anywhere on a network, as seen below.  MediaView Document Manager · Search Controls · Search · Find · Reset · 0 found · search in: Document, Read-it · 3D Viewer · This is a document that contains an embedded 3D dataset viewer. · Dataset · The viewer and house dataset were part of the content of the MediaView file. The dataset has been manipulated by the reader using the embedded sliders. Other datasets can be selected by the "Dataset" button.

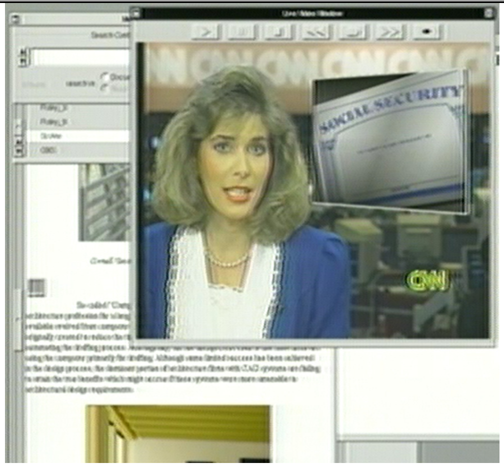
Claim Text from ’985 Patent	MediaView
	As another example, a custom component, the EmbeddedVideo viewer displays a live video data stream in a sub-window embedded in a MediaView document. An example of this component is shown here.  MediaView Document Manager Search Controls Search Find Reset 0 found search in: Document Read-it MediaView Einstein This is an example of an embedded video view in the midst of a MediaView document. The overall document can be scrolled to move the view up or down. The position of the video frame within the view can also be scrolled, both up or down and left or right. The size of the view can also be adjusted to show the entire video frame.

Claim Text from ’985 Patent	MediaView
	In MediaView, if the ViewCell character is parsed out of the text stream it contains information about its type of hypermedia object and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an executable application. The server application is automatically invoked, enabling direct interaction with the 3D button object. In MediaView, the ViewCell character is parsed out of the text stream contains information about its hypermedia object type and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded Live Equation viewer, its type is an instance of the tiffButton class and its function is performed by the mathAction method. Moreover, the tiffButton instance will be rendered by a Mathematica server, an executable application. The server application is automatically invoked, enabling direct interaction with the Live Equation object. MediaView discloses that interaction with the object is at a first location in the hypermedia document. In MediaView, if the ViewCell character is parsed out of the text stream it contains information about its type of hypermedia object and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an executable application. The server application is automatically invoked, enabling interaction with the 3D button object at its first location

Claim Text from ’985 Patent	MediaView
	in the hypermedia document, i.e. the location in the text stream where it was determined by parsing. In MediaView, the ViewCell character is parsed out of the text stream contains information about its hypermedia object type and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded Live Equation viewer, its type is an instance of the tiffButton class and its function is performed by the mathAction method. Moreover, the tiffButton instance will be rendered by a Mathematica server, an executable application. The server application is automatically invoked, enabling interaction with the Live Equation object at its first location in the hypermedia document, i.e. the location in the text stream where it was determined by parsing. See also, [LANL93], e.g.:  MediaView Document Manager Search Controls Search Find Reset  MediaView Document Manager Search Controls Search Find Reset

Claim Text from ’985 Patent	MediaView
	  COMPUTER GRAPHICS FOR ARCHITECTURE DESIGN Dr. Donald P. Greenberg Jacob Gould Schurman Professor of Computer Graphics Cornell University  COMPUTER GRAPHICS FOR ARCHITECTURE DESIGN Dr. Donald P. Greenberg Jacob Gould Schurman Professor of Computer Graphics Cornell University  Figure 3: Normal Canine Liver

Claim Text from ’985 Patent	MediaView
	Also note [LANL93] (while not literally satisfying the limitation, the window is place separate from the MediaView document in a location that corresponds to the location of the component – here the upper right edge):  MediaView Document Manager Search Controls Start Stop Frame: MediaView Document Manager Search Controls Search Find Reset Start Stop Frame: [LANL93] also shows video functionality and while it would not satisfy the limitation “displayed within a display area created at the first location within the portion of the hypermedia document being displayed,” as is shown in other embodiments the components could be moved within the document.

