CLAIM CHART EXHIBIT 13
"HYPERCARD AND DIRECTOR"

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

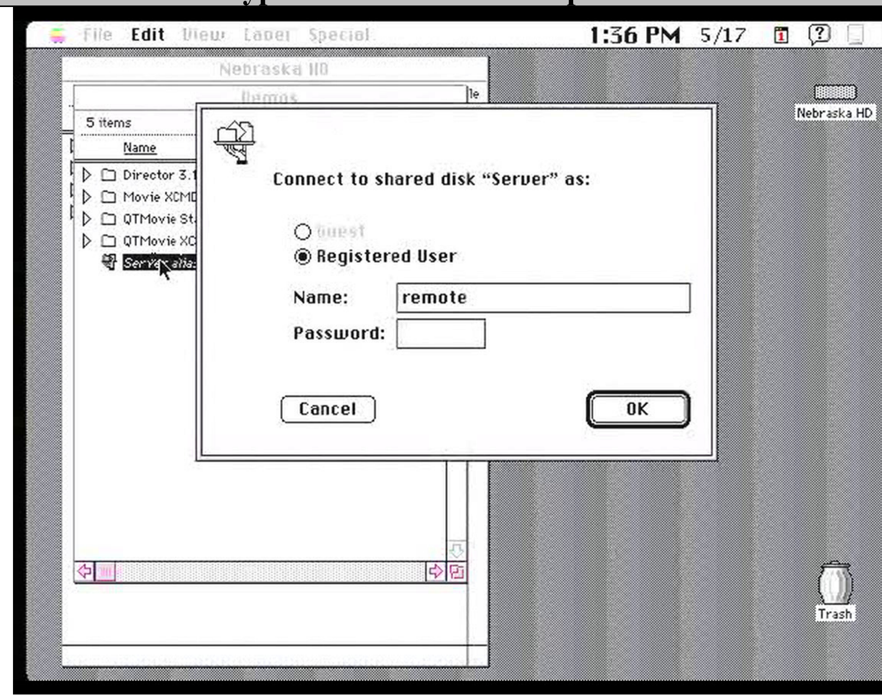
- **"HYPERCARD AND DIRECTOR"[1] -- DIRECTOR SOFTWARE [DIRECTOR], INCLUDING MACROMIND PLAYER MANUAL DISTRIBUTED WITH DIRECTOR 3.1.3 ("DIRECTOR PRIOR ART") AS INTENED TO BE USED IN A COMPUTER SYSTEM AND DEMONSTRATION OF SAME, FURTHER INFORMED BY:**
  - **DANNY GOODMAN. THE COMPLETE HYPERCARD 2.0 HANDBOOK. 3RD EDITION. BANTAM BOOKS, INC., AUGUST 1990. ("GOODMAN") [PA-00288603] [GOODMAN90];**
  - **HYPERCARD VERSIONS 2.0, 2.1, AND 2.2 ("HYPERCARD PRIOR ART") [HYPERCARD];**
  - **ERIC LEASE MORGAN. "IMPLEMENTING TCP/IP COMMUNICATIONS WITH HYPERCARD." INFORMATION TECHNOLOGY AND LIBRARIES, DEC. 1992; 11, 4; ABI/INFORM GLOBAL. PP. 421-432;**
  - **JOHN R. POWERS, III. "MAC TO MAINFRAME WITH HYPERCARD." MACTUTOR, JUNE 1990. [PA-00288589] [POWERS90]; AND**
  - **DECLARATION OF DANIEL SADOWSKI, JUNE 2011 [SADOWSKI11].**
  - **THE BODY OF MY REPORT HAS A NARRATIVE DESCRIPTION THAT AUGMENTS AND SHOULD BE CONSIDERED PART OF THIS CHART.**

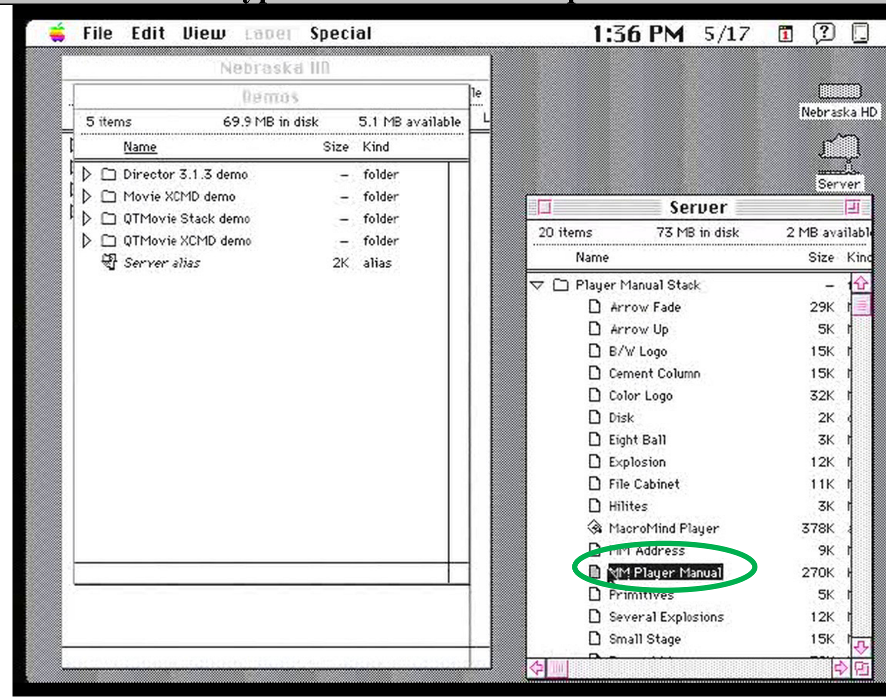
Claim Text from '906 Patent	HyperCard and Director prior art
906-1.a: A method for running an application program in a computer network environment, comprising:	HyperCard and Director discloses an application program. *See, e.g.*, : HyperCard is a computer program. It was installed onto a computer, such as an Apple Macintosh computer, and launched as an executable program. The videos that I am submitting with this report show how this was done. (See also Goodman at pp. 17-20) (describing installation and operation of HyperCard on a computer); (Goodman at pp. xxxiii) (describing that HyperCard is a computer program).

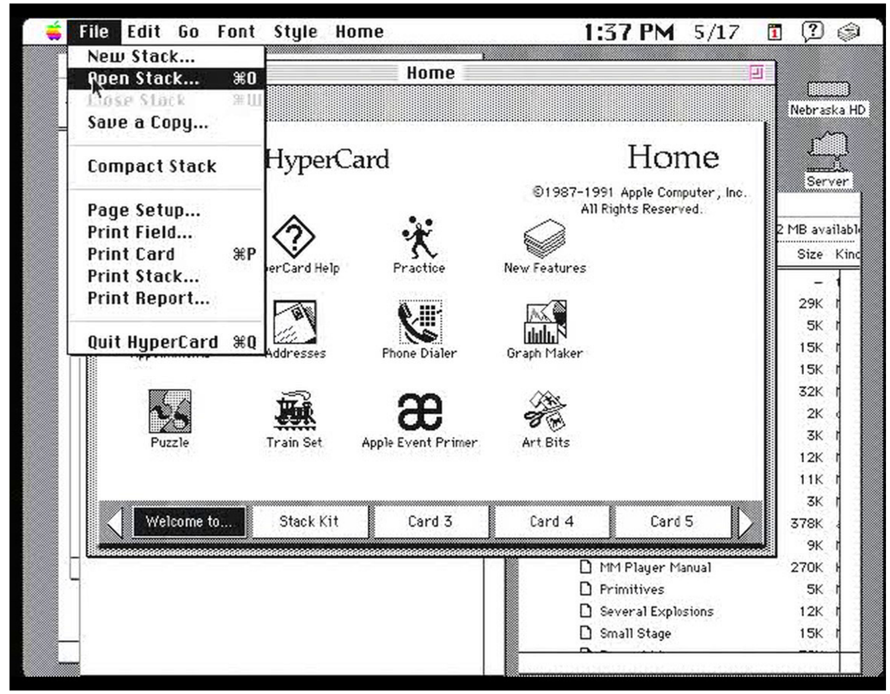
[1] For all asserted claims this reference is a 103 reference due to my understanding of the plain meaning of the limitations relating to "location" (e.g. 901-1.f and 906-1.g and 985-1.f and 985.1g) and the Court's discussion of the issue on page 17 of its August 22, 2011 Order. Thus, for these particular limitations, the reference is not anticipatory, but rather, as explained in the body of my report, this limitation would be combined with a prior art web browser like Mosaic, CERN's web browser, Viola, or MediaView. Likewise, to satisfy the HTML limitations in the '985 patent, the reference must be combined with a web browser or HTML teaching, such as Mosaic, CERN's web browser, or Viola. For both all such limitations it would have been obvious to a person of ordinary skill in the art at the time to do so as explained in the body of my report and the teachings, for example, of Tim Berners-Lee posted on the CERN website discussing the Web and relating features and pointers to other browser technologies including HyperCard, Viola and MediaView. This was an obvious and natural extension of prior hypermedia functions and features and an inevitable development in the marketplace at the time of the invention and based on the state of the art.

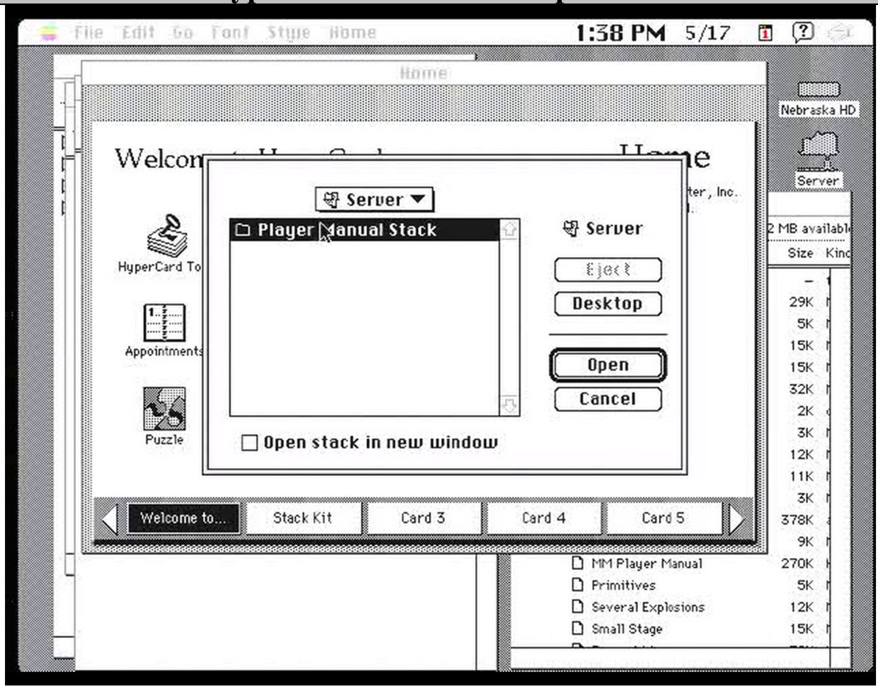
Claim Text from '906 Patent	HyperCard and Director prior art
	HyperCard and Director discloses a computer network environment. *See, e.g.*, : HyperCard operating in a distributed hypermedia environment that included clients and servers. Specifically, stacks can be stored on or published to a file server and then accessed by a user using HyperCard on a client workstation. (See Goodman at pp. 737-739.) "On networks such as AppleShare and TOPS, HyperCard sees file servers or other user's published volumes just as another disk drive attached to the Macintosh. If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack's pathname cared for stacks…" (See Goodman at p. 737.) "If you develop what we call information publishing stacks – those that come full of information for users to browse through – you should be aware that such a stack might be used on a network." (See Goodman at p. 739.) Stacks shared over a network "should be on the file server or, in the case of a TOPS network, on a published volume." (See Goodman at p. 738; see also [Sadowski11] at 54, 55, 60, 74, 76.) A network of computers containing HyperCard stacks is a distributed hypermedia environment because HyperCard stacks contain hypermedia. (See, e.g., Goodman at Chapter 52) (describing text, sound, animation, and movies.) ("If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack's pathname card for stacks, just as it would be if it were on your own hard disk"). As another example, HyperCard could interoperate with the TCP/IP Internet. A software package called MacTCP and a set of XCMDs provided with the HyperCard TCP Toolkit provided this functionality. (See Morgan at 421.)
906-1.b: providing at least one client workstation and one	HyperCard and Director discloses a client workstation. *See, e.g.*, :

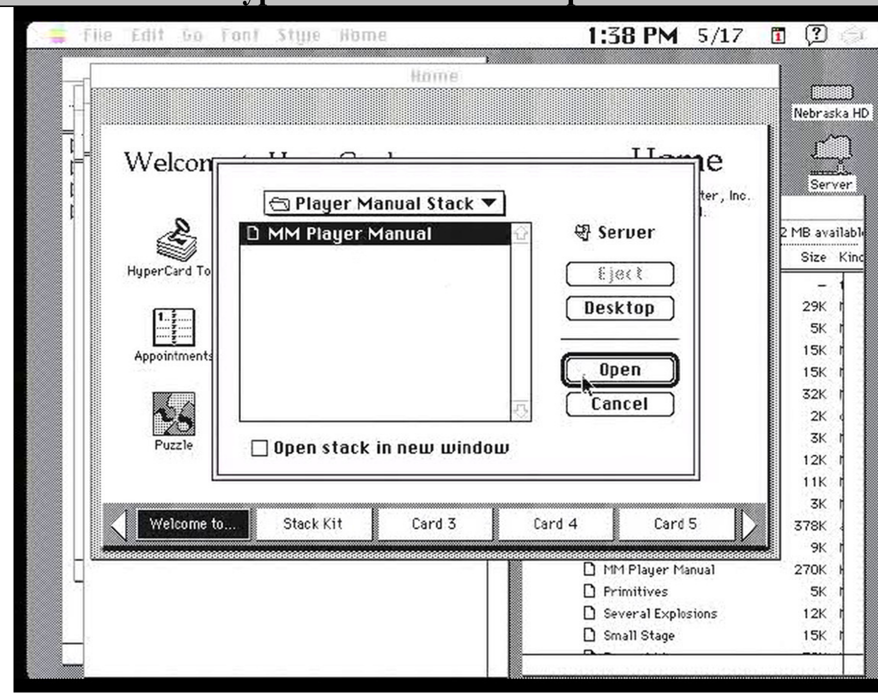
Claim Text from ’906 Patent	HyperCard and Director prior art
network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment;	HyperCard works in a distributed environment that includes clients and servers. Specifically, stacks can be stored on or published to a file server and then accessed by a user using HyperCard on a client workstation. (See Goodman at pp. 737-739; see also [Sadowski11] at 54, 55, 60, 74, 76.) “On networks such as AppleShare and TOPS, HyperCard sees file servers or other user’s published volumes just as another disk drive attached to the Macintosh. If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname cared for stacks…” (See Goodman at p. 737.) “If you develop what we call information publishing stacks – those that come full of information for users to browse through – you should be aware that such a stack might be used on a network.” (See Goodman at p. 739.) Stacks shared over a network “should be on the file server or, in the case of a TOPS network, on a published volume.” (See Goodman at p. 738.) One mechanism by which stacks on a server were accessed from a client was through buttons. (See Goodman at p. 737) ("If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname card for stacks, just as it would be if it were on your own hard disk") Examples of HyperCard’s client-server functionality are shown in the videos I am submitting with this report. In addition, screenshots from my video depicting the client-server functionality are shown below. For example, the screenshot below shows a client-server arrangement. This first screenshot shows the connection to the server.

Claim Text from ’906 Patent	HyperCard and Director prior art
	 The screenshot below shows the MM Player Manual stack file being stored remotely on a server. The MM Player Manual stack is highlighted in green.

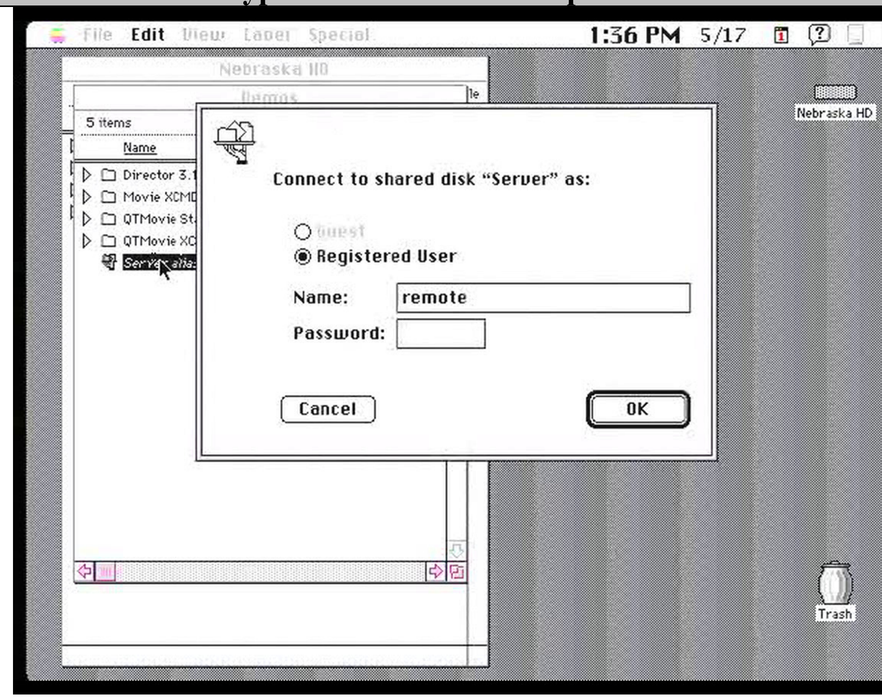
Claim Text from ’906 Patent	HyperCard and Director prior art
	 The screenshots below show the sequence of steps taken to open the MM Player Manual stack which is stored remotely on a server. The steps are taken on a client workstation.