Claim Text from ’985 Patent	MediaView
	
985-2.a: The method of claim 1 where: the information to enable comprises text formats.	MediaView discloses that the enabling information in the file is text formats. In this report, under the heading “Parsing” the digital structure of a MediaView document is described. There it states, “MediaView…parses a stream of data in Rich Text Format (RTF) in order to discover the location of multimedia components embedded therein. It does this by looping through a series of “runs” of RTF data and searching for a “character” with an embedded flag that identifies it as a pointer to a multimedia component.” That pointer causes hypermedia components like those shown in Figures 6 and 7 of my report to become active and be displayed. The markup that handles the formatting in a MediaView document are text formats under the Court’s construction. Examples are show in the dumps found in 985-1.f.
985-3.a: The method of claim 2 where the text formats are HTML tags.	This limitation is not expressly taught in MediaView, though it is obvious in its own right to a person of ordinary skill in the art at the time of the alleged invention for the reasons described in my report, which include the teachings on the CERN website maintained by Tim Berners-Lee and his citation to and

Claim Text from ’985 Patent	MediaView
	interest in my work, the inventors knowledge of it, and thus the combination of the features and functionality of MediaView with any web browser, including but not limited to NCSA’s Mosaic browser, CERN’s browser, or Wei’s Viola browser. HTML, as described in my report, is a type of markup language. My report further explains that RTF was described as also being a markup language. Just as HTML provides tags for specifying a format for rendering text on the screen, RTF does too. Other tags in HTML can specify actions to be taken, like including hypermedia data or executing an application. RTF, as used in the MediaViewText object, provides exactly the same capabilities. For example, the ViewCell character parsed out of the text stream contains information about its type of hypermedia object and the computer code necessary to perform its function. Thus, if the ViewCell represents a Read-it note, its type is an instance of the Post_itButton class and its function is performed by the post-itAction method (subroutine). Likewise, if the ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. As further evidence, I’d point out that in [Phillips91b] it is noted at pg. 82 that "The most obvious and important enhancement is a hyperlinking capability. This has been designed and will be implemented in the next few months. Its design draws upon the rich NeXT development environment, in particular the suite of BTree classes that are available." Furthermore, in this report at ¶94 I have pointed out that while at CERN, Tim Berners-Lee implemented his web browser/editor using NeXTSTEP by sub-classing the Text class to form the HyperText class. That browser parsed HTML.
985-4.a: The method of claim 1 where the information contained in the file received comprises at least one embed text format.	MediaView discloses that the enabling information in the file includes an embed text format. As stated above in this report, “A Text object or any subclass allows a “graphic character” to be embedded in the text stream. In MediaView

Claim Text from ’985 Patent	MediaView
	“graphic characters” are subclasses of a ViewCell, whose address is the data contained in the info field described in my report under the subsection entitled Parsing. The ViewCell character directly specifies the processing to be done for the hypermedia component.
985-5.a: The method of claim 1 where the step of identifying an embed text format comprises: parsing the received file to identify text formats included in the received file.	MediaView discloses that the embed text format is identified by parsing the file containing enabling information. MediaView parses a stream of data in Rich Text Format (RTF) in order to discover the location of multimedia components embedded therein. It does this by looping through a series of “runs” of RTF data and searching for a “character” with an embedded flag that identifies it as a pointer to a multimedia component.
985-6.a: The method of claim 5 where the parsing is by a parser in the browser.	MediaView discloses that the parser is in the browser In this report, under the heading “Parsing” the digital structure of a MediaView document is described. There it states, “MediaView…parses a stream of data in Rich Text Format (RTF) in order to discover the location of multimedia components embedded therein. It does this by looping through a series of “runs” of RTF data and searching for a “character” with an embedded flag that identifies it as a pointer to a multimedia component.” The parser is an integral part of the MediaView browser as discussed in [Phillips91c]. There, on pg. 5, it states, “The design of the inventorying scheme, updating its dynamic data structure, and associated routines that access it was one of the challenges met in developing MediaView. Briefly, the important components are a HashTable (a common class that is part of the Objective-C environment [6]) for maintaining the runtime data structure and a method for populating it. The hash table entries consist of the id of the MediaViewCell and its current integer ordinal position in the text stream.” The code for the parser, represented by the subroutine takeInventory follows on that page and

Claim Text from ’985 Patent	MediaView
	appears in this report.
985-7.a: The method of claim 1 where the processing specified by the text formats is specified directly.	MediaView discloses that the text formats directly specify the processing. As stated above in this report, “A Text object or any subclass allows a “graphic character” to be embedded in the text stream. As stated in [NeXT89], NeXT’s Text Class description: “Each graphic is treated as a single character: The text’s line height and character placement are adjusted to accommodate the graphic “character.” Graphics are embedded in the text in either of two ways: programmatically or directly through user actions. In the programmatic approach, you add an object—generally a subclass of Cell—to the text. This object will manage the graphic image by drawing it when appropriate.” In MediaView “graphic characters” are subclasses of a ViewCell, whose address is the data contained in the info field described in my report under the subsection Parsing. The ViewCell character directly specifies the processing to be done for the hypermedia component.
985-8.a: The method of claim 1 where the correspondence is implied by the order of the text format in a set of all of the text formats.	MediaView discloses that the correspondence is implied by the order of text formats. The “graphic characters” embedded in a text stream appear on the screen in the rendered Text object exactly in the same position as they appear in the stream.
985-9.a: The method of claim 1 where the embed text format specifies the location of at least a portion of an object directly.	MediaView discloses that the embed text format specifies the location of the object directly. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is

Claim Text from ’985 Patent	MediaView
	performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an executable application. If the parsed ViewCell character represents an embedded Live Equation viewer, its type is an instance of the tiffButton class and its function is performed by the mathAction method. Moreover, the tiffButton instance will be rendered by a Mathematica server, an executable application.
985-10.a: The method of claim 1 where having type information associated is by including type information in the embed text format.	MediaView discloses that the type information is in the embed text format. In MediaView, if the ViewCell character is parsed out of the text stream it contains information about its type of hypermedia object and the computer code necessary to perform its function. Thus, if the ViewCell represents a Read-it note, its type is an instance of the Post_itButton class and its function is performed by the post-itAction method (subroutine). Likewise, if the ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Likewise, if the ViewCell character represents an embedded Live Equation viewer, its type is an instance of the tiffButton class and its function is performed by the mathAction method.
985-11.a: The method of claim 1 where automatically invoking does not require interactive action by the user.	MediaView discloses that automatic invocation does not require interactive action by the user. In MediaView, if the ViewCell character is parsed out of the text stream it contains information about its type of hypermedia object and the computer code necessary to perform its function. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan