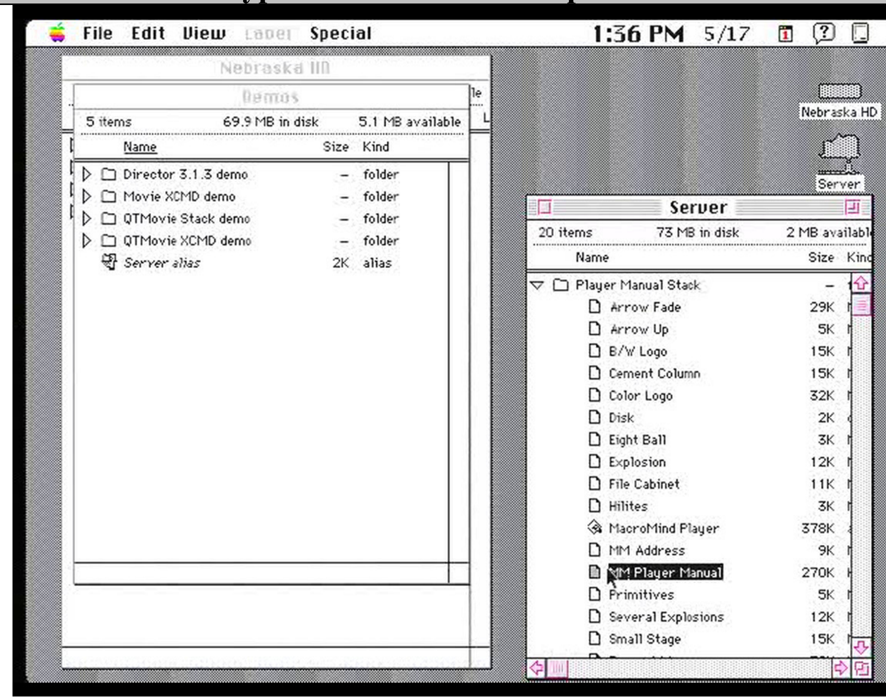
Claim Text from ’906 Patent	HyperCard and Director prior art
	

Claim Text from '906 Patent	HyperCard and Director prior art
	

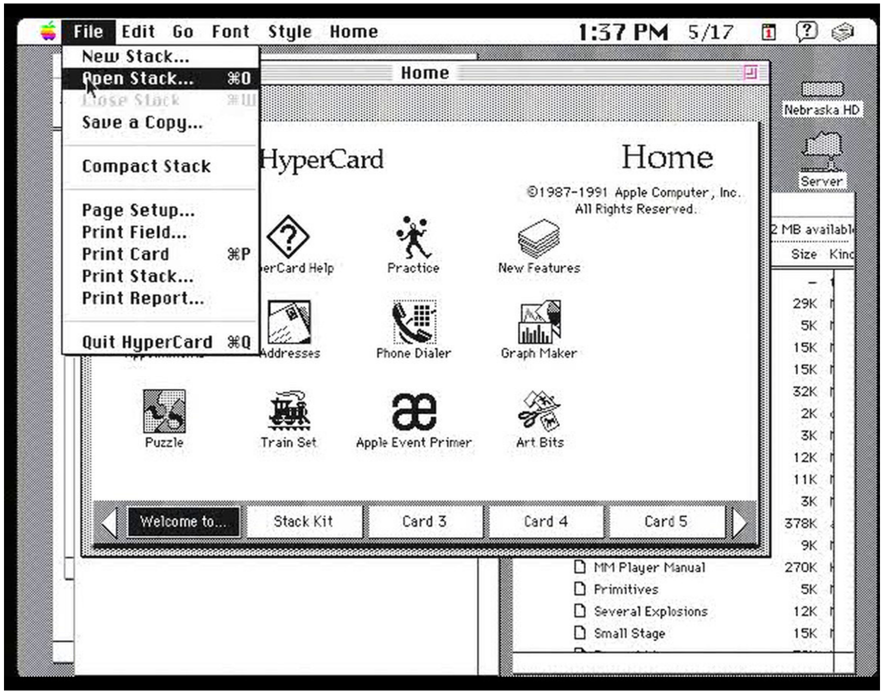
Claim Text from ’906 Patent	HyperCard and Director prior art
	 HyperCard and Director discloses a network server. *See, e.g.*, : HyperCard works in a distributed environment that includes clients and servers. Specifically, stacks can be stored on or published to a file server and then accessed by HyperCard on a client workstation. (See Goodman at pp. 737-739.) “On networks such as AppleShare and TOPS, HyperCard sees file servers or other user’s published volumes just as another disk drive attached to the Macintosh. If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname cared for stacks…” (See Goodman at p. 737.) “If you develop what we call information

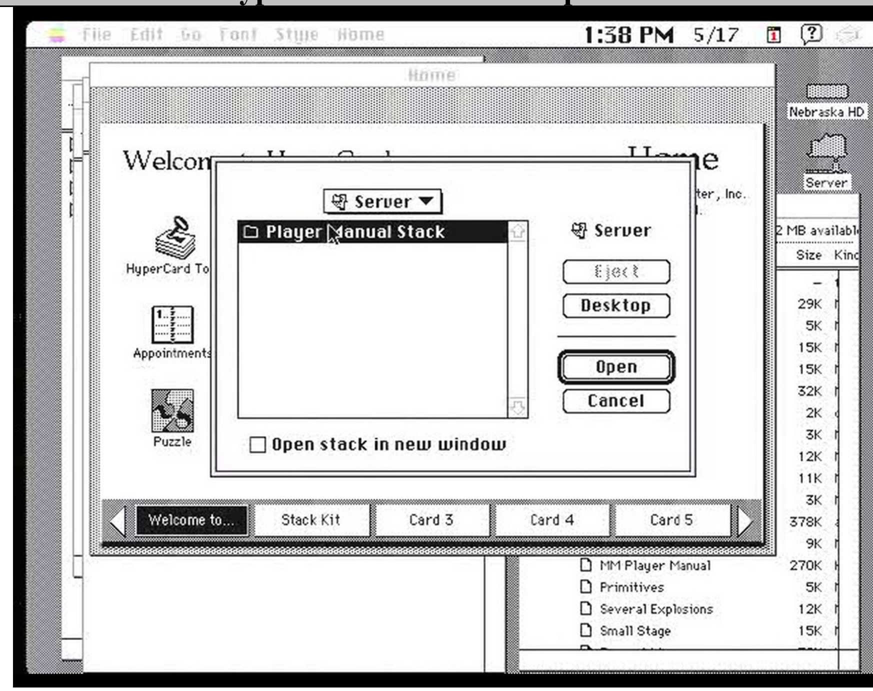
Claim Text from ’906 Patent	HyperCard and Director prior art
	publishing stacks – those that come full of information for users to browse through – you should be aware that such a stack might be used on a network.” (See Goodman at p. 739.) Stacks shared over a network “should be on the file server or, in the case of a TOPS network, on a published volume.” (See Goodman at p. 738; see also [Sadowski11] at 54, 55, 60, 74, 76.) One mechanism by which stacks on a server were accessed from a client was through buttons. (See Goodman at p. 737) ("If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname card for stacks, just as it would be if it were on your own hard disk"). Examples of HyperCard's client-server functionality are shown in the videos I am submitting with this report. In addition, screenshots from my video showing the client-server functionality are shown below. The screenshot below shows a client-server arrangement.

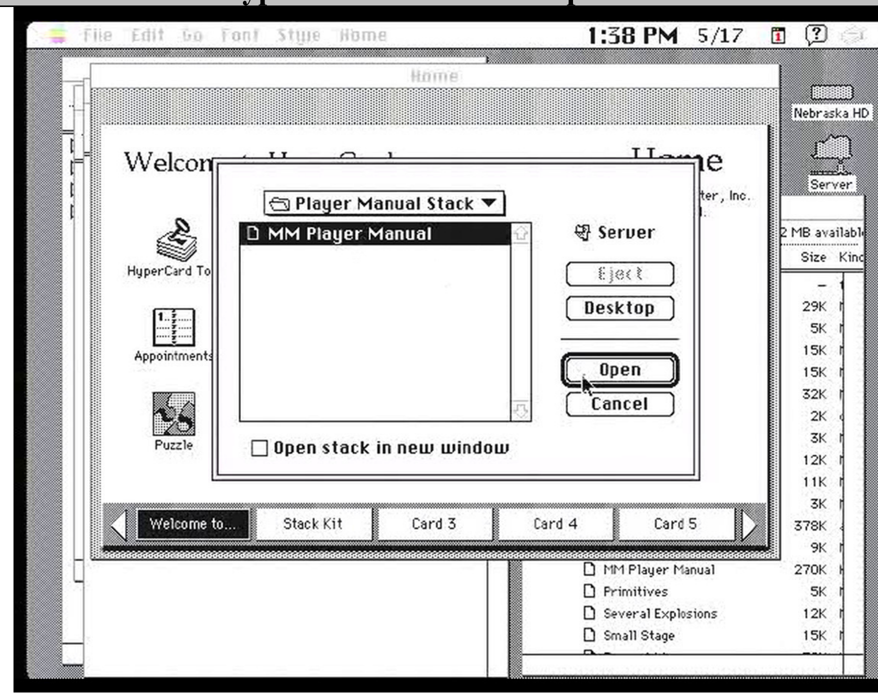
Claim Text from ’906 Patent	HyperCard and Director prior art
	 The screenshot below shows the MM Player Manual stack file being stored remotely on a server.

<table>
<tr><th>Claim Text from ’906 Patent</th><th>HyperCard and Director prior art</th></tr>
<tr><td></td><td>


The screenshots below show the sequence of steps taken to open the MM Player Manual stack file which is stored remotely on a server.
</td></tr>
</table>

Claim Text from ’906 Patent	HyperCard and Director prior art
	

Claim Text from ’906 Patent	HyperCard and Director prior art
	

<table>
<tr><th>Claim Text from ’906 Patent</th><th>HyperCard and Director prior art</th></tr>
<tr><td></td><td>


HyperCard and Director discloses a distributed hypermedia environment. *See, e.g.*, :

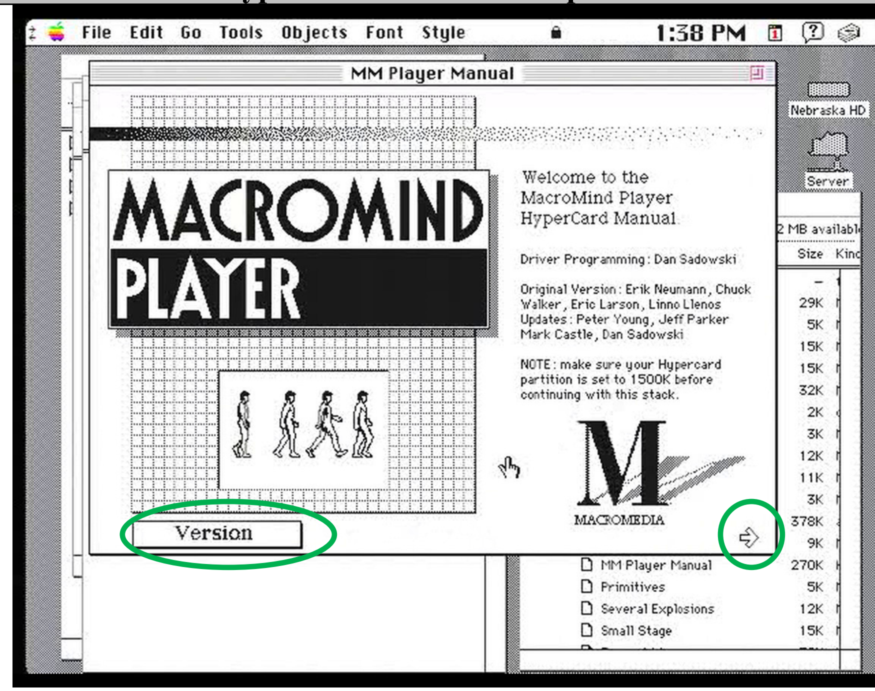
HyperCard operating in a distributed hypermedia environment that included clients and servers.

Specifically, stacks can be stored on or published to a file server and then accessed by a user using HyperCard on a client workstation. (See Goodman at pp. 737-739.) “On networks such as AppleShare and TOPS, HyperCard sees file servers or other user’s published volumes just as another disk drive attached to the Macintosh. If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s
</td></tr>
</table>

Claim Text from '906 Patent	HyperCard and Director prior art
	pathname cared for stacks…" (See Goodman at p. 737.) "If you develop what we call information publishing stacks – those that come full of information for users to browse through – you should be aware that such a stack might be used on a network." (See Goodman at p. 739.) Stacks shared over a network "should be on the file server or, in the case of a TOPS network, on a published volume." (See Goodman at p. 738; see also [Sadowski11] at 54, 55, 60, 74, 76.) A network of computers containing HyperCard stacks is a distributed hypermedia environment because HyperCard stacks contain hypermedia. (See, e.g., Goodman at Chapter 52) (describing text, sound, animation, and movies.) ("If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack's pathname card for stacks, just as it would be if it were on your own hard disk"). As another example, HyperCard could interoperate with the TCP/IP Internet. A software package called MacTCP and a set of XCMDs provided with the HyperCard TCP Toolkit provided this functionality. (See Morgan at 421.)
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;	HyperCard and Director discloses a browser application. *See, e.g.*, : HyperCard is a browser application because it displays hypermedia documents and allows a user to browse through different parts of the hypermedia documents and to other hypermedia documents using, for example, buttons and links between cards and stacks. As one example, HyperCard provided buttons, which were a navigational tool used to browse through a HyperCard stack. (See Goodman at p. 35); (Goodman at Chapter 11.) The functionality of buttons could be specified using the HyperTalk

Claim Text from ’906 Patent	HyperCard and Director prior art
	language. At the core of most HyperCard button activity is the link, which ties together one card with another card. That other card can be the next card in the stack; a previously viewed card in the same or a different stack; the first card in another stack; or a specific card in another stack. (See Goodman at Chapter 12.) HyperCard also provided for linked text. (See Goodman at 72.) Buttons could link to cards or stacks on the same computer, or on a server computer. (See Goodman at p. 737.) Examples of all this functionality are shown in the videos I am submitting with this report. Below is a screenshot from my video of the HyperCard application. By way of clarification, I have also highlighted navigational buttons in green from the HyperCard application.

Claim Text from ’906 Patent	HyperCard and Director prior art



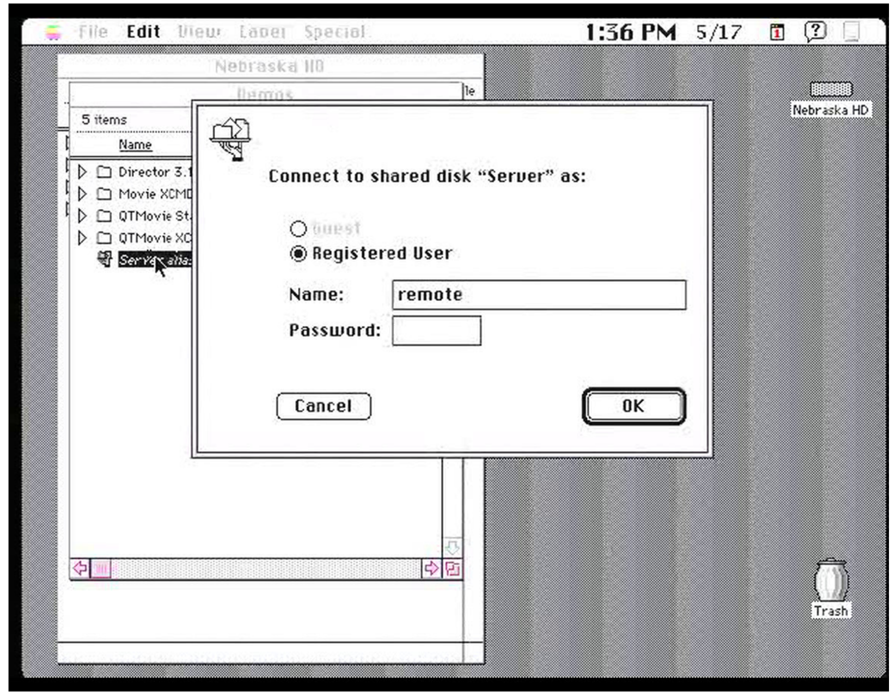
HyperCard and Director discloses that the browser application parses a hypermedia document. *See, e.g.*, :

> HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file.
>
> Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When

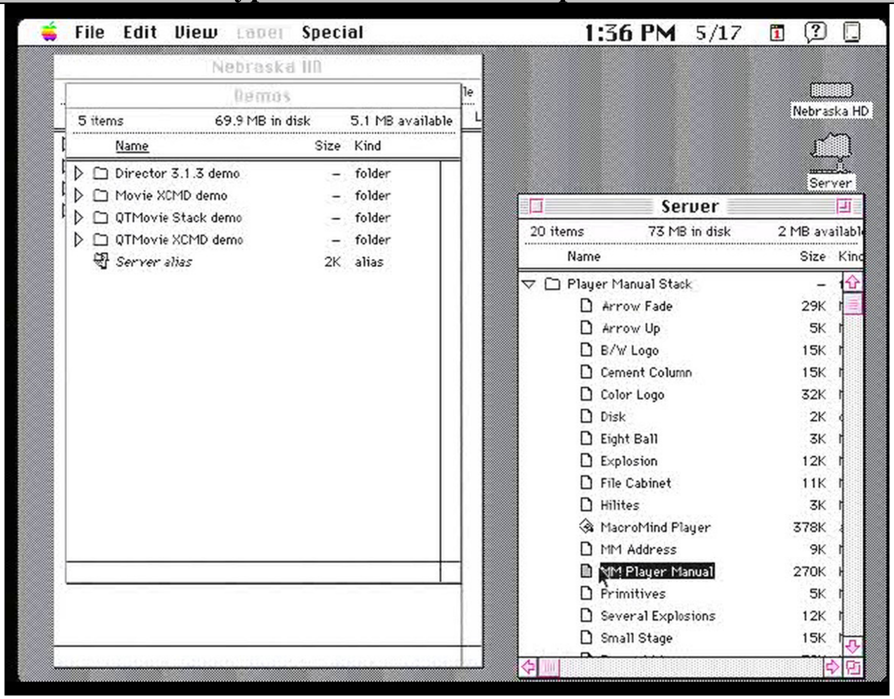
Claim Text from '906 Patent	HyperCard and Director prior art
	presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." The videos that I am submitting with this report show examples of HyperTalk scripts associated with hypermedia cards. HyperCard and Director discloses a hypermedia document with text formats. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." Several types of text objects were stored in the HyperCard hypermedia document. The videos that I am submitting with this report show examples of HyperTalk scripts associated with hypermedia cards.
906-1.d:	HyperCard and Director discloses that a hypermedia document is received from

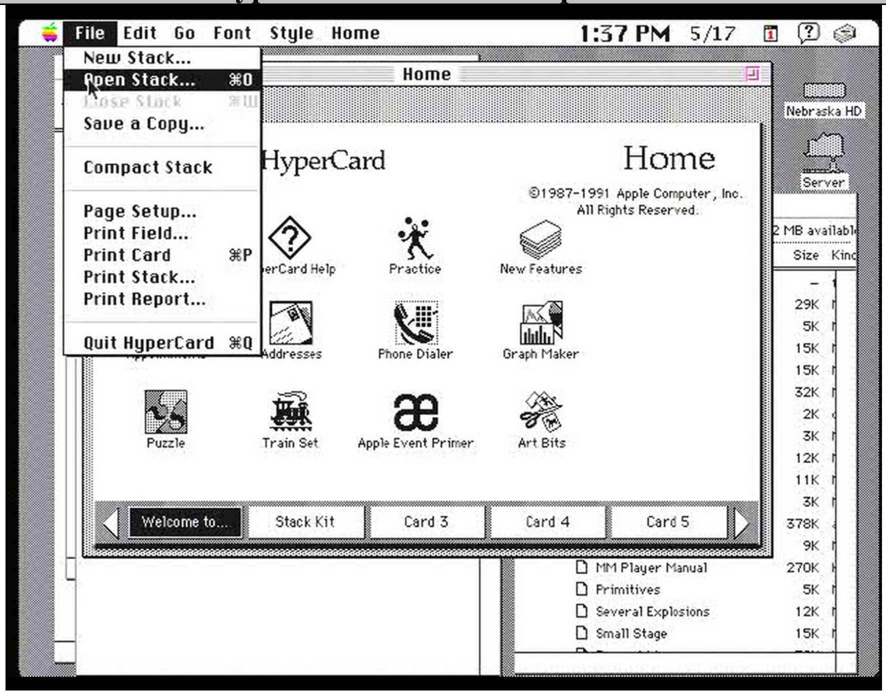
Claim Text from ’906 Patent	HyperCard and Director prior art
utilizing said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server,	the server. *See, e.g.*, : HyperCard works in a distributed environment that includes clients and servers. In such environments, HyperCard operating on a client computer receives hypermedia cards from a server computer. Specifically, stacks can be stored on or published to a file server and then accessed by HyperCard on a client workstation. (See Goodman at pp. 737-739.) “On networks such as AppleShare and TOPS, HyperCard sees file servers or other user’s published volumes just as another disk drive attached to the Macintosh. If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname cared for stacks…” (See Goodman at p. 737.) “If you develop what we call information publishing stacks – those that come full of information for users to browse through – you should be aware that such a stack might be used on a network.” (See Goodman at p. 739.) Stacks shared over a network “should be on the file server or, in the case of a TOPS network, on a published volume.” (See Goodman at p. 738; see also [Sadowski11] at 54, 55, 60, 74, 76.) HyperCard operating on a client received hypermedia cards from a server. As one example, a button could be configured to retrieve hypermedia cards from a server. (See Goodman at p. 737.) ("If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname card for stacks, just as it would be if it were on your own hard disk") (See also [Sadowski11] at 54, 55, 60, 74, 76.) Examples of HyperCard's client-server functionality are shown in the videos I am submitting with this expert report. In addition, screenshots from my video depicting the client-server functionality are shown below.