Claim Text from ’985 Patent	MediaView
	server, an executable application, which is automatically invoked, ready for interaction with the client, the user. If the parsed ViewCell character represents an embedded Live Equation viewer, its type is an instance of the tiffButton class and its function is performed by the mathAction method. Moreover, the tiffButton instance will be rendered by a Mathematica server, an executable application, which is automatically invoked, ready for interaction with the client, the user. In addition, should the parsing operation discover an object represented by the 3D Line Dataset Viewer, as seen in Figure 9 of this report, the viewer is immediately and automatically active, awaiting user input. The situation would be the same for an instance of a EmbeddedVideo viewer.
985-16.a: One or more computer readable media encoded with software comprising computer executable instructions, for use in a distributed hypermedia network environment, wherein the network environment comprises at least one client workstation and one network server coupled to the network environment, and when the software is executed operable to:	MediaView discloses computer code physically embodied on a medium. Complete computer source code for MediaView in MV.TAR on PA-NAT-71 (and has been produced elsewhere too, e.g. [LANL92]), which when compiled on a NeXT computer, under NeXTSTEP Release 3.2, produces the executable MediaView browser seen in Fig.2 of [Phillips91b] , Fig.1 of [Phillips91a and Fig. 5 of this report]. From pg. 75 of [Phillips91b], “MediaView is a multimedia digital publication system that was designed to be flexible and free from restrictions. It was also designed to take maximum advantage of the media-rich hardware and software capabilities of the NeXT [5] computer, especially the features of the NeXTdimension[17] subsystem.” From pg 78 of [Phillips91b], “The Application Kit is a collection of about 50 classes of commonly used graphics user-interface objects. The application programmer's interface to the Application Kit is based on Objective-C. The style of programming in NeXTstep is to subclass objects in the Application Kit and then to override default behavior or add new

Claim Text from ’985 Patent	MediaView
	behavior. While one's application-specific code can be written in ordinary C, MediaView has benefitted greatly from being programmed primarily in Objective-C.” MediaView discloses a client workstation and a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.
985-16.b: 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;	MediaView discloses a browser application; a file containing enabling information received from a server; that the browser displays at least a portion of a distributed hypermedia document; and that the display is in a browser-controlled window. *See* evidence recited for 985-1.b.
985-16.c: cause the client workstation to utilize the browser to:	MediaView discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c.
985-16.d: respond to text formats to initiate processing specified by the text formats;	MediaView discloses parsing text formats. *See* evidence recited for 985-1.d.
985-16.e: display at least a portion of the document within the browser-controlled window;	MediaView discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.
985-16.f: identify an embed text format corresponding to a first location in the document, the embed text format specifying the location of at least a portion of an object external to the file, with the object having type information associated with it;	MediaView discloses identifying an embed text format; that the embed text format corresponds to a first location in a hypermedia document; that the embed text format specifies the location of at least a portion of an object external to the file containing enabling information; and that the object has associated type information. *See* evidence recited for 985-1.f.
985-16.g: utilize the type information to identify and locate an executable application external to the file; and	MediaView discloses using type information to identify and locate an executable application external to the file. *See* evidence recited for 985-1.g.
985-16.h:	MediaView discloses automatically invoking the executable application; that the

Claim Text from '985 Patent	MediaView
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 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.	executable application displays the object and enables an end-user to directly interact with it; and that the interaction with the object is at a first location in a hypermedia document. *See* evidence recited for 985-1.h.
985-17.a: The computer readable media of claim 16 where: the information to enable comprises text formats.	MediaView discloses that the enabling information in the file is text formats. *See* evidence recited for 985-2.a.
985-18.a: The computer readable media of claim 17 where: the text formats are HTML tags.	This limitation is obvious for the same reasons as discussed in the evidence for 985-3.a.
985-19.a: The computer readable media of claim 16 where: the information contained in the file received comprises at least one embed text format.	MediaView discloses that the enabling information in the file includes an embed text format. *See* evidence recited for 985-4.a.
985-20.a: A method of serving digital information in a computer network environment having a network server coupled the network environment, and where the network environment is a distributed hypermedia environment, the method comprising:	MediaView discloses digital information. All information contained in a MediaView document and displayed in a MediaView window is represented digitally. In this report, under the heading "Parsing" the digital structure of a MediaView document is described. There it states, "MediaView…parses a stream of data in Rich Text Format (RTF) in order to discover the location of multimedia components embedded therein. It does this by looping through a series of "runs" of RTF data and searching for a "character" with an embedded flag

Claim Text from ’985 Patent	MediaView
	that identifies it as a pointer to a multimedia component.” MediaView discloses a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.
985-20.b: communicating via the network server with at least one client workstation over said network in order to cause said client workstation to:	MediaView discloses a client workstation. *See* evidence recited for 985-1.a. MediaView discloses communicating via network server in order to cause the client workstation to act. The applications depicted by Figures 6 and 7 are distributed applications where the work done prior to dispatching a task to a server, is done on a client workstation. The task thus dispatched is performed on the appropriate server. Once a task is partially or completely finished, the server sends resulting information, e.g. text or graphics, which causes the client workstation to act.
985-20.c: 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 within a browser-controlled window;	MediaView discloses a browser application; a file containing enabling information received from a server; that the browser displays at least a portion of a distributed hypermedia document; and that the display is in a browser-controlled window. *See* evidence recited for 985-1.b.
985-20.d: execute, at said client workstation, a browser application, with the browser application:	MediaView discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c.
985-20.e: responding to text formats to initiate processing specified by the text formats;	MediaView discloses parsing text formats. *See* evidence recited for 985-1.d.
985-20.f: displaying, on said client workstation, at least a portion of the document within the browser-controlled window;	MediaView discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.
985-20.g:	MediaView discloses identifying an embed text format; that the embed text

Claim Text from ’985 Patent	MediaView
identifying an embed text format which corresponds to a first location in the document, where the embed text format specifies the location of at least a portion of an object external to the file, where the object has type information associated with it;	format corresponds to a first location in a hypermedia document; that the embed text format specifies the location of at least a portion of an object external to the file containing enabling information; and that the object has associated type information. *See* evidence recited for 985-1.f.
985-20.h: utilizing the type information to identify and locate an executable application external to the file; and	MediaView discloses using type information to identify and locate an executable application external to the file. *See* evidence recited for 985-1.g.
985-20.i: 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.	MediaView discloses automatically invoking the executable application; that the executable application displays the object and enables an end-user to directly interact with it; and that the interaction with the object is at a first location in a hypermedia document. *See* evidence recited for 985-1.h.
985-21.a: The method of claim 20 where: the information to enable comprises text formats.	MediaView discloses that the enabling information in the file is text formats. *See* evidence recited for 985-2.a.
985-22.a: The method of claim 21 where: the text formats are HTML tags.	This limitation is obvious for the same reasons as discussed in the evidence for 985-3.a.
985-23.a: The method of claim 20 where: the information contained in the file received comprises at least one embed text format.	MediaView discloses that the enabling information in the file includes an embed text format. *See* evidence recited for 985-4.a.

Claim Text from ’985 Patent	MediaView
985-24.a: A method for running an executable application in a computer network environment, wherein said network environment has at least one client workstation and one network server coupled to a network environment, the method comprising:	MediaView discloses a client workstation and a network server in a network environment. *See* evidence recited for 985-1.a. MediaView discloses an executable application. *See* evidence recited for 985-1.g.
985-24.b: 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,	MediaView discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e. MediaView discloses an object external to a file containing enabling information. *See* evidence recited for 985-1.f. MediaView discloses that there is enabling of an end-user to directly interact with the object. The MediaView objects depicted in Figures 6, 7, and 9 all enable an end-user interaction. MediaView discloses that the interaction with the object is at a first location in a hypermedia document. *See* evidence recited for 985-1.h. MediaView discloses that the object is displayed at a first location within a portion of the hypermedia document being displayed. The “graphic characters” embedded in a text stream appear on the screen in the rendered Text object exactly in the same location (first location) as they appear in the stream.
985-24.c: wherein said network environment is a distributed hypermedia environment,	MediaView discloses a client workstation and a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.
985-24.d:	MediaView discloses a browser application; a file containing enabling