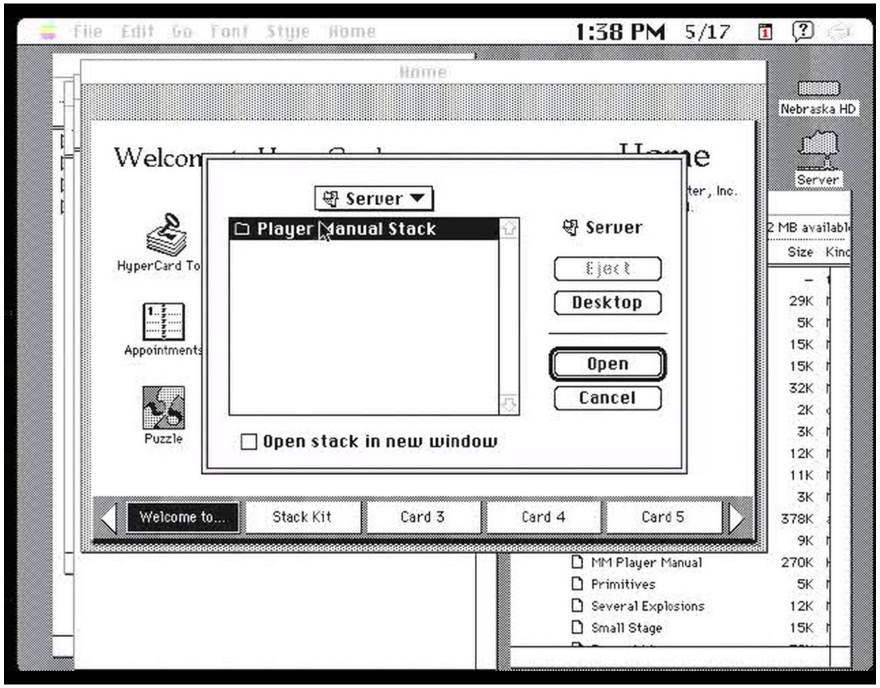
<table>
<tr><th>Claim Text from ’906 Patent</th><th>HyperCard and Director prior art</th></tr>
<tr><td></td><td>
The screenshot below shows a client-server arrangement.

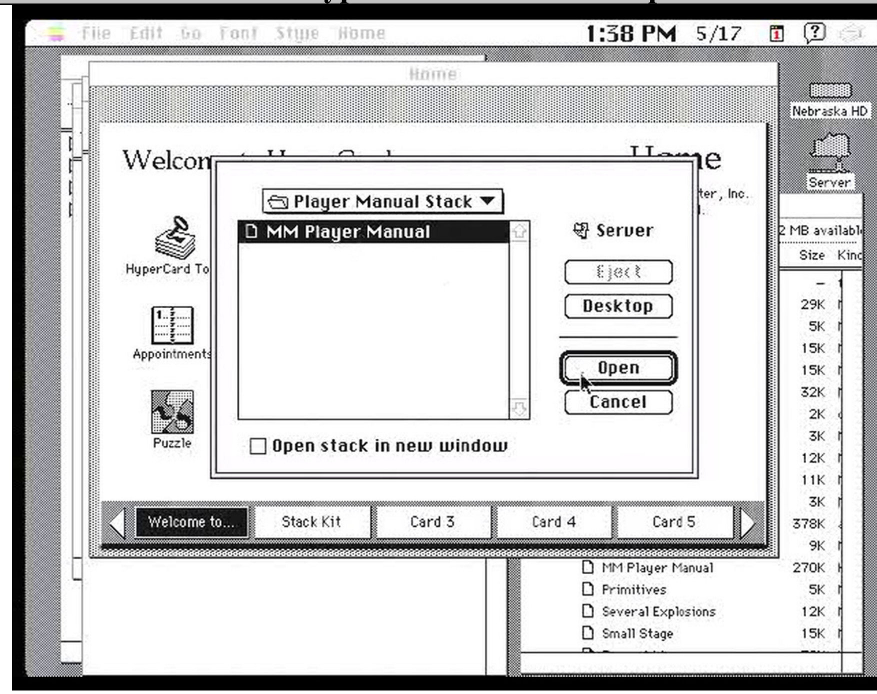


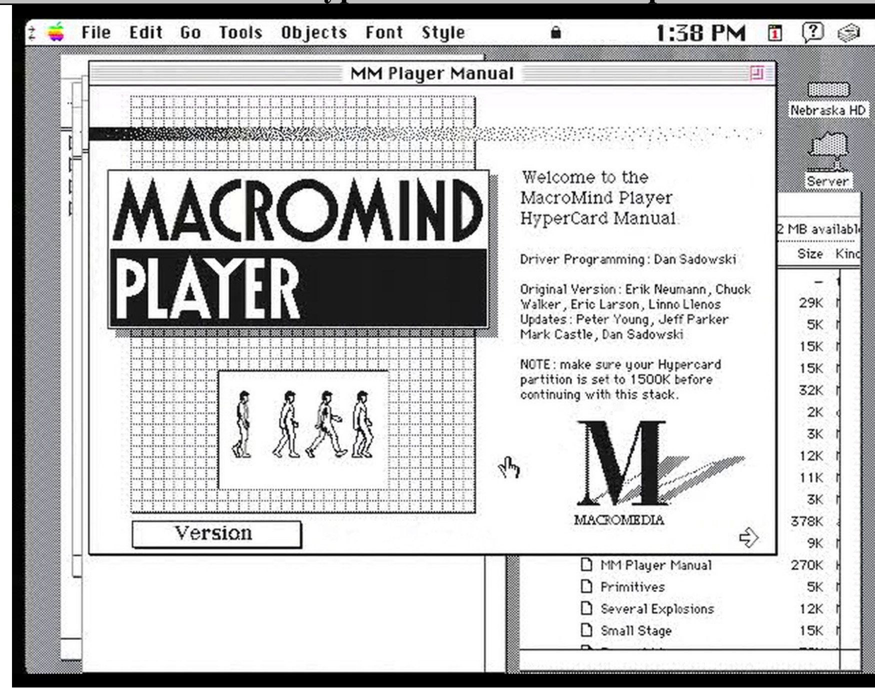
The screenshot below shows the MM Player Manual stack file being stored remotely on a server.
</td></tr>
</table>

Claim Text from ’906 Patent	HyperCard and Director prior art
	 The screenshots below show the sequence of steps taken to open the MM Player Manual stack file which is located remotely on a server.

Claim Text from '906 Patent	HyperCard and Director prior art
	

Claim Text from '906 Patent	HyperCard and Director prior art
	

Claim Text from '906 Patent	HyperCard and Director prior art
	 The screenshot below shows the opened MM Player Manual stack which is received from a server.

Claim Text from ’906 Patent	HyperCard and Director prior art
	 HyperCard and Director discloses that the browser displays a hypermedia document. *See, e.g.*, : HyperCard displays cards that are part of stacks. Cards are hypermedia documents because they can include a variety of media types, including text, sound, animation, and movies. (See, e.g., Goodman at Chapter 52) (describing text, sound, animation, and movies in cards.) The videos that I am submitting with this report show examples of text, animations, movies, and interactive movies.
906-1.e: wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation,	HyperCard and Director discloses that a hypermedia document is displayed in a browser window. *See, e.g.*, : HyperCard displays hypermedia cards in a HyperCard window. By way

<table>
<tr><th>Claim Text from ’906 Patent</th><th>HyperCard and Director prior art</th></tr>
<tr><td></td><td>of example, hypermedia cards displayed in the HyperCard window are shown in Goodman at pp. 791-836. In addition, the videos I am submitting with this report show hypermedia cards displayed within a HyperCard window.</td></tr>
<tr><td>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,</td><td>HyperCard and Director discloses an embed text format at a first location in a hypermedia document. See, e.g., :

HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.”
A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be:
on presentMovie
 global returnSound
 playMovie “Cement Column”
 lock screen
 play returnSound
 pop card
 unlock screen with wipe right
end presentMovie</td></tr>
</table>

Claim Text from ’906 Patent	HyperCard and Director prior art
	HyperCard and Director discloses that the embed text format specifies the location of an object. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name “Cement Column,” stored elsewhere as a separate file. (See e.g., [Sadowski11] at 72-74]).

Claim Text from ’906 Patent	HyperCard and Director prior art
	HyperCard and Director discloses an object that is external to a hypermedia document. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a with the name “Cement Column,” which is specified directly and is stored elsewhere as a separate file and is thus external to the hypermedia (HyperCard) document. (See e.g., [Sadowski11] at 72-74]).

Claim Text from ’906 Patent	HyperCard and Director prior art
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	HyperCard and Director discloses that the object has associated type information. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name “Cement Column,” which is specified directly and is stored elsewhere as a separate file. In this case, the syntax "playMovie" is type information specifying that the object “Cement Column” is a Director XCMD for a Director movie object. (See

Claim Text from '906 Patent	HyperCard and Director prior art
	e.g., [Sadowski11] at 72-74]). HyperCard and Director discloses that the browser uses type information to identify and locate an executable application. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie "Cement Column" lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name "Cement Column," which is specified directly and is stored elsewhere as a separate file. In this case,

Claim Text from ’906 Patent	HyperCard and Director prior art
	the syntax "playMovie" is type information specifying that the object “Cement Column” is a playMovie XCMD for a Director movie object. That type information is found in the embed text format. (See e.g., [Sadowski11] at 72-74]). The thus-located XCMD is a compiled executable application. HyperCard and Director discloses that the executable application is external to the hypermedia document. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card

Claim Text from '906 Patent	HyperCard and Director prior art
	unlock screen with wipe right end presentMovie The object in this script is a file with the name "Cement Column," which is specified directly and is stored elsewhere as a separate file. In this case, the syntax "playMovie" is type information specifying that the object "Cement Column" is a playMovie XCMD for a Director movie object. That type information is found in the embed text format. (See e.g., [Sadowski11] at 72-74]). The thus-located XCMD is a compiled executable application which can be stored anywhere on a computer disk.
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 created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.	HyperCard and Director discloses that the browser parses the embed text format. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie "Cement Column"

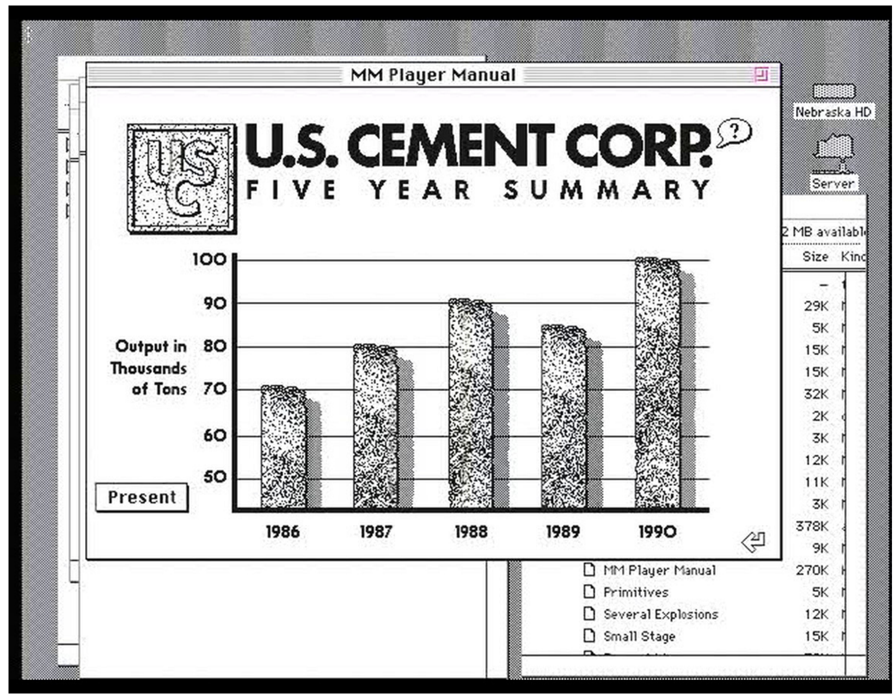
Claim Text from '906 Patent	HyperCard and Director prior art
	lock screen play returnSound pop card unlock screen with wipe right end presentMovie The browser (HyperCard) parses the embed text format to cause the movie "Cement Column" to be played on the relevant card. (See e.g., [Sadowski11] at 72-74]). HyperCard and Director discloses automatic invocation of the executable application. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie "Cement Column"

Claim Text from ’906 Patent	HyperCard and Director prior art
	lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name Cement Column, which is specified directly and is stored elsewhere as a separate file. In this case, the syntax "playMovie" is type information specifying that the object “Cement Column” is a playMovie XCMD for a Director movie object. That type information is found in the embed text format. (See e.g., [Sadowski11] at 72-74]). The thus-located XCMD is a compiled executable application which can be stored anywhere on a computer disk. The executable application, XCMD, displays the object, “Cement Column.” HyperCard provided for automatic invocation of executable applications, without interactive action by the user, such as in the on presentMovie - end presentMovie script above. HyperCard and Director discloses that the executable application displays the object. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise

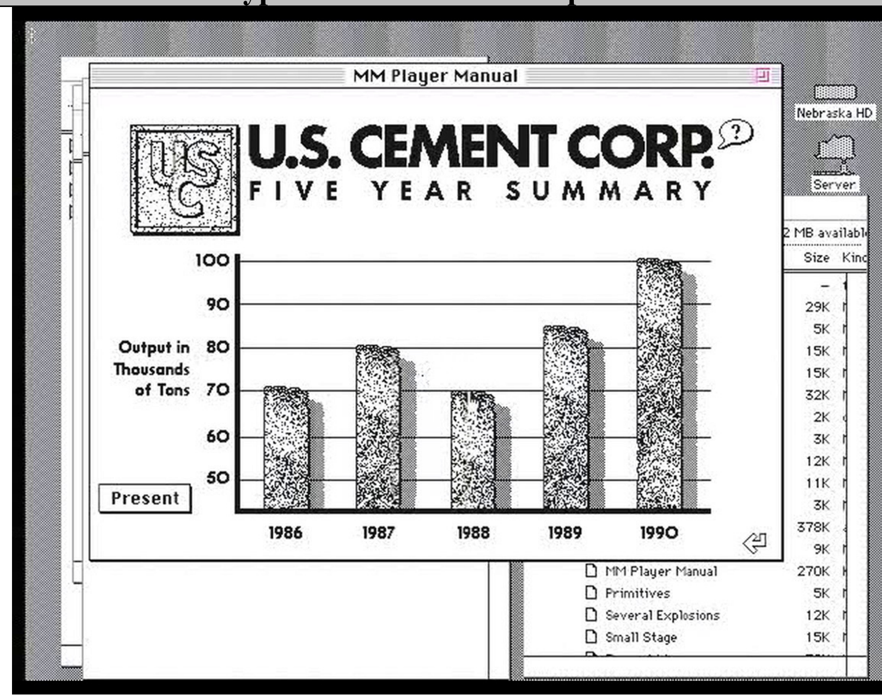
Claim Text from ’906 Patent	HyperCard and Director prior art
	coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name “Cement Column,” which is specified directly and is stored elsewhere as a separate file. In this case, the syntax "playMovie" is type information specifying that the object “Cement Column” is a playMovie XCMD for a Director movie object. That type information is found in the embed text format. (See e.g., [Sadowski11] at 72-74]). The thus-located XCMD is a compiled executable application which can be stored anywhere on a computer disk. The executable application, XCMD, displays the object, “Cement Column.” HyperCard and Director discloses that the executable application enables direct interaction with the object. *See, e.g.*, : The Director Runtime Library enables direct interaction with the object. (See [Sadowski11] at 28, 42, 46, 65-70.) By way of example, the Director Runtime Library plays back interactive Director animations. The videos I am submitting with this report show a

<table>
<tr><th>Claim Text from ’906 Patent</th><th>HyperCard and Director prior art</th></tr>
<tr><td></td><td>
HyperCard application with text and graphics, and a portion of the screen reserved for the Director Runtime Library.

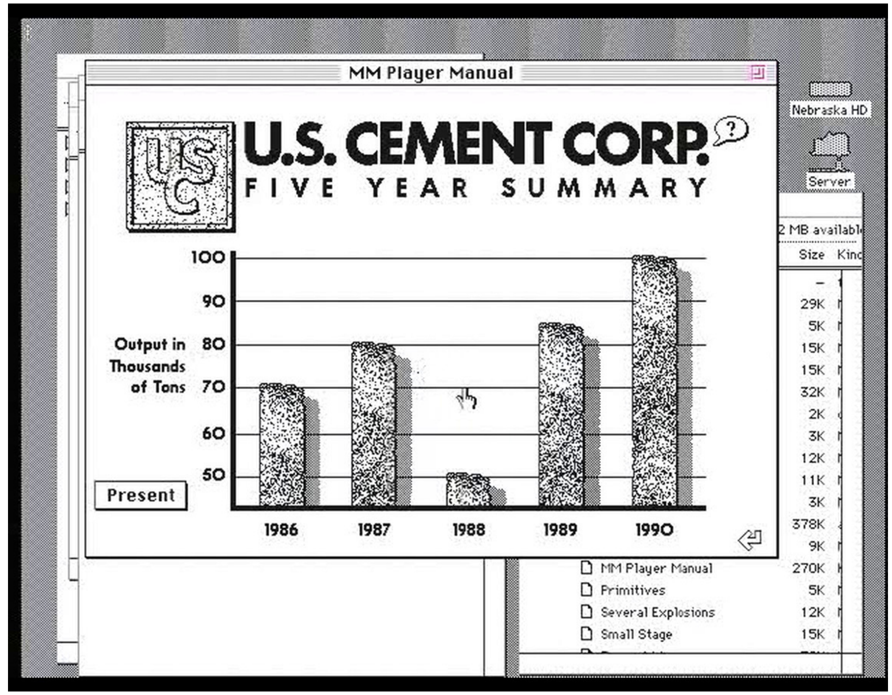
By way of example, the screenshots below show the playback of interactive Director animations within a HyperCard application. The user interacts with this animation by clicking the mouse on the bar chart. Based on the location of the click, the bar chart changes its size. In the example below, the size of the 1988 chart is changed from 90 tons to 70 tons through a user’s mouse click.