Claim Text from ’985 Patent	MediaView
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,	information received from a server; that the browser displays at least a portion of a distributed hypermedia document; and that the display is in a browser-controlled window. *See* evidence recited for 985-1.b.
985-24.e: wherein said executable application is external to said file,	MediaView discloses an executable application external to the file. *See* evidence recited for 985-1.g.
985-24.f: wherein said client workstation executes the browser application, with the browser application responding to text formats to initiate processing specified by the text formats,	MediaView discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c. MediaView discloses parsing text formats. *See* evidence recited for 985-1.d.
985-24.g: wherein at least said portion of the document is displayed within the browser-controlled window,	MediaView discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.
985-24.h: wherein an embed text format which corresponds to said first location in the document is identified by the browser,	MediaView discloses identifying an embed text format and that the embed text format corresponds to a first location in a hypermedia document. *See* evidence recited for 985-1.f.
985-24.i: wherein the embed text format specifies the location of at least a portion of said object external to the file,	MediaView discloses that the embed text format specifies the location of at least a portion of an object external to the file containing enabling information. *See* evidence recited for 985-1.f.
985-24.j: wherein the object has type information associated with it,	MediaView discloses that the object has associated type information. *See* evidence recited for 985-1.f.
985-24.k: wherein the type information is utilized by the browser to identify and locate said executable application, and	MediaView discloses using type information to identify and locate an executable application external to the file. *See* evidence recited for 985-1.g.

Claim Text from ’985 Patent	MediaView
985-24.l: wherein the executable application is automatically invoked by the browser, in response to the identifying of the embed text format.	MediaView discloses automatically invoking the executable application. *See* evidence recited for 985-1.h.
985-25.a: The method of claim 24 where: the information to enable comprises text formats.	MediaView discloses that the enabling information in the file is text formats. *See* evidence recited for 985-2.a.
985-26.a: The method of claim 25 where: the text formats are HTML tags.	This limitation is obvious for the same reasons as discussed in the evidence for 985-3.a.
985-27.a: The method of claim 24 where: the information contained in the file received comprises at least one embed text format.	MediaView discloses that the enabling information in the file includes an embed text format. *See* evidence recited for 985-4.a.
985-28.a: 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 coupled to a network environment, operable to:	MediaView discloses computer code physically embodied on a medium. *See* evidence recited for 985-16.a. MediaView discloses a client workstation and a network server in a network environment. *See* evidence recited for 985-1.a. MediaView discloses an executable application. *See* evidence recited for 985-1.g.
985-28.b: 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	MediaView discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e. MediaView discloses an object external to a file containing enabling information. *See* evidence recited for 985-1.f.

Claim Text from '985 Patent	MediaView
displayed in a browser-controlled window,	MediaView discloses that there is enabling of an end-user to directly interact with the object. *See* evidence recited for 985-24.b. MediaView discloses that the interaction with the object is at a first location in a hypermedia document. *See* evidence recited for 985-1.h. MediaView discloses that the object is displayed within a display area created at the first location.. The "graphic characters" embedded in a text stream appear on the screen in the rendered Text object exactly in the same location (first location) as they appear in the stream.
985-28.c: wherein said network environment is a distributed hypermedia environment,	MediaView discloses a client workstation and a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.
985-28.d: 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,	MediaView discloses a browser application; a file containing enabling information received from a server; that the browser displays at least a portion of a distributed hypermedia document; and that the display is in a browser-controlled window. *See* evidence recited for 985-1.b.
985-28.e: wherein said executable application is external to said file,	MediaView discloses an executable application external to the file. *See* evidence recited for 985-1.g.
985-28.f: wherein said client workstation executes said browser application, with the browser application responding to text formats to initiate processing specified by the text formats,	MediaView discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c. MediaView discloses parsing text formats. *See* evidence recited for 985-1.d.
985-28.g: wherein at least said portion of the document is	MediaView discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.

Claim Text from ’985 Patent	MediaView
displayed within the browser-controlled window,	
985-28.h: wherein an embed text format which corresponds to said first location in the document is identified by the browser,	MediaView discloses identifying an embed text format and that the embed text format corresponds to a first location in a hypermedia document. *See* evidence recited for 985-1.f.
985-28.i: wherein the embed text format specifies the location of at least a portion of said object external to the file,	MediaView discloses that the embed text format specifies the location of at least a portion of an object external to the file containing enabling information. *See* evidence recited for 985-1.f.
985-28.j: wherein the object has type information associated with it,	MediaView discloses that the object has associated type information. *See* evidence recited for 985-1.f.
985-28.k: wherein the type information is utilized by the browser to identify and locate said executable application, and	MediaView discloses using type information to identify and locate an executable application external to the file. *See* evidence recited for 985-1.g.
985-28.l: wherein the executable application is automatically invoked by the browser, in response to the identifying of the embed text format.	MediaView discloses automatically invoking the executable application. *See* evidence recited for 985-1.h.
985-36.a: A method for running an application program in a distributed hypermedia network environment, wherein the distributed hypermedia network environment comprises at least one client workstation and one remote network server coupled to the distributed hypermedia network environment, the method comprising:	MediaView discloses an application program in a distributed hypermedia environment comprising at least client workstation and network server. *See* evidence recited for 985-1.a.
985-36.b: receiving, at the client workstation from the network server over the distributed hypermedia	MediaView discloses a browser application; a file containing enabling information; that the file is received at the client workstation from the network server; that the browser displays at least a portion of a distributed hypermedia

Claim Text from '985 Patent	MediaView
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;	document; and that at least a portion of a hypermedia document is displayed in a browser-controlled window. *See* evidence recited for 985-1.b.
985-36.c: executing the browser application on the client workstation, with the browser application:	MediaView discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c.
985-36.d: responding to text formats to initiate processing specified by the text formats;	MediaView discloses parsing text formats. *See* evidence recited for 985-1.d.
985-36.e: displaying at least a portion of the document within the browser-controlled window;	MediaView discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.
985-36.f: identifying an embed text format which corresponds to a first location in the document, where the embed text format specifies the location of at least a portion of an object;	MediaView discloses an object. If the parsed ViewCell character represents an embedded RenderMan viewer (an object), its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance will be rendered by a RenderMan server, an external application. MediaView discloses identifying an embed text format; that the embed text format corresponds to a first location in the hypermedia document; and that the embed text format specifies the location of an object. *See* evidence recited for 985-1.f.
985-36.g: identifying and locating an executable application associated with the object; and	MediaView discloses that the browser identifies and locates an executable application associated with the object. If the parsed ViewCell character represents an embedded RenderMan viewer, its type is an instance of the _3DButton class and its function is performed by the _3DAction method. Moreover, the _3DButton instance

Claim Text from '985 Patent	**MediaView**
	will be rendered by a RenderMan server, an executable application.
985-36.h: 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,	MediaView discloses identifying an embed text format. *See* evidence recited in 985-1.f. MediaView discloses automatic invocation of the executable application; that the executable application displays the object; that the executable application enables direct interaction with the object; and that interaction with the object is at a first location in the hypermedia document. *See* evidence recited in 985-1.h. MediaView discloses that the object is displayed at a first location within a portion of the hypermedia document being displayed. *See* evidence recited at 985-24.b. MediaView discloses that a hypermedia document is displayed in a browser window. *See, e.g.*, evidence recited for 985-1.e.
985-36.i: wherein the executable application is part of a distributed application, and	MediaView discloses a distributed application. The applications depicted by Figures 6 and 7 of my report are distributed applications where the work done prior to dispatching a task to a server, is done on a client workstation. MediaView discloses that the executable application is part of a distributed application.
985-36.j: wherein at least a portion of the distributed application is for execution on a remote network server coupled to the distributed hypermedia network environment.	MediaView discloses that the distributed application executes at least partially on a network server. The applications depicted by Figures 6 and 7 of my report are distributed applications where the work done prior to dispatching a task to a server, is done on a client workstation. The task thus dispatched is performed on the