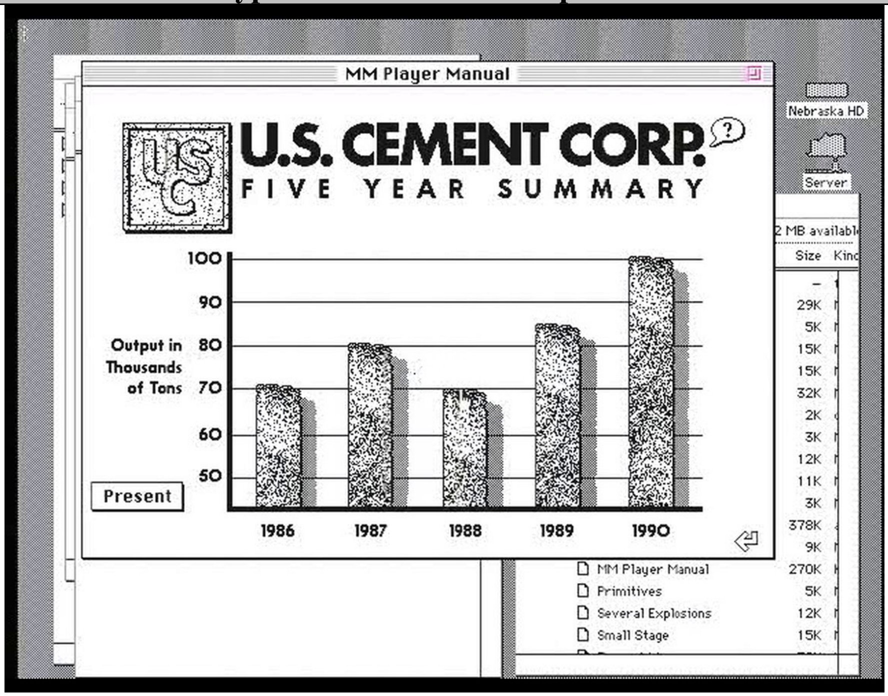


</td></tr>
</table>

Claim Text from ’906 Patent	HyperCard and Director prior art
	 HyperCard and Director discloses that interaction with the object is at a first location in the hypermedia document. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is

Claim Text from ’906 Patent	HyperCard and Director prior art
	dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a movie, Cement Column, which is specified directly and is stored elsewhere as a separate file. The object is displayed in a location that corresponds to the first location (e.g. the segment of the stack corresponding to the card). Interaction with the object at the first location is provided by mouse clicks within the object which resizes the size of columns in the chart depending on the location of the mouse click. (See e.g., [Sadowski11] at 72-74]). The videos submitted with this report show examples of this interaction. The screenshots below exemplify the interaction:

Claim Text from '906 Patent	HyperCard and Director prior art
	

Claim Text from ’906 Patent	HyperCard and Director prior art
	
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.	HyperCard and Director discloses interactive control via inter-process communications between a browser and an application. *See, e.g.*, : HyperCard communicates with XCMDs to invoke the Director Runtime Library, including the Director HyperCard Player. (See, e.g., Goodman at pp. 751, 767; see also [Sadowski11] at 17, 46, 50, 65-69.) HyperCard communicates with XCMDs using callbacks. The XCMDs interoperate with the Director HyperCard Player extension to allow playback and interaction with an object.
906-3.a:	HyperCard and Director discloses ongoing inter-process communications. *See,*

Claim Text from ’906 Patent	HyperCard and Director prior art
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.	*e.g.*, : HyperCard communicates with XCMDs using callbacks. (See, e.g., Goodman at pp. 751, 767; see also [Sadowski11] at 17, 50, 65-69.) The callback-based communications are ongoing because they occur as a user interacts with a movie object. The XCMDs interoperate with the Director HyperCard Player extension to allow playback and interaction with an object.
906-6.a: A computer program product for use in a system having at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment, the computer program product comprising:	HyperCard and Director discloses an application program in a computer network environment. *See* evidence recited for 906-1.a. HyperCard prior art 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:	HyperCard and Director discloses computer code physically embodied on a medium. *See, e.g.*, : The computer on which HyperCard executes includes a computer usable media having computer readable program code physically embodied therein. As one example, the videos I am submitting with this report show HyperCard version 2.1 executing on Apple Macintosh IIsi computers running a System 7.1 operating system. (See also Goodman at pp. 17-20) (describing installation of HyperCard onto a computer.) Below is a screenshot from my video of HyperCard 2.1:

Claim Text from '906 Patent	HyperCard and Director prior art
	
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 identify text formats included in said distributed hypermedia document and to respond to predetermined text formats to initiate processes specified by said text formats;	HyperCard and Director discloses a browser application that parses a hypermedia document with text formats. *See* evidence recited for 906-1.c.
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	HyperCard and Director discloses a hypermedia document received from a server and a browser that displays the hypermedia document. *See* evidence recited for 906-1.d.

Claim Text from ’906 Patent	HyperCard and Director prior art
over said network from said server,	
906-6.e: wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation,	HyperCard and Director 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,	HyperCard and Director 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	HyperCard and Director 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 created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.	HyperCard and Director 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.
906-7.a: The computer program product of claim 6, wherein said executable application is a controllable	HyperCard and Director discloses interactive control via inter-process communications between a browser and an application. *See* evidence recited for 906-2.a.

Claim Text from ’906 Patent	HyperCard and Director prior art
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.	
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.	HyperCard and Director 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 substeps:	HyperCard and Director discloses additional instructions on the server. *See, e.g.*, : HyperCard interoperated with distributed applications. For example, HyperCard provided for XCMDs that interoperated with applications executing on server computers or network-connected devices. These applications had additional instructions that allowed them to execute on the server. For example, HyperCard can connect to “any other computer (like a bulletin board service, MCI Mail, or Dow Jones News Retrieval) that offers asynchronous modem access” through HyperTalk script control. (See Goodman at pp. 725 – 726. see also [Sadowski11] at 54, 55, 60, 74, 76.) As another example, “HyperCard is also actively used in business as a tool to design what are known as 'front ends' to information stored on IBM (and other) mainframe computers.” (See Goodman at p. 726.) To accomplish the connections to an IBM mainframe computer, external

Claim Text from ’906 Patent	HyperCard and Director prior art
	commands (XCMDs), also called Application Programming Interfaces (APIs), that link HyperCard to a 3270-style terminal (a terminal that is used to connect to IBM mainframe computers) are added to HyperCard. These are typically supplied by the 3270-style hardware manufacturers. (See, e.g., Goodman at p. 727.) Concentrix Technology, Inc. designed front ends to IBM’s PROFS using Avatar, DCA and Tri-data APIs (XCMDs). IBM’s PROFS is an electronic mail and group scheduling program that runs on IBM mainframe computers. (See, e.g., Goodman at p. 727 – 728.) These front ends together with IBM’s PROFS constitute a distributed application. Another example is described in [Powers], in which the author developed software to enable a Macintosh to communicate with an IBM mainframe computer from within HyperCard. As another example, “HyperCard is also used extensively in business for accessing Structured Query Language (SQL) databases, usually running on mainframes or minicomputers (but also on database servers on local area networks).” (See Goodman at p. 728.) The databases, e.g., Oracle and Sybase, provide XCMD toolkits for HyperCard users to allow HyperCard stacks to access, retrieve and write data to the databases. “The HyperCard XCMDs extract the data, and regular HyperTalk scripting puts the data into fields or draws graphs based on that data.” See, e.g., Goodman at pp. 727 – 728. The XCMD together with the application executing on the databases (connected through network) constitute a distributed application. As another example in which a HyperCard XCMD served as a front end to applications running on remote servers, [Morgan] discloses XCMDs that enable TCP-based client-server interactions. The XCMD together with the applications running on the remote servers constituted a distributed application. Morgan discloses two examples: Mini-atlas and Listmanager. “A connection is established using the TCPActiveOpen function, which establishes a connection with the remote socket (a connection between

Claim Text from ’906 Patent	HyperCard and Director prior art
	computer processors allowing them to communicate in a fast, reliable manner) and returns a connection ID.” (See Morgan at 421.) “Alternatively, TCPPassiveOpen will allow a connection to be accepted on a particular socket.” (See Morgan at 422.) “Mini-Atlas is a client for the Geographic Name Server. The Geographic Name Server contains brief information about most United States cities and geographic landmarks.” (See Morgan at 422.) “Another, more interesting application is the ListManager, a front end to LISTSERV programs operating electronic lists such as PACSL, AUTOCAT, and LIBREF-L. ListManager automates the procedures necessary to search the archives of these lists by keyword Boolean queries, tum off mail from the list temporarily, retrieve a list of the list's participants, or retrieve files from the lists.” (See Morgan at 423.)
906-11.b: issuing, from the client workstation, one or more commands to the network server;	HyperCard and Director discloses that the client issues commands to the server. *See, e.g.*, : HyperCard interoperated with distributed applications. For example, HyperCard provided for XCMDs that interoperated with applications executing on server computers or network-connected devices. HyperCard, through the XCMDs, issued commands to the applications executing on the server computers. For example, HyperCard can connect to “any other computer (like a bulletin board service, MCI Mail, or Dow Jones News Retrieval) that offers asynchronous modem access” through HyperTalk script control. (See Goodman at pp. 725 – 726; see also [Sadowski11] at 54, 55, 58, 60.)) As another example, “HyperCard is also actively used in business as a tool to design what are known as 'front ends' to information stored on IBM (and other) mainframe computers.” (See Goodman at p. 726.) To accomplish the connections to an IBM mainframe computer, external commands (XCMDs), also called Application Programming Interfaces (APIs), that link HyperCard to a 3270-style terminal (a terminal that is used to connect to IBM mainframe computers) are added to HyperCard.

Claim Text from '906 Patent	HyperCard and Director prior art
	These are typically supplied by the 3270-style hardware manufacturers. (See, e.g., Goodman at p. 727.) Concentrix Technology, Inc. designed front ends to IBM's PROFS using Avatar, DCA and Tri-data APIs (XCMDs). IBM's PROFS is an electronic mail and group scheduling program that runs on IBM mainframe computers. (See, e.g., Goodman at p. 727 – 728.) These front ends together with IBM's PROFS constitute a distributed application. Another example is described in [Powers], in which the author developed software to enable a Macintosh to communicate with an IBM mainframe computer from within HyperCard. As another example, "HyperCard is also used extensively in business for accessing Structured Query Language (SQL) databases, usually running on mainframes or minicomputers (but also on database servers on local area networks)." (See Goodman at p. 728.) The databases, e.g., Oracle and Sybase, provide XCMD toolkits for HyperCard users to allow HyperCard stacks to access, retrieve and write data to the databases. "The HyperCard XCMDs extract the data, and regular HyperTalk scripting puts the data into fields or draws graphs based on that data." See, e.g., Goodman at pp. 727 – 728. The XCMD together with the application executing on the databases (connected through network) constitute a distributed application. As another example in which a HyperCard XCMD served as a front end to applications running on remote servers, [Morgan] discloses XCMDs that enable TCP-based client-server interactions. The XCMD together with the applications running on the remote servers constituted a distributed application. Morgan discloses two examples: Mini-atlas and Listmanager. "A connection is established using the TCPActiveOpen function, which establishes a connection with the remote socket (a connection between computer processors allowing them to communicate in a fast, reliable manner) and returns a connection ID." (See Morgan at 421.) "Alternatively, TCPPassiveOpen will allow a connection to be accepted

Claim Text from ’906 Patent	HyperCard and Director prior art
	on a particular socket.” (See Morgan at 422.) “Mini-Atlas is a client for the Geographic Name Server. The Geographic Name Server contains brief information about most United States cities and geographic landmarks.” (See Morgan at 422.) “Another, more interesting application is the ListManager, a front end to LISTSERV programs operating electronic lists such as PACSL, AUTOCAT, and LIBREF-L. ListManager automates the procedures necessary to search the archives of these lists by keyword Boolean queries, tum off mail from the list temporarily, retrieve a list of the list's participants, or retrieve files from the lists.” (See Morgan at 423.)
906-11.c: executing, on the network server, one or more instructions in response to said commands;	HyperCard and Director discloses that the server executes instructions in response to client commands. *See, e.g.*, : HyperCard interoperated with distributed applications. For example, HyperCard provided for XCMDs that interoperated with applications executing on server computers or network-connected devices. The applications on the server executed in response to commands from the XCMDs. Additionally, HyperCard can connect to “any other computer (like a bulletin board service, MCI Mail, or Dow Jones News Retrieval) that offers asynchronous modem access” through HyperTalk script control. (See Goodman at pp. 725 – 726; see also [Sadowski11] at 54, 55, 60, 74, 76.) As another example, “HyperCard is also actively used in business as a tool to design what are known as 'front ends' to information stored on IBM (and other) mainframe computers.” (See Goodman at p. 726.) To accomplish the connections to an IBM mainframe computer, external commands (XCMDs), also called Application Programming Interfaces (APIs), that link HyperCard to a 3270-style terminal (a terminal that is used to connect to IBM mainframe computers) are added to HyperCard. These are typically supplied by the 3270-style hardware manufacturers. (See, e.g., Goodman at p. 727.) Concentrix Technology, Inc. designed

Claim Text from '906 Patent	HyperCard and Director prior art
	front ends to IBM's PROFS using Avatar, DCA and Tri-data APIs (XCMDs). IBM's PROFS is an electronic mail and group scheduling program that runs on IBM mainframe computers. (See, e.g., Goodman at p. 727 – 728.) These front ends together with IBM's PROFS constitute a distributed application. Another example is described in [Powers], in which the author developed software to enable a Macintosh to communicate with an IBM mainframe computer from within HyperCard As another example, "HyperCard is also used extensively in business for accessing Structured Query Language (SQL) databases, usually running on mainframes or minicomputers (but also on database servers on local area networks)." (See Goodman at p. 728.) The databases, e.g., Oracle and Sybase, provide XCMD toolkits for HyperCard users to allow HyperCard stacks to access, retrieve and write data to the databases. "The HyperCard XCMDs extract the data, and regular HyperTalk scripting puts the data into fields or draws graphs based on that data." See, e.g., Goodman at pp. 727 – 728. The XCMD together with the application executing on the databases (connected through network) constitute a distributed application. As another example in which a HyperCard XCMD served as a front end to applications running on remote servers, [Morgan] discloses XCMDs that enable TCP-based client-server interactions. The XCMD together with the applications running on the remote servers constituted a distributed application. Morgan discloses two examples: Mini-atlas and Listmanager. "A connection is established using the TCPActiveOpen function, which establishes a connection with the remote socket (a connection between computer processors allowing them to communicate in a fast, reliable manner) and returns a connection ID." (See Morgan at 421.) "Alternatively, TCPPassiveOpen will allow a connection to be accepted on a particular socket." (See Morgan at 422.) "Mini-Atlas is a client for the Geographic Name Server. The Geographic Name Server contains brief

Claim Text from ’906 Patent	HyperCard and Director prior art
	information about most United States cities and geographic landmarks.” (See Morgan at 422.) “Another, more interesting application is the ListManager, a front end to LISTSERV programs operating electronic lists such as PACSL, AUTOCAT, and LIBREF-L. ListManager automates the procedures necessary to search the archives of these lists by keyword Boolean queries, tum off mail from the list temporarily, retrieve a list of the list's participants, or retrieve files from the lists.” (See Morgan at 423.)
906-11.d: sending information from said network server to said client workstation in response to said executed instructions; and	HyperCard and Director discloses that the server responds with information to the client. *See, e.g.*, : HyperCard interoperated with distributed applications. As an example, HyperCard provided for XCMDs that interoperated with applications executing on server computers or network-connected devices. In response to communication from the XCMD, the server application responded with information to the client. For example, HyperCard can connect to “any other computer (like a bulletin board service, MCI Mail, or Dow Jones News Retrieval) that offers asynchronous modem access” through HyperTalk script control. (See Goodman at pp. 725 – 726; see also [Sadowski11] at 54, 55, 60, 74, 76.) As another example, “HyperCard is also actively used in business as a tool to design what are known as 'front ends' to information stored on IBM (and other) mainframe computers.” (See Goodman at p. 726.) To accomplish the connections to an IBM mainframe computer, external commands (XCMDs), also called Application Programming Interfaces (APIs), that link HyperCard to a 3270-style terminal (a terminal that is used to connect to IBM mainframe computers) are added to HyperCard. These are typically supplied by the 3270-style hardware manufacturers. (See, e.g., Goodman at p. 727.) Concentrix Technology, Inc. designed front ends to IBM’s PROFS using Avatar, DCA and Tri-data APIs (XCMDs). IBM’s PROFS is an electronic mail and group scheduling

Claim Text from ’906 Patent	HyperCard and Director prior art
	program that runs on IBM mainframe computers. (See, e.g., Goodman at p. 727 – 728.) These front ends together with IBM’s PROFS constitute a distributed application. Another example is described in [Powers], in which the author developed software to enable a Macintosh to communicate with an IBM mainframe computer from within HyperCard. As another example, “HyperCard is also used extensively in business for accessing Structured Query Language (SQL) databases, usually running on mainframes or minicomputers (but also on database servers on local area networks).” (See Goodman at p. 728.) The databases, e.g., Oracle and Sybase, provide XCMD toolkits for HyperCard users to allow HyperCard stacks to access, retrieve and write data to the databases. “The HyperCard XCMDs extract the data, and regular HyperTalk scripting puts the data into fields or draws graphs based on that data.” See, e.g., Goodman at pp. 727 – 728. The XCMD together with the application executing on the databases (connected through network) constitute a distributed application. As another example in which a HyperCard XCMD served as a front end to applications running on remote servers, [Morgan] discloses XCMDs that enable TCP-based client-server interactions. The XCMD together with the applications running on the remote servers constituted a distributed application. Morgan discloses two examples: Mini-atlas and Listmanager. “A connection is established using the TCPActiveOpen function, which establishes a connection with the remote socket (a connection between computer processors allowing them to communicate in a fast, reliable manner) and returns a connection ID.” (See Morgan at 421.) “Alternatively, TCPPassiveOpen will allow a connection to be accepted on a particular socket.” (See Morgan at 422.) “Mini-Atlas is a client for the Geographic Name Server. The Geographic Name Server contains brief information about most United States cities and geographic landmarks.” (See Morgan at 422.) “Another, more interesting application is the

Claim Text from ’906 Patent	HyperCard and Director prior art
	ListManager, a front end to LISTSERV programs operating electronic lists such as PACSL, AUTOCAT, and LIBREF-L. ListManager automates the procedures necessary to search the archives of these lists by keyword Boolean queries, tum off mail from the list temporarily, retrieve a list of the list's participants, or retrieve files from the lists.” (See Morgan at 423.)
906-11.e: processing said information at the client workstation to interactively control said controllable application.	HyperCard and Director discloses that the client uses information from the server to interactively control the application. *See, e.g.*, : HyperCard interoperated with distributed applications. As an example, HyperCard provided for XCMDs that interoperated with applications executing on server computers or network-connected devices. In such usage, HyperCard and the XCMD operating on the client used information from the server. For example, HyperCard can connect to “any other computer (like a bulletin board service, MCI Mail, or Dow Jones News Retrieval) that offers asynchronous modem access” through HyperTalk script control. (See Goodman at pp. 725 – 726; see also [Sadowski11] at 54, 55, 60, 74, 76.) As another example, “HyperCard is also actively used in business as a tool to design what are known as 'front ends' to information stored on IBM (and other) mainframe computers.” (See Goodman at p. 726.) To accomplish the connections to an IBM mainframe computer, external commands (XCMDs), also called Application Programming Interfaces (APIs), that link HyperCard to a 3270-style terminal (a terminal that is used to connect to IBM mainframe computers) are added to HyperCard. These are typically supplied by the 3270-style hardware manufacturers. (See, e.g., Goodman at p. 727.) Concentrix Technology, Inc. designed front ends to IBM’s PROFS using Avatar, DCA and Tri-data APIs (XCMDs). IBM’s PROFS is an electronic mail and group scheduling program that runs on IBM mainframe computers. (See, e.g., Goodman at p. 727 – 728.) These front ends together with IBM’s PROFS constitute a

Claim Text from ’906 Patent	HyperCard and Director prior art
	distributed application. Another example is described in [Powers], in which the author developed software to enable a Macintosh to communicate with an IBM mainframe computer from within HyperCard. As another example, “HyperCard is also used extensively in business for accessing Structured Query Language (SQL) databases, usually running on mainframes or minicomputers (but also on database servers on local area networks).” (See Goodman at p. 728.) The databases, e.g., Oracle and Sybase, provide XCMD toolkits for HyperCard users to allow HyperCard stacks to access, retrieve and write data to the databases. “The HyperCard XCMDs extract the data, and regular HyperTalk scripting puts the data into fields or draws graphs based on that data.” See, e.g., Goodman at pp. 727 – 728. The XCMD together with the application executing on the databases (connected through network) constitute a distributed application. As another example in which a HyperCard XCMD served as a front end to applications running on remote servers, [Morgan] discloses XCMDs that enable TCP-based client-server interactions. The XCMD together with the applications running on the remote servers constituted a distributed application. Morgan discloses two examples: Mini-atlas and Listmanager. “A connection is established using the TCPActiveOpen function, which establishes a connection with the remote socket (a connection between computer processors allowing them to communicate in a fast, reliable manner) and returns a connection ID.” (See Morgan at 421.) “Alternatively, TCPPassiveOpen will allow a connection to be accepted on a particular socket.” (See Morgan at 422.) “Mini-Atlas is a client for the Geographic Name Server. The Geographic Name Server contains brief information about most United States cities and geographic landmarks.” (See Morgan at 422.) “Another, more interesting application is the ListManager, a front end to LISTSERV programs operating electronic lists such as PACSL, AUTOCAT, and LIBREF-L. ListManager automates the

Claim Text from ’906 Patent	HyperCard and Director prior art
	procedures necessary to search the archives of these lists by keyword Boolean queries, tum off mail from the list temporarily, retrieve a list of the list's participants, or retrieve files from the lists.” (See Morgan at 423.)
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:	HyperCard and Director 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;	HyperCard and Director 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;	HyperCard and Director 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	HyperCard and Director 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.	HyperCard and Director discloses that the client uses information from the server to interactively control the application. *See* evidence recited for 906-11.e.

Claim Text from ’906 Patent	HyperCard and Director prior art

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

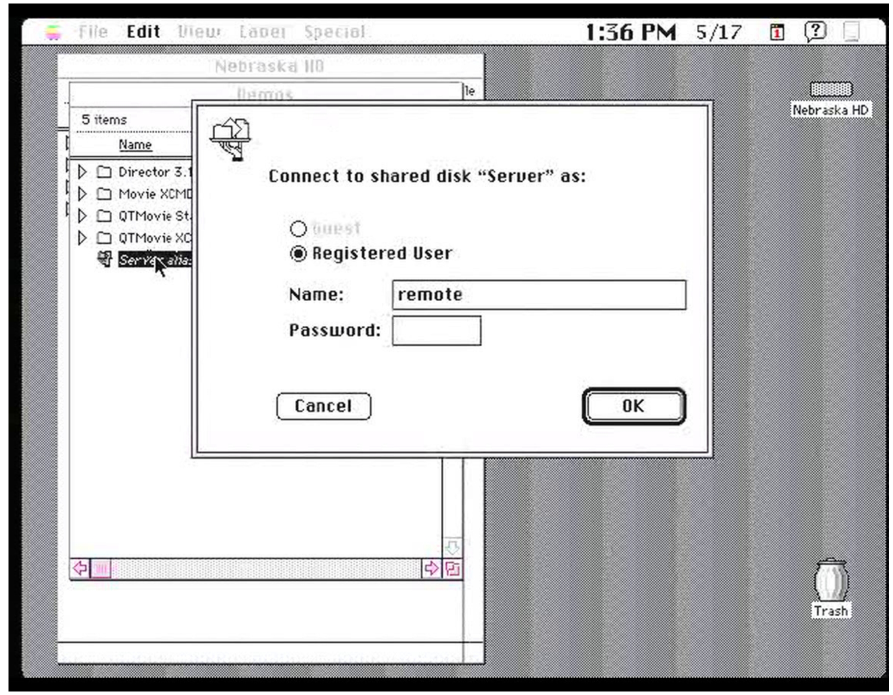
- **"HYPERCARD AND DIRECTOR"[2] -- DIRECTOR SOFTWARE [DIRECTOR], INCLUDING MACROMIND PLAYER MANUAL DISTRIBUTED WITH DIRECTOR 3.1.3 ("DIRECTOR PRIOR ART") AS INTENED TO BE USED IN A COMPUTER SYSTEM AND DEMONSTRATION OF SAME, FURTHER INFORMED BY:**
  - **DANNY GOODMAN. THE COMPLETE HYPERCARD 2.0 HANDBOOK. 3RD EDITION. BANTAM BOOKS, INC., AUGUST 1990. ("GOODMAN") [PA-00288603] [GOODMAN90];**
  - **HYPERCARD VERSIONS 2.0, 2.1, AND 2.2 ("HYPERCARD PRIOR ART") [HYPERCARD];**
  - **ERIC LEASE MORGAN. "IMPLEMENTING TCP/IP COMMUNICATIONS WITH HYPERCARD." INFORMATION TECHNOLOGY AND LIBRARIES, DEC. 1992; 11, 4; ABI/INFORM GLOBAL. PP. 421-432;**
  - **JOHN R. POWERS, III. "MAC TO MAINFRAME WITH HYPERCARD." MACTUTOR, JUNE 1990. [PA-00288589] [POWERS90]; AND**
  - **DECLARATION OF DANIEL SADOWSKI, JUNE 2011 [SADOWSKI11].**
  - **THE BODY OF MY REPORT HAS A NARRATIVE DESCRIPTION THAT AUGMENTS AND SHOULD BE CONSIDERED PART OF THIS CHART.**

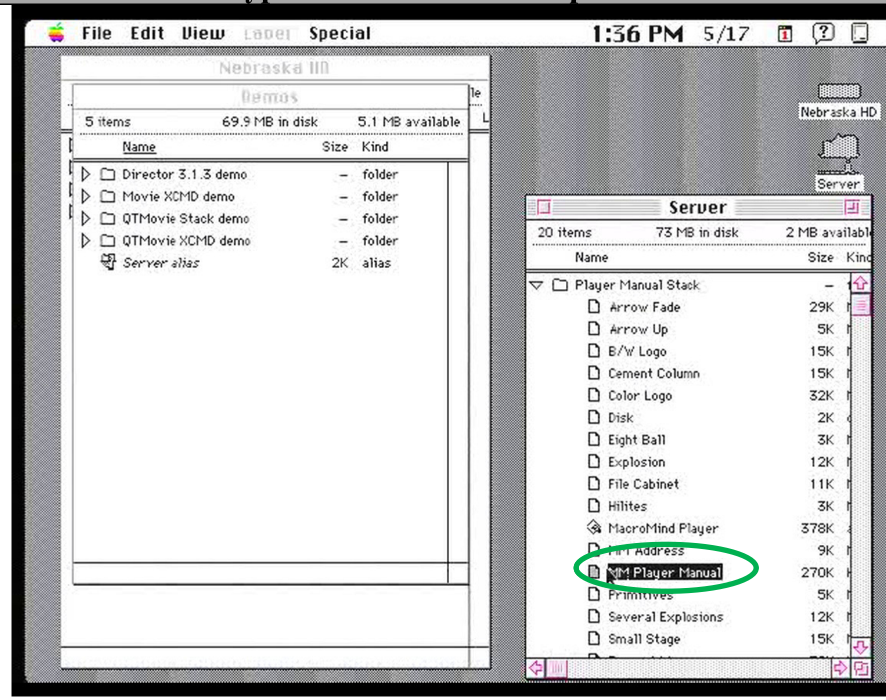
Claim Text from '985 Patent	HyperCard and Director prior art
985-1.a: A method for running an application program in a distributed hypermedia network environment, wherein the network environment comprises at least one client workstation and one network server coupled to the network environment, the method comprising:	HyperCard and Director discloses an application program. *See, e.g.*, : HyperCard is a computer program. It was installed onto a computer, such as an Apple Macintosh computer, and launched as an executable program. The videos that I am submitting with this report show how this was done. (See also Goodman at pp. 17-20) (describing installation and operation of HyperCard on a computer); (Goodman at pp. xxxiii) (describing that HyperCard is a computer program).

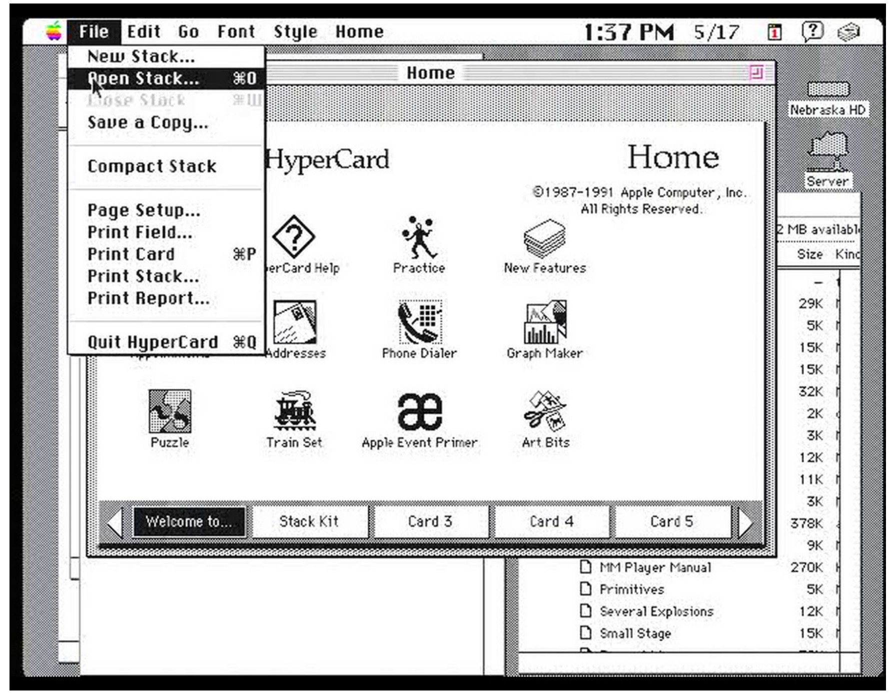
[2] For all asserted claims this reference is a 103 reference due to my understanding of the plain meaning of the limitations relating to "location" (e.g. 901-1.f and 906-1.g and 985-1.f and 985.1g) and the Court's discussion of the issue on page 17 of its August 22, 2011 Order. Thus, for these particular limitations, the reference is not anticipatory, but rather, as explained in the body of my report, this limitation would be combined with a prior art web browser like Mosaic, CERN's web browser, Viola, or MediaView. Likewise, to satisfy the HTML limitations in the '985 patent, the reference must be combined with a web browser or HTML teaching, such as Mosaic, CERN's web browser, or Viola. For both all such limitations it would have been obvious to a person of ordinary skill in the art at the time to do so as explained in the body of my report and the teachings, for example, of Tim Berners-Lee posted on the CERN website discussing the Web and relating features and pointers to other browser technologies including HyperCard, Viola and MediaView. This was an obvious and natural extension of prior hypermedia functions and features and an inevitable development in the marketplace at the time of the invention and based on the state of the art.

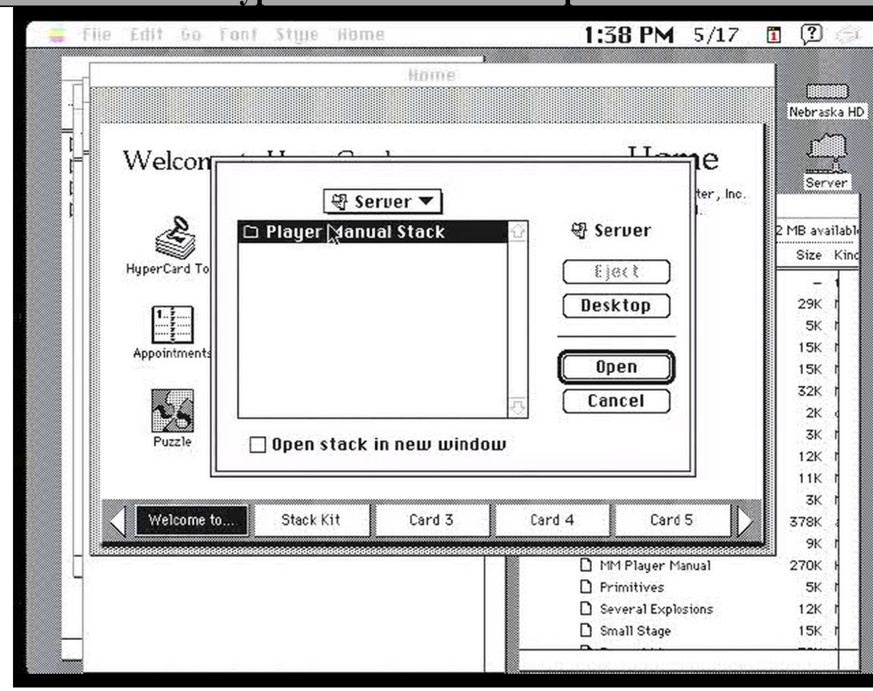
Claim Text from ’985 Patent	HyperCard and Director prior art
	HyperCard and Director discloses a computer network environment. *See, e.g.*, : HyperCard operating in a distributed hypermedia environment that included clients and servers. Specifically, stacks can be stored on or published to a file server and then accessed by a user using HyperCard on a client workstation. (See Goodman at pp. 737-739.) “On networks such as AppleShare and TOPS, HyperCard sees file servers or other user’s published volumes just as another disk drive attached to the Macintosh. If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname cared for stacks…” (See Goodman at p. 737.) “If you develop what we call information publishing stacks – those that come full of information for users to browse through – you should be aware that such a stack might be used on a network.” (See Goodman at p. 739.) Stacks shared over a network “should be on the file server or, in the case of a TOPS network, on a published volume.” (See Goodman at p. 738; see also [Sadowski11] at 54, 55, 60, 74, 76.) A network of computers containing HyperCard stacks is a distributed hypermedia environment because HyperCard stacks contain hypermedia. (See, e.g., Goodman at Chapter 52) (describing text, sound, animation, and movies.) ("If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname card for stacks, just as it would be if it were on your own hard disk"). As another example, HyperCard could interoperate with the TCP/IP Internet. A software package called MacTCP and a set of XCMDs provided with the HyperCard TCP Toolkit provided this functionality. (See Morgan at 421.)

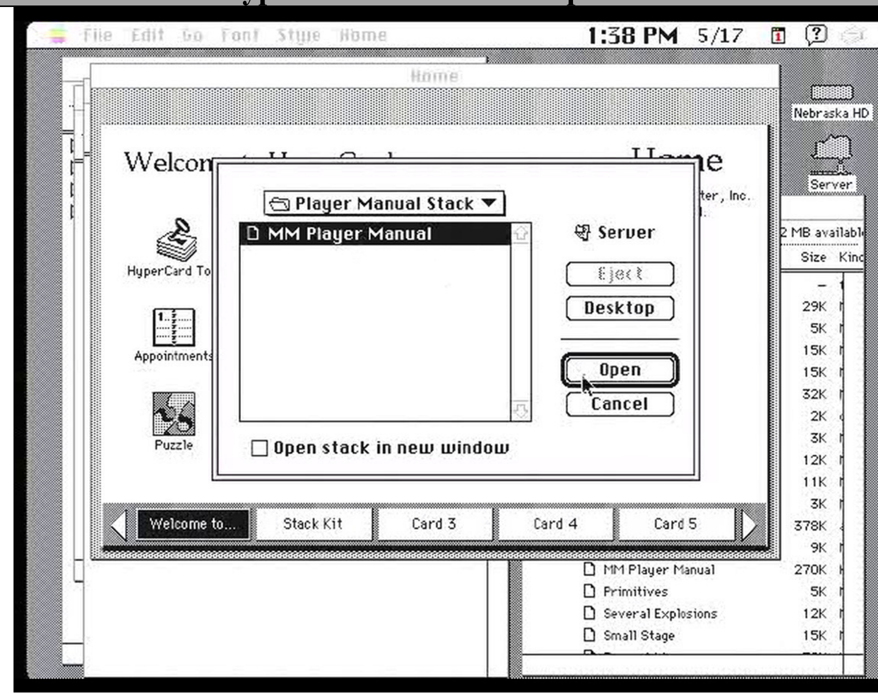
Claim Text from ’985 Patent	HyperCard and Director prior art
	HyperCard and Director discloses a client workstation. *See, e.g.*, : HyperCard works in a distributed environment that includes clients and servers. Specifically, stacks can be stored on or published to a file server and then accessed by a user using HyperCard on a client workstation. (See Goodman at pp. 737-739; see also [Sadowski11] at 54, 55, 60, 74, 76.) “On networks such as AppleShare and TOPS, HyperCard sees file servers or other user’s published volumes just as another disk drive attached to the Macintosh. If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname cared for stacks…” (See Goodman at p. 737.) “If you develop what we call information publishing stacks – those that come full of information for users to browse through – you should be aware that such a stack might be used on a network.” (See Goodman at p. 739.) Stacks shared over a network “should be on the file server or, in the case of a TOPS network, on a published volume.” (See Goodman at p. 738.) One mechanism by which stacks on a server were accessed from a client was through buttons. (See Goodman at p. 737) ("If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname card for stacks, just as it would be if it were on your own hard disk") Examples of HyperCard’s client-server functionality are shown in the videos I am submitting with this report. In addition, screenshots from my video depicting the client-server functionality are shown below. For example, the screenshot below shows a client-server arrangement. This first screenshot shows the connection to the server.

Claim Text from '985 Patent	HyperCard and Director prior art
	 The screenshot below shows the MM Player Manual stack file being stored remotely on a server. The MM Player Manual stack is highlighted in green.