Claim Text from ’985 Patent	**MediaView**
	appropriate server.
985-37.a: The method of claim 36 where: the information to enable comprises text formats.	MediaView discloses that the enabling information in the file is text formats. *See* evidence recited for 985-2.a.
985-38.a: The method of claim 37 where: the text formats are HTML tags.	This limitation is obvious for the same reasons as discussed in the evidence for 985-3.a.
985-39.a: The method of claim 36 where: the information contained in the file received comprises at least one embed text format.	MediaView discloses that the enabling information in the file includes an embed text format. *See* evidence recited for 985-4.a.
985-40.a: A method of serving digital information in a computer network environment having a network server coupled to said computer network environment, and where the network environment is a distributed hypermedia network environment, the method comprising:	MediaView discloses digital information. *See* evidence recited for 985-20.a. MediaView discloses a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.
985-40.b: communicating via the network server with at least one remote client workstation over said computer network environment in order to cause said client workstation to:	MediaView discloses a client workstation. *See* evidence recited for 985-1.a. MediaView discloses communicating via network server in order to cause the client workstation to act. *See* evidence recited for 985-20.b.
985-40.c: 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	MediaView discloses a browser application; a file containing enabling information received from a server; that the browser displays at least a portion of a distributed hypermedia document; and that the display is in a browser-controlled window. *See* evidence recited for 985-1.b.

Claim Text from ’985 Patent	MediaView
distributed hypermedia document within a browser-controlled window;	
985-40.d: execute, at said client workstation, a browser application, with the browser application:	MediaView discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c.
985-40.e: responding to text formats to initiate processing specified by the text formats;	MediaView discloses parsing text formats. *See* evidence recited for 985-1.d.
985-40.f: displaying, on said client workstation, at least a portion of the document within the browser-controlled window;	MediaView discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.
985-40.g: identifying an embed text format which corresponds to a first location in the document, where the embed text format specifies the location of at least a portion of an object;	MediaView discloses an object. *See* evidence recited for 985-36.f. MediaView discloses identifying an embed text format; that the embed text format corresponds to a first location in the hypermedia document; and that the embed text format specifies the location of an object. *See* evidence recited for 985-1.f.
985-40.h: identifying and locating an executable application associated with the object; and	MediaView discloses that the browser identifies and locates an executable application associated with the object. *See* evidence recited for 985-36.g.
985-40.i: 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,	MediaView discloses identifying an embed text format. *See* evidence recited in 985-1.f. MediaView discloses automatic invocation of the executable application; that the executable application displays the object; that the executable application enables direct interaction with the object; and that interaction with the object is at a first location in the hypermedia document. *See* evidence recited in 985-1.h. MediaView discloses that the object is displayed at a first location within a portion of the hypermedia document being displayed. *See* evidence recited for 985-24.b.

Claim Text from ’985 Patent	**MediaView**
	MediaView discloses that a hypermedia document is displayed in a browser window. *See, e.g.*, evidence recited for 985-1.e.
985-40.j: wherein the executable application is part of a distributed application, and	MediaView discloses that the executable application is part of a distributed application. *See* evidence recited in 985-36.i.
985-40.k: wherein at least a portion of the distributed application is for execution on the network server.	MediaView discloses that the distributed application executes at least partially on a network server. *See* evidence recited for 985-36.j.
985-41.a: The method of claim 40 where: the information to enable comprises text formats.	MediaView discloses that the enabling information in the file is text formats. *See* evidence recited for 985-2.a.
985-42.a: The method of claim 41 where: the text formats are HTML tags.	This limitation is obvious for the same reasons as discussed in the evidence for 985-3.a.
985-43.a: The method of claim 40 where: the information contained in the file received comprises at least one embed text format.	MediaView discloses that the enabling information in the file includes an embed text format. *See* evidence recited for 985-4.a.