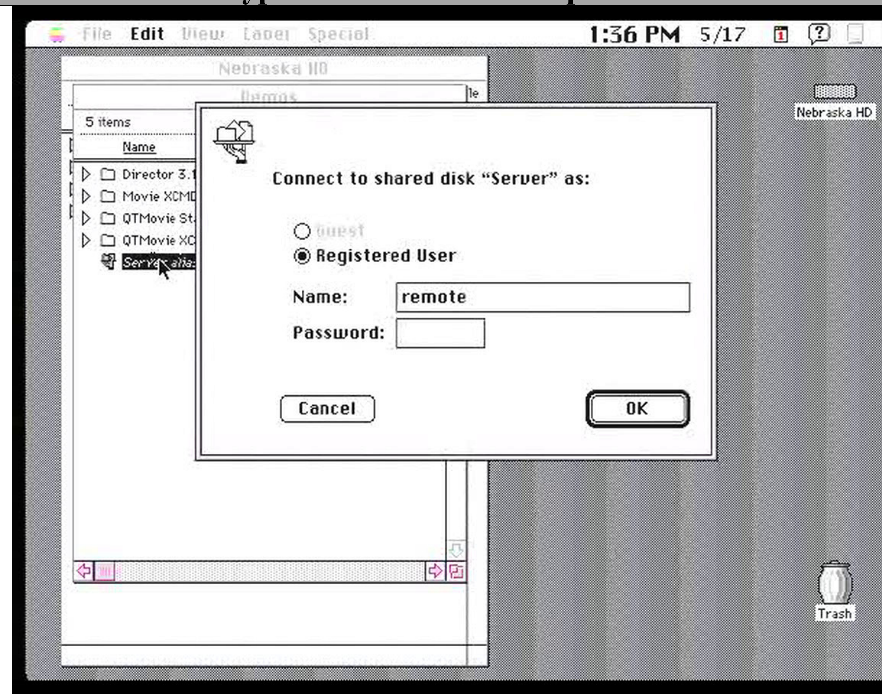
Claim Text from '985 Patent	HyperCard and Director prior art
	 The screenshots below show the sequence of steps taken to open the MM Player Manual stack which is stored remotely on a server. The steps are taken on a client workstation.

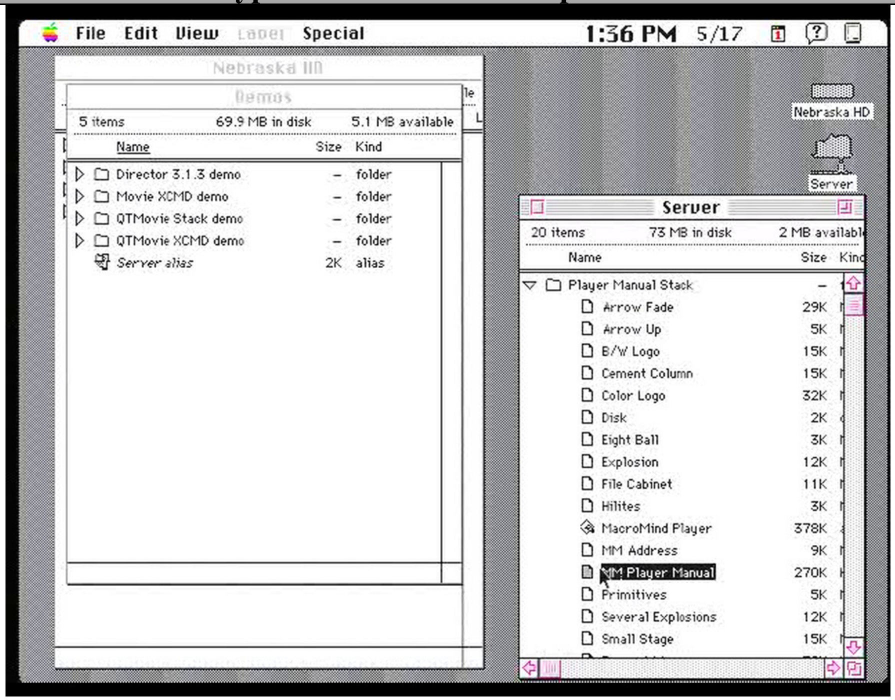
Claim Text from ’985 Patent	HyperCard and Director prior art
	

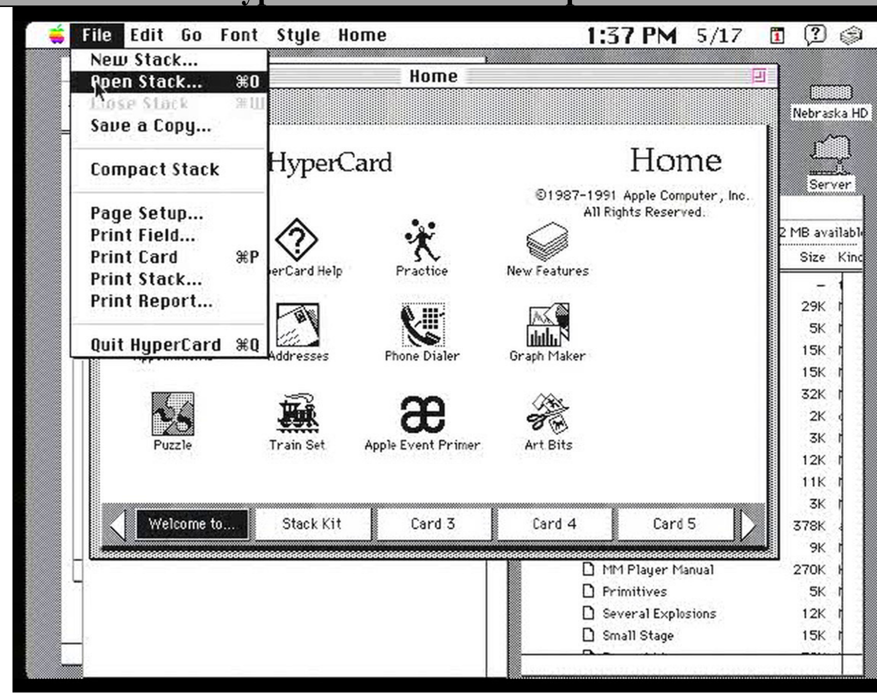
Claim Text from ’985 Patent	HyperCard and Director prior art
	

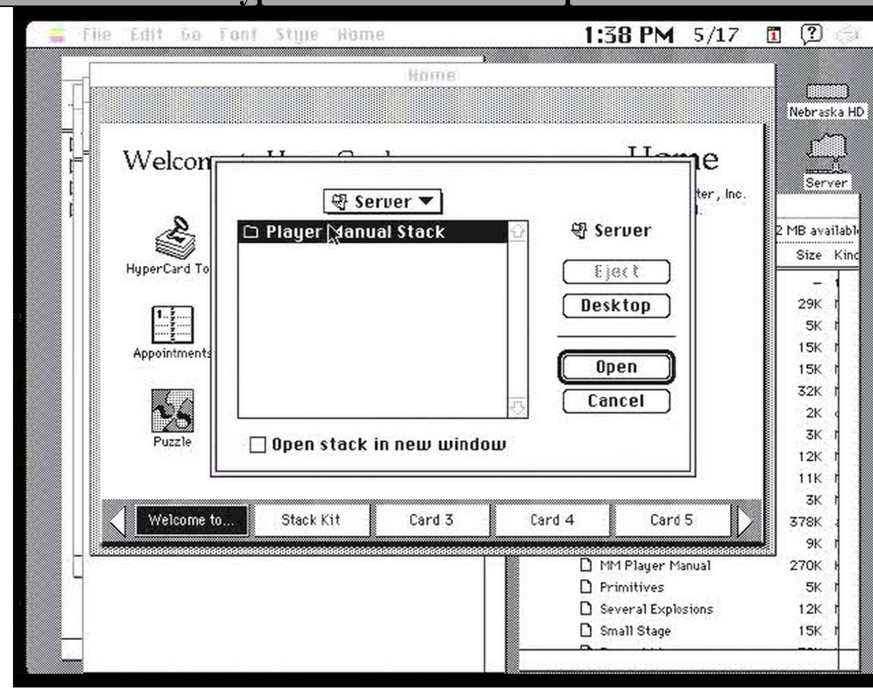
Claim Text from ’985 Patent	HyperCard and Director prior art
	 HyperCard and Director discloses a network server. *See, e.g.*, : HyperCard works in a distributed environment that includes clients and servers. Specifically, stacks can be stored on or published to a file server and then accessed by HyperCard on a client workstation. (See Goodman at pp. 737-739.) “On networks such as AppleShare and TOPS, HyperCard sees file servers or other user’s published volumes just as another disk drive attached to the Macintosh. If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname cared for stacks…”

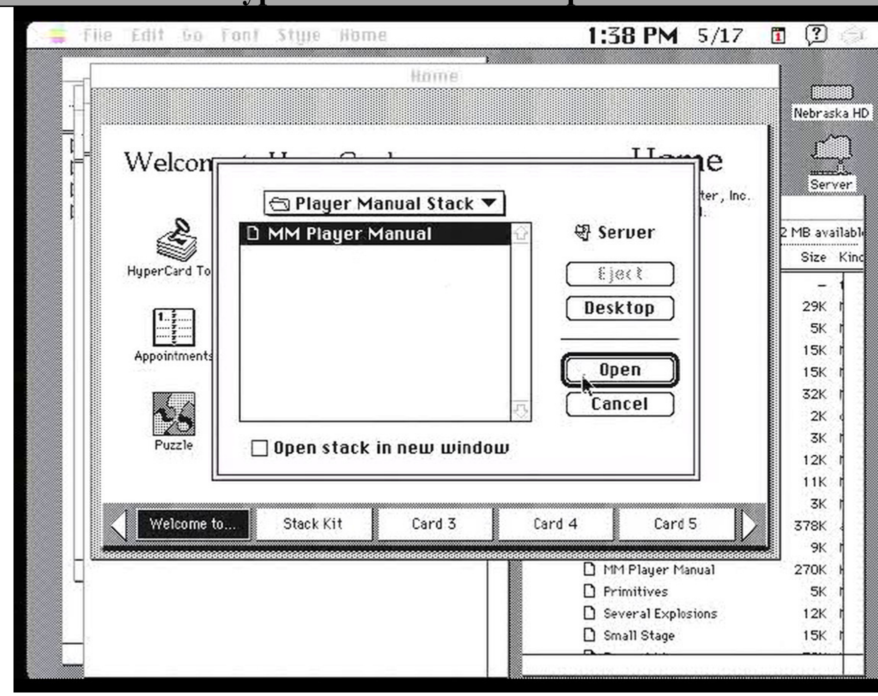
Claim Text from ’985 Patent	HyperCard and Director prior art
	(See Goodman at p. 737.) “If you develop what we call information publishing stacks – those that come full of information for users to browse through – you should be aware that such a stack might be used on a network.” (See Goodman at p. 739.) Stacks shared over a network “should be on the file server or, in the case of a TOPS network, on a published volume.” (See Goodman at p. 738; see also [Sadowski11] at 54, 55, 60, 74, 76.) One mechanism by which stacks on a server were accessed from a client was through buttons. (See Goodman at p. 737) ("If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname card for stacks, just as it would be if it were on your own hard disk"). Examples of HyperCard's client-server functionality are shown in the videos I am submitting with this report. In addition, screenshots from my video showing the client-server functionality are shown below. The screenshot below shows a client-server arrangement.

Claim Text from ’985 Patent	HyperCard and Director prior art
	 The screenshot below shows the MM Player Manual stack file being stored remotely on a server.

Claim Text from ’985 Patent	HyperCard and Director prior art
	 The screenshots below show the sequence of steps taken to open the MM Player Manual stack file which is stored remotely on a server.

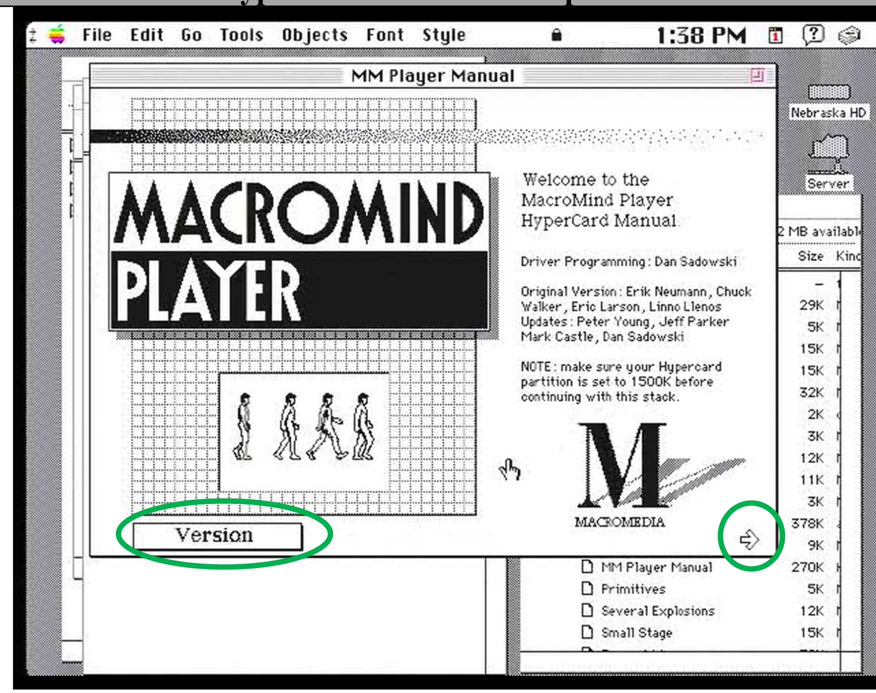
Claim Text from ’985 Patent	HyperCard and Director prior art
	

Claim Text from ’985 Patent	HyperCard and Director prior art
	

Claim Text from ’985 Patent	HyperCard and Director prior art
	 HyperCard and Director discloses a distributed hypermedia environment. *See, e.g.*, : HyperCard operating in a distributed hypermedia environment that included clients and servers. Specifically, stacks can be stored on or published to a file server and then accessed by a user using HyperCard on a client workstation. (See Goodman at pp. 737-739.) “On networks such as AppleShare and TOPS, HyperCard sees file servers or other user’s published volumes just as another disk drive attached to the Macintosh. If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s

Claim Text from ’985 Patent	HyperCard and Director prior art
	pathname cared for stacks…” (See Goodman at p. 737.) “If you develop what we call information publishing stacks – those that come full of information for users to browse through – you should be aware that such a stack might be used on a network.” (See Goodman at p. 739.) Stacks shared over a network “should be on the file server or, in the case of a TOPS network, on a published volume.” (See Goodman at p. 738; see also [Sadowski11] at 54, 55, 60, 74, 76.) A network of computers containing HyperCard stacks is a distributed hypermedia environment because HyperCard stacks contain hypermedia. (See, e.g., Goodman at Chapter 52) (describing text, sound, animation, and movies.) ("If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname card for stacks, just as it would be if it were on your own hard disk"). As another example, HyperCard could interoperate with the TCP/IP Internet. A software package called MacTCP and a set of XCMDs provided with the HyperCard TCP Toolkit provided this functionality. (See Morgan at 421.)
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;	HyperCard and Director discloses a browser application. *See, e.g.*, : HyperCard is a browser application because it displays hypermedia documents and allows a user to browse through different parts of the hypermedia documents and to other hypermedia documents using, for example, buttons and links between cards and stacks. As one example, HyperCard provided buttons, which were a navigational tool used to browse through a HyperCard stack. (See Goodman at p. 35); (Goodman at Chapter 11.)

Claim Text from ’985 Patent	HyperCard and Director prior art
	The functionality of buttons could be specified using the HyperTalk language. At the core of most HyperCard button activity is the link, which ties together one card with another card. That other card can be the next card in the stack; a previously viewed card in the same or a different stack; the first card in another stack; or a specific card in another stack. (See Goodman at Chapter 12.) HyperCard also provided for linked text. (See Goodman at 72.) Buttons could link to cards or stacks on the same computer, or on a server computer. (See Goodman at p. 737.) Examples of all this functionality are shown in the videos I am submitting with this report. Below is a screenshot from my video of the HyperCard application. By way of clarification, I have also highlighted navigational buttons in green from the HyperCard application.

<table>
<tr><th>Claim Text from ’985 Patent</th><th>HyperCard and Director prior art</th></tr>
<tr><td></td><td>


HyperCard and Director discloses a file containing enabling information. See, e.g., :

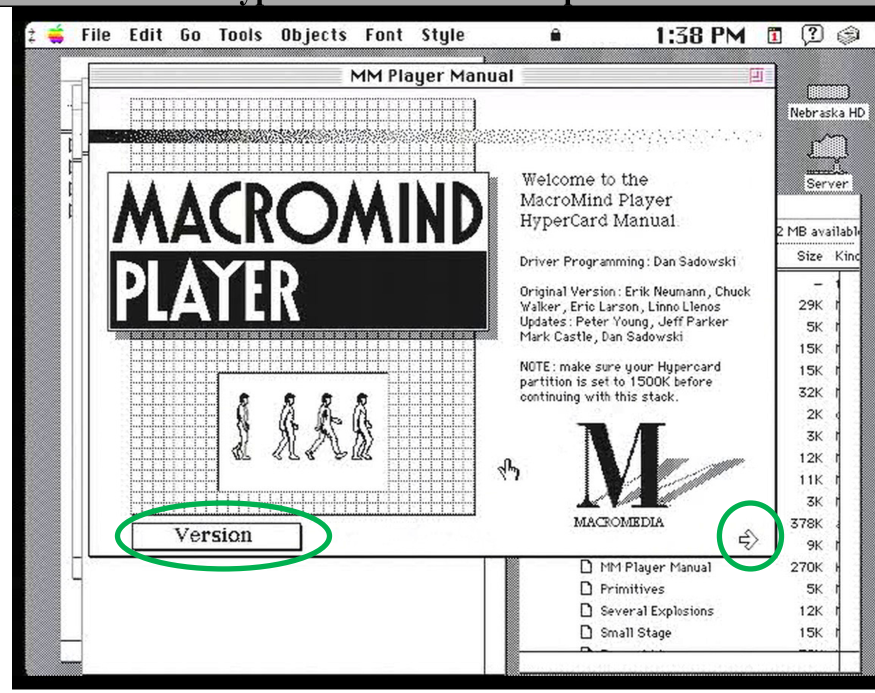
<blockquote>
HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file.

Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When
</blockquote>
</td></tr>
</table>

Claim Text from ’985 Patent	HyperCard and Director prior art
	presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” The script code found during parsing a segment is enabling information. The videos that I am submitting with this report show examples of HyperTalk scripts associated with hypermedia cards. HyperCard and Director discloses that the file is received at the client workstation from the network server. *See, e.g.*, : HyperCard works in a distributed environment that includes clients and servers. In such environments, HyperCard operating on a client computer receives HyperCard files from a server computer. Specifically, stacks can be stored on or published to a file server and then accessed by a user using HyperCard on a client workstation. (See Goodman at pp. 737-739.) “On networks such as AppleShare and TOPS, HyperCard sees file servers or other user’s published volumes just as another disk drive attached to the Macintosh. If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname cared for stacks…” (See Goodman at p. 737.) “If you develop what we call information publishing stacks – those that come full of information for users to browse through – you should be aware that such a stack might be used on a network.” (See Goodman at p. 739.) Stacks shared over a network “should be on the file server or, in the case of a TOPS network, on a published volume.” (See Goodman at p. 738; see also [Sadowski11] at 54, 55, 60, 74, 76.)

Claim Text from ’985 Patent	HyperCard and Director prior art
	HyperCard operating on a client received hypermedia files from a server. As one example, a button could be configured to retrieve hypermedia cards from a server. (See Goodman at p. 737) ("If you have a button in a stack on your own disk that is linked to a stack on a file server, the pathname for that server stack will be stored in your Home stack’s pathname card for stacks, just as it would be if it were on your own hard disk"). Examples of HyperCard's client-server functionality are shown in the videos I am submitting with this expert report. HyperCard and Director discloses that the browser displays at least a portion of a distributed hypermedia document. *See, e.g.*, : HyperCard displays cards that are part of stacks. Cards are hypermedia documents because they can include a variety of media types, including text, sound, animation, and movies. (See, e.g., Goodman at Chapter 52) (describing text, sound, animation, and movies in cards.) The videos that I am submitting with this report show examples of text, animations, movies, and interactive movies. HyperCard and Director discloses that at least a portion of a hypermedia document is displayed in a browser-controlled window. *See, e.g.*, : HyperCard displays hypermedia cards in a HyperCard window. By way of example, hypermedia cards displayed in the HyperCard window are shown in Goodman at pp. 791-836. In addition, the videos I am submitting with this report show hypermedia cards displayed within a HyperCard window.
985-1.c: executing the browser application on the client workstation, with the browser application:	HyperCard and Director discloses a browser application executing on the client workstation. *See, e.g.*, :

Claim Text from ’985 Patent	HyperCard and Director prior art
	HyperCard is a browser application because it displays hypermedia documents and allows a user to browse through different parts of the hypermedia documents and to other hypermedia documents using, for example, buttons and links between cards and stacks. As one example, HyperCard provided buttons, which were a navigational tool used to browse through a HyperCard stack. (See Goodman at p. 35); (Goodman at Chapter 11.) The functionality of buttons could be specified using the HyperTalk language. At the core of most HyperCard button activity is the link, which ties together one card with another card. That other card can be the next card in the stack; a previously viewed card in the same or a different stack; the first card in another stack; or a specific card in another stack. (See Goodman at Chapter 12.) HyperCard also provided for linked text. (See Goodman at 72.) Buttons could link to cards or stacks on the same computer, or on a server computer. (See Goodman at p. 737.) Examples of all this functionality are shown in the videos I am submitting with this report. Below is a screenshot from my video of the HyperCard application. By way of clarification, I have also highlighted navigational buttons in green from the HyperCard application.

Claim Text from ’985 Patent	HyperCard and Director prior art
	
985-1.d: responding to text formats to initiate processing specified by the text formats;	HyperCard and Director discloses responding to text formats to initiate processing specified by the text formats, i.e., parsing text formats. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is

Claim Text from '985 Patent	HyperCard and Director prior art
	parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates."
985-1.e: displaying at least a portion of the document within the browser-controlled window;	HyperCard and Director discloses that the browser displays a hypermedia document. *See, e.g.*, : HyperCard displays cards that are part of stacks. Cards are hypermedia documents because they can include a variety of media types, including text, sound, animation, and movies. (See, e.g., Goodman at Chapter 52) (describing text, sound, animation, and movies in cards.) The videos that I am submitting with this report show examples of text, animations, movies, and interactive movies. HyperCard and Director discloses that a hypermedia document is displayed in a browser window. *See, e.g.*, : HyperCard displays hypermedia cards in a HyperCard window. By way of example, hypermedia cards displayed in the HyperCard window are shown in Goodman at pp. 791-836. In addition, the videos I am submitting with this report show hypermedia cards displayed within a HyperCard window.
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;	HyperCard and Director discloses identifying an embed text format. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is

Claim Text from '985 Patent	HyperCard and Director prior art
	dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie "Cement Column" lock screen play returnSound pop card unlock screen with wipe right end presentMovie HyperCard and Director discloses that the embed text format corresponds to a first location in the hypermedia document. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise

Claim Text from ’985 Patent	HyperCard and Director prior art
	coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie HyperCard and Director discloses that the embed text format specifies the location of an object. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a

Claim Text from '985 Patent	HyperCard and Director prior art
	HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie "Cement Column" lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name "Cement Column," stored elsewhere as a separate file. (See e.g., [Sadowski11] at 72-74]). HyperCard and Director discloses that the object is external to the file containing enabling information. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered

Claim Text from ’985 Patent	HyperCard and Director prior art
	at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name “Cement Column,” which is specified directly and is stored elsewhere as a separate file and is thus external to the file containing enabling information, the (HyperCard) document. (See e.g., [Sadowski11] at 72-74]). HyperCard and Director discloses that the object has associated type information. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a

Claim Text from ’985 Patent	HyperCard and Director prior art
	HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name “Cement Column,” which is specified directly and is stored elsewhere as a separate file. In this case, the syntax "playMovie" is type information specifying that the object “Cement Column” is a Director XCMD for a Director movie object. (See e.g., [Sadowski11] at 72-74]).
985-1.g: utilizing the type information to identify and locate an executable application external to the file; and	HyperCard and Director discloses that the browser uses type information to identify and locate an executable application. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.”

Claim Text from ’985 Patent	HyperCard and Director prior art
	A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name “Cement Column,” which is specified directly and is stored elsewhere as a separate file. In this case, the syntax "playMovie" is type information specifying that the object “Cement Column” is a playMovie XCMD for a Director movie object. That type information is found in the embed text format. (See e.g., [Sadowski11] at 72-74]). The thus-located XCMD is a compiled executable application. HyperCard and Director discloses that the executable application is external to the file containing enabling information. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When

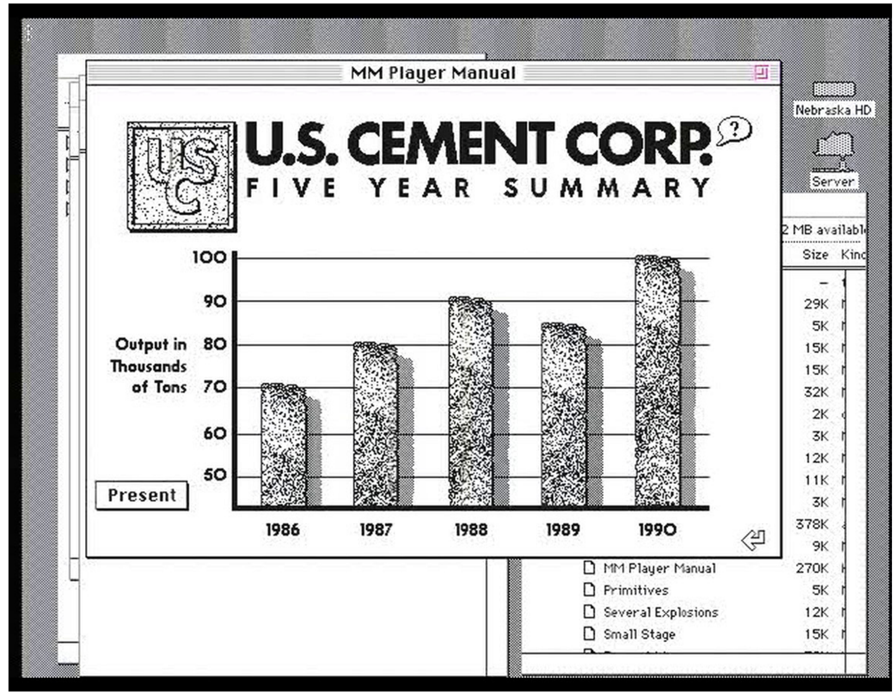
Claim Text from '985 Patent	HyperCard and Director prior art
	presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie "Cement Column" lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name "Cement Column," which is specified directly and is stored elsewhere as a separate file. In this case, the syntax "playMovie" is type information specifying that the object "Cement Column" is a playMovie XCMD for a Director movie object. That type information is found in the embed text format. (See e.g., [Sadowski11] at 72-74]). The thus-located XCMD is a compiled executable application which can be stored anywhere on a computer disk, external to the file containing enabling information.
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	HyperCard and Director discloses that the browser parses the embed text format. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file.

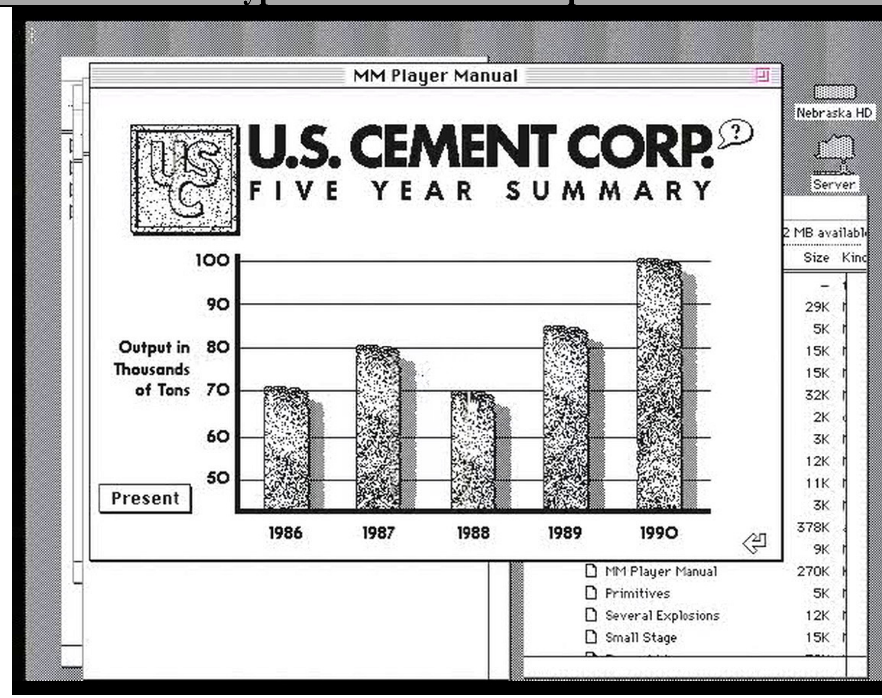
Claim Text from ’985 Patent	HyperCard and Director prior art
hypermedia document being displayed in the browser-controlled window.	Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The browser (HyperCard) parses the embed text format to cause the movie “Cement Column” to be played on the relevant card. (See e.g., [Sadowski11] at 72-74]). HyperCard and Director discloses automatic invocation of the executable application. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file.

Claim Text from ’985 Patent	HyperCard and Director prior art
	Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name Cement Column, which is specified directly and is stored elsewhere as a separate file. In this case, the syntax "playMovie" is type information specifying that the object “Cement Column” is a playMovie XCMD for a Director movie object. That type information is found in the embed text format. (See e.g., [Sadowski11] at 72-74]). The thus-located XCMD is a compiled executable application which can be stored anywhere on a computer disk. The executable application, XCMD, displays the object, “Cement Column.” HyperCard provided for automatic invocation of executable applications, without interactive action by the user, such as in the on presentMovie - end presentMovie script above.

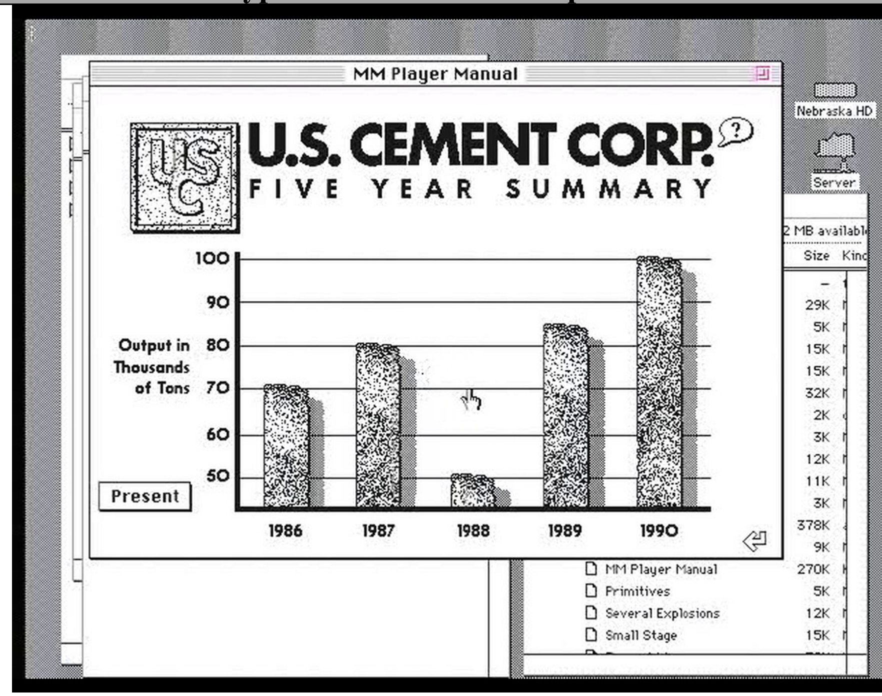
Claim Text from ’985 Patent	HyperCard and Director prior art
	HyperCard and Director discloses that the executable application displays the object. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name “Cement Column,” which is specified directly and is stored elsewhere as a separate file. In this case, the syntax "playMovie" is type information specifying that the object

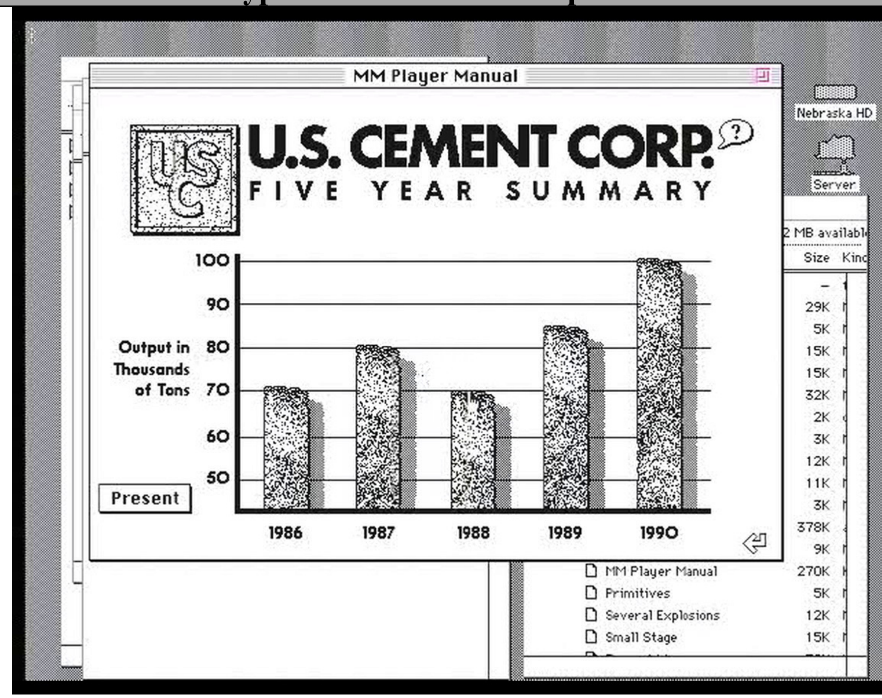
Claim Text from '985 Patent	HyperCard and Director prior art
	"Cement Column" is a playMovie XCMD for a Director movie object. That type information is found in the embed text format. (See e.g., [Sadowski11] at 72-74]). The thus-located XCMD is a compiled executable application which can be stored anywhere on a computer disk. The executable application, XCMD, displays the object, "Cement Column." HyperCard and Director discloses that the executable application enables direct interaction with the object. *See, e.g.*, : The Director Runtime Library enables direct interaction with the object. (See [Sadowski11] at 28, 42, 46, 65-70.) By way of example, the Director Runtime Library plays back interactive Director animations. The videos I am submitting with this report show a HyperCard application with text and graphics, and a portion of the screen reserved for the Director Runtime Library. By way of example, the screenshots below show the playback of interactive Director animations within a HyperCard application. The user interacts with this animation by clicking the mouse on the bar chart. Based on the location of the click, the bar chart changes its size. In the example below, the size of the 1988 chart is changed from 90 tons to 70 tons through a user's mouse click.

Claim Text from '985 Patent	HyperCard and Director prior art
	

Claim Text from '985 Patent	HyperCard and Director prior art
	 HyperCard and Director discloses that interaction with the object is at a first location in the hypermedia document. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is

Claim Text from ’985 Patent	HyperCard and Director prior art
	dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a movie, Cement Column, which is specified directly and is stored elsewhere as a separate file. The object is displayed in a location that corresponds to the first location (e.g. the segment of the stack corresponding to the card). Interaction with the object at the first location is provided by mouse clicks within the object which resizes the size of columns in the chart depending on the location of the mouse click. (See e.g., [Sadowski11] at 72-74]). The videos submitted with this report show examples of this interaction. The screenshots below exemplify the interaction:

Claim Text from ’985 Patent	HyperCard and Director prior art
	

Claim Text from '985 Patent	HyperCard and Director prior art
	
985-2.a: The method of claim 1 where: the information to enable comprises text formats.	HyperCard and Director discloses that the enabling information in the file is text formats. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is

Claim Text from '985 Patent	HyperCard and Director prior art
	dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of enabling information. The videos that I am submitting with this report show examples of HyperTalk scripts associated with hypermedia cards.
985-3.a: The method of claim 2 where the text formats are HTML tags.	HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." The text format tags used by HyperCard, while not HTML, are nonetheless tags that HyperCard can recognize to direct the way it lays out each card associated with each segment in the stack. HTML was known to the HyperCard developers but storing and reading binary data to and from the resource fork achieved an efficiency and speed not possible by parsing raw text. The videos that I am submitting with this report show examples of

Claim Text from '985 Patent	HyperCard and Director prior art
	HyperTalk scripts associated with hypermedia cards.
985-4.a: The method of claim 1 where the information contained in the file received comprises at least one embed text format.	HyperCard and Director discloses that the enabling information in the file includes an embed text format. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format. The videos that I am submitting with this report show examples of HyperTalk scripts associated with hypermedia cards.
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.	HyperCard and Director discloses that the embed text format is identified by parsing the file containing enabling information. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the

Claim Text from ’985 Patent	HyperCard and Director prior art
	stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of enabling information and is discovered by parsing a segment. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The videos that I am submitting with this report show examples of HyperTalk scripts associated with hypermedia cards.
985-6.a: The method of claim 5 where the parsing is by a parser in the browser.	HyperCard and Director discloses that the parser is in the browser *See, e.g.*, : HyperCard includes a parser. “Making cards is a procedure accomplished through the HyperTalk scripting language…” (See Goodman at p. 77.) HyperCard parses the stacks to display their contents to users. “… HyperCard is interpreting the handler while executing it.” (See Goodman at p. 336.)
985-7.a: The method of claim 1 where the processing	HyperCard and Director discloses that the text formats directly specify the processing. *See, e.g.*, :

Claim Text from ’985 Patent	HyperCard and Director prior art
specified by the text formats is specified directly.	HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of text format that directly specifies processing. The videos that I am submitting with this report show examples of HyperTalk scripts associated with hypermedia cards.
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.	HyperCard and Director discloses that the correspondence is implied by the order of text formats. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When

Claim Text from '985 Patent	HyperCard and Director prior art
	presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." Correspondence is implied by the order in which HyperCard parses the objects it finds in the segment that defines a card. This is generally in the order that was specified by the creator of the card. The videos that I am submitting with this report show examples of HyperTalk scripts associated with hypermedia cards.
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.	HyperCard and Director discloses that the embed text format specifies the location of the object directly. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound

Claim Text from ’985 Patent	HyperCard and Director prior art
	playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name “Cement Column,” which is specified directly and is stored elsewhere as a separate file. (See e.g., [Sadowski11] at 72-74]).
985-10.a: The method of claim 1 where having type information associated is by including type information in the embed text format.	HyperCard and Director discloses that the type information is in the embed text format. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound

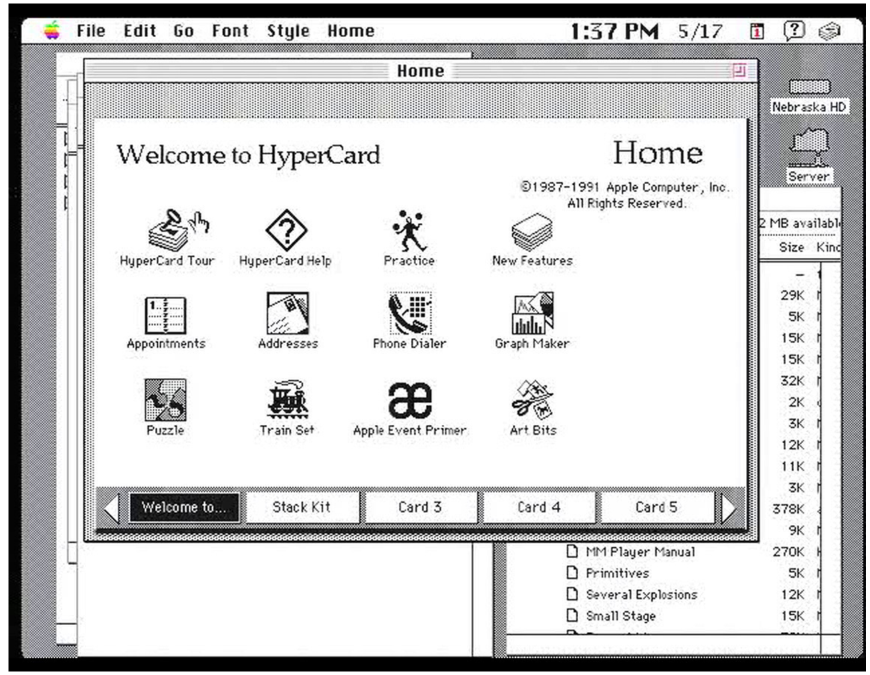
Claim Text from '985 Patent	HyperCard and Director prior art
	playMovie "Cement Column" lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name "Cement Column," which is specified directly and is stored elsewhere as a separate file. In this case, the syntax "PlayMovie" is type information specifying that the object "Cement Column" is a PlayMovie XCMD for a Director movie object. That type information is found in the embed text format. (See e.g., [Sadowski11] at 72-74]).
985-11.a: The method of claim 1 where automatically invoking does not require interactive action by the user.	HyperCard and Director discloses that automatic invocation does not require interactive action by the user. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a

Claim Text from ’985 Patent	HyperCard and Director prior art
	card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name “Cement Column,” which is specified directly and is stored elsewhere as a separate file. In this case, the syntax "playMovie" is type information specifying that the object “Cement Column” is a playMovie XCMD for a Director movie object. That type information is found in the embed text format. (See e.g., [Sadowski11] at 72-74]). The thus-located XCMD is a compiled executable application which can be stored anywhere on a computer disk. The executable application, XCMD, displays the object, “Cement Column.” HyperCard provided for automatic invocation of executable applications, without interactive action by the user, such as in the on presentMovie - end presentMovie script above.
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:	HyperCard and Director discloses computer code physically embodied on a medium. *See, e.g.*, : The computer on which HyperCard executes includes a computer usable media having computer readable program code physically embodied therein. As one example, the videos I am submitting with this report show HyperCard version 2.1 executing on Apple Macintosh IIsi computers running a System 7.1 operating system. (See also Goodman at pp. 17-20) (describing installation of HyperCard

<table>
<tr><th>Claim Text from ’985 Patent</th><th>HyperCard and Director prior art</th></tr>
<tr><td></td><td>

onto a computer.)

Below is a screenshot from my video of HyperCard 2.1:



HyperCard and Director discloses a client workstation and a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.

</td></tr>
<tr><td>

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

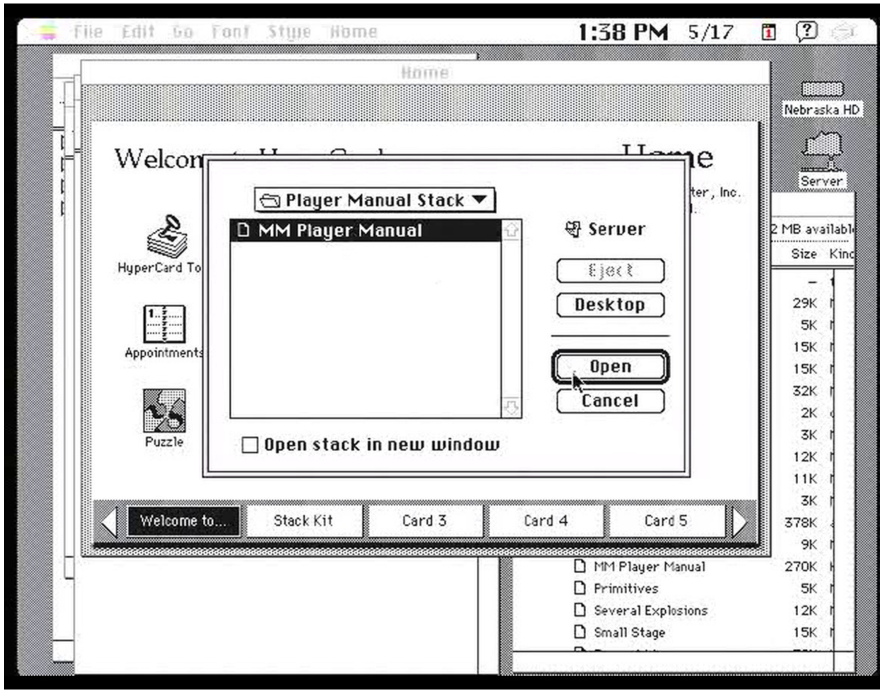
</td><td>

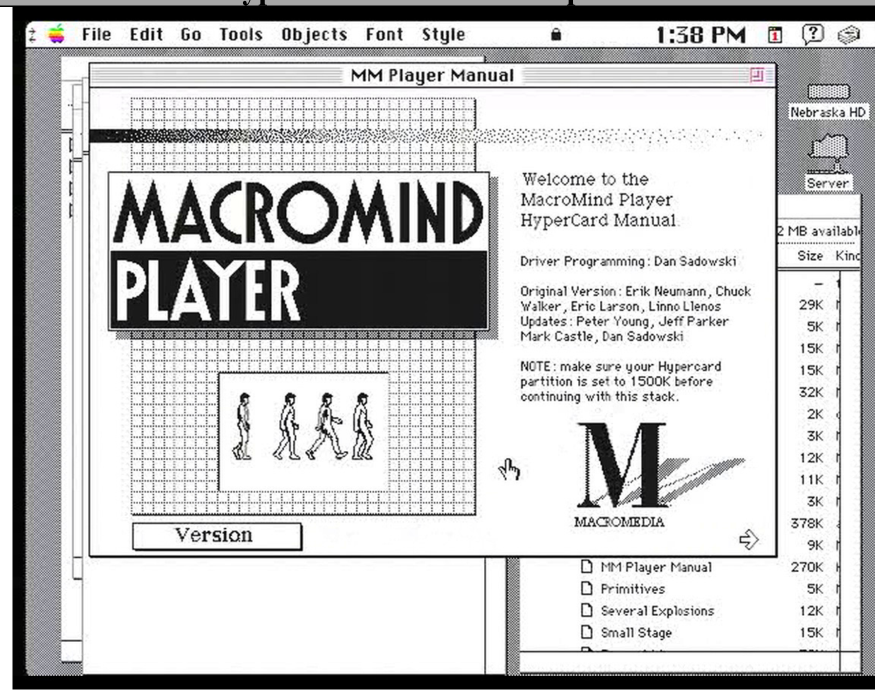
HyperCard and Director discloses a browser application; a file containing enabling information received from a server; that the browser displays at least a portion of a distributed hypermedia document; and that the display is in a browser-controlled window. *See* evidence recited for 985-1.b.

</td></tr>
</table>

Claim Text from ’985 Patent	HyperCard and Director prior art
browser-controlled window;	
985-16.c: cause the client workstation to utilize the browser to:	HyperCard and Director 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;	HyperCard and Director 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;	HyperCard and Director 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;	HyperCard and Director 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	HyperCard and Director 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: 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.	HyperCard and Director 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.
985-17.a: The computer readable media of claim 16 where:	HyperCard and Director discloses that the enabling information in the file is text formats. *See* evidence recited for 985-2.a.

Claim Text from ’985 Patent	HyperCard and Director prior art
the information to enable comprises text formats.	
985-18.a: The computer readable media of claim 17 where: the text formats are HTML tags.	The text format tags used by HyperCard, while not HTML, are nonetheless tags that HyperCard can recognize to direct the way it lays out each card associated with each segment in the stack. HTML was known to the HyperCard developers but storing and reading binary data to and from the resource fork achieved an efficiency and speed not possible by parsing raw text. *See* evidence recited 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.	HyperCard and Director 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:	HyperCard and Director discloses digital information. *See, e.g.*, : The information that is exchanged between a client workstation operating HyperCard and a network server in a Macintosh-based computer network environment is digital information. For example, a stack developed using HyperCard is stored as digital information. A stack can be stored on a network server and can be accessed by the client workstation using networking protocols that transmitted information in digital form. Examples include TOPS networks (described in Goodman at p. 738) or Ethertalk (shown in the videos I am submitting with this report). In addition, the videos I am submitting with this report make use digital information. In addition, screenshots from my video showing the digital information representing the HyperCard stack are shown below.

Claim Text from '985 Patent	HyperCard and Director prior art
	

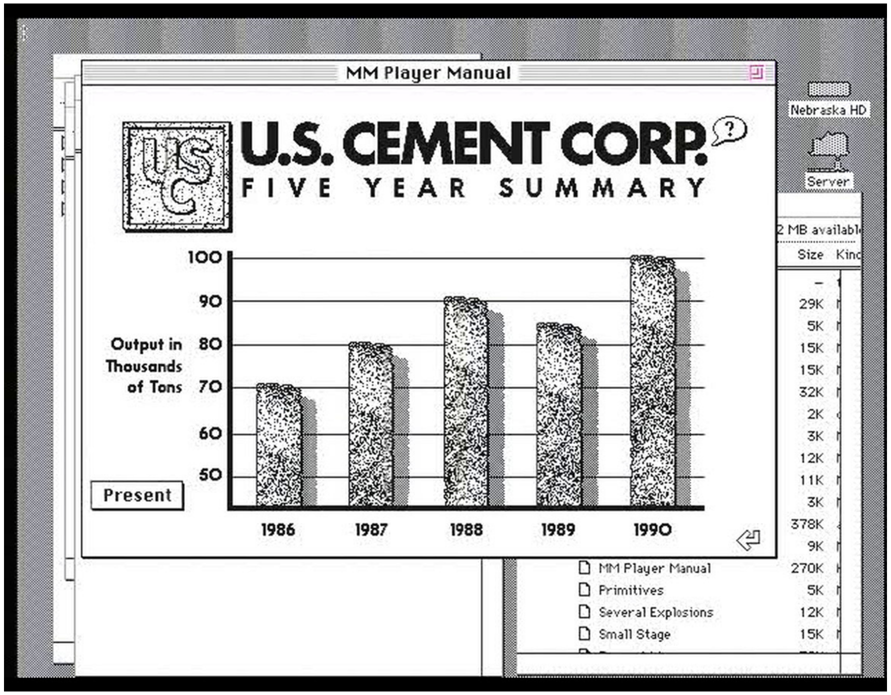
Claim Text from ’985 Patent	HyperCard and Director prior art
	 HyperCard and Director discloses a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.
985-20.b: communicating via the network server with at least one client workstation over said network in order to cause said client workstation to:	HyperCard and Director discloses a client workstation. *See* evidence recited for 985-1.a. HyperCard and Director discloses communicating via network server in order to cause the client workstation to act. *See, e.g.*, : > HyperCard interoperated with distributed applications. For example, HyperCard provided for XCMDs that interoperated with applications executing on other computers or network-connected devices. The applications executing on the other computers or network-connected

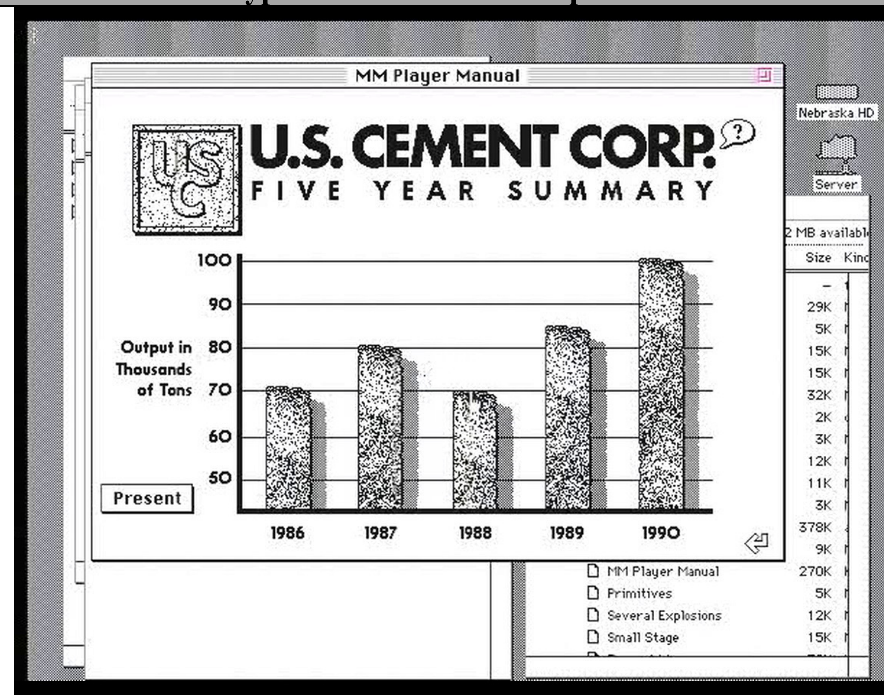
Claim Text from ’985 Patent	HyperCard and Director prior art
	devices communicated back to HyperCard operating in the client in order to cause the client to act. For example, HyperCard can connect to “any other computer (like a bulletin board service, MCI Mail, or Dow Jones News Retrieval) that offers asynchronous modem access” through HyperTalk script control. (See Goodman at pp. 725 – 726; see also [Sadowski11] at 54, 55, 60, 74, 76.) As another example, “HyperCard is also actively used in business as a tool to design what are known as 'front ends' to information stored on IBM (and other) mainframe computers.” (See Goodman at p. 726.) To accomplish the connections to an IBM mainframe computer, external commands (XCMDs), also called Application Programming Interfaces (APIs), that link HyperCard to a 3270-style terminal (a terminal that is used to connect to IBM mainframe computers) are added to HyperCard. These are typically supplied by the 3270-style hardware manufacturers. (See, e.g., Goodman at p. 727.) Concentrix Technology, Inc. designed front ends to IBM’s PROFS using Avatar, DCA and Tri-data APIs (XCMDs). IBM’s PROFS is an electronic mail and group scheduling program that runs on IBM mainframe computers. (See, e.g., Goodman at p. 727 – 728.) These front ends together with IBM’s PROFS constitute a distributed application. Another example is described in [Powers], in which the author developed software to enable a Macintosh to communicate with an IBM mainframe computer from within HyperCard. As another example, “HyperCard is also used extensively in business for accessing Structured Query Language (SQL) databases, usually running on mainframes or minicomputers (but also on database servers on local area networks).” (See Goodman at p. 728.) The databases, e.g., Oracle and Sybase, provide XCMD toolkits for HyperCard users to allow HyperCard stacks to access, retrieve and write data to the databases. “The HyperCard XCMDs extract the data, and regular HyperTalk scripting puts the data into fields or draws graphs based on that data.” See, e.g.,

Claim Text from ’985 Patent	HyperCard and Director prior art
	Goodman at pp. 727 – 728. The XCMD together with the application executing on the databases (connected through network) constitute a distributed application. As another example in which a HyperCard XCMD served as a front end to applications running on remote servers, [Morgan] discloses XCMDs that enable TCP-based client-server interactions. The XCMD together with the applications running on the remote servers constituted a distributed application. Morgan discloses two examples: Mini-atlas and Listmanager. “A connection is established using the TCPActiveOpen function, which establishes a connection with the remote socket (a connection between computer processors allowing them to communicate in a fast, reliable manner) and returns a connection ID.” (See Morgan at 421.) “Alternatively, TCPPassiveOpen will allow a connection to be accepted on a particular socket.” (See Morgan at 422.) “Mini-Atlas is a client for the Geographic Name Server. The Geographic Name Server contains brief information about most United States cities and geographic landmarks.” (See Morgan at 422.) “Another, more interesting application is the ListManager, a front end to LISTSERV programs operating electronic lists such as PACSL, AUTOCAT, and LIBREF-L. ListManager automates the procedures necessary to search the archives of these lists by keyword Boolean queries, tum off mail from the list temporarily, retrieve a list of the list's participants, or retrieve files from the lists.” (See Morgan at 423.)
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;	HyperCard and Director 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:	HyperCard and Director discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c.

Claim Text from ’985 Patent	HyperCard and Director prior art
985-20.e: responding to text formats to initiate processing specified by the text formats;	HyperCard and Director 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;	HyperCard and Director discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.
985-20.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 external to the file, where the object has type information associated with it;	HyperCard and Director 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.
985-20.h: utilizing the type information to identify and locate an executable application external to the file; and	HyperCard and Director 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.	HyperCard and Director 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.	HyperCard and Director discloses that the enabling information in the file is text formats. *See* evidence recited for 985-2.a.

Claim Text from ’985 Patent	HyperCard and Director prior art
985-22.a: The method of claim 21 where: the text formats are HTML tags.	The text format tags used by HyperCard, while not HTML, are nonetheless tags that HyperCard can recognize to direct the way it lays out each card associated with each segment in the stack. HTML was known to the HyperCard developers but storing and reading binary data to and from the resource fork achieved an efficiency and speed not possible by parsing raw text. *See* evidence recited 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.	HyperCard and Director discloses that the enabling information in the file includes an embed text format. *See* evidence recited for 985-4.a.
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:	HyperCard and Director discloses a client workstation and a network server in a network environment. *See* evidence recited for 985-1.a. HyperCard and Director 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,	HyperCard and Director discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e. HyperCard and Director discloses an object external to a file containing enabling information. *See* evidence recited for 985-1.f. HyperCard and Director discloses that there is enabling of an end-user to directly interact with the object. *See, e.g.*, : The Director Runtime Library enables an end-user to directly interact with the object. (See [Sadowski11] at 28, 42, 46, 65-70.) By way of example, the Director Runtime Library plays back interactive

Claim Text from ’985 Patent	HyperCard and Director prior art
	Director animations. The videos I am submitting with this report show a HyperCard application with text and graphics, and a portion of the screen reserved for the Director Runtime Library. For example, the screenshots below show the playback of interactive Director animations within a HyperCard application. The user interacts with this animation by clicking the mouse on the bar chart. Based on the location of the click, the bar chart changes its size. In the example below, the size of the 1988 chart is changed from 90 tons to 70 tons through a user’s mouse click. 

Claim Text from ’985 Patent	HyperCard and Director prior art
	 HyperCard and Director discloses that the interaction with the object is at a first location in a hypermedia document. *See* evidence recited for 985-1.h. HyperCard and Director discloses that the object is displayed at a first location within a portion of the hypermedia document being displayed. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the

Claim Text from '985 Patent	HyperCard and Director prior art
	stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie "Cement Column" lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name "Cement Column," which is specified directly and is stored elsewhere as a separate file. The object is displayed in a location that corresponds to the first location (e.g. the segment of the stack corresponding to the card). (See e.g., [Sadowski11] at 72-74]).
985-24.c: wherein said network environment is a distributed hypermedia environment,	HyperCard and Director discloses a client workstation and a network server in a distributed hypermedia environment. *See* evidence recited for 985-1.a.
985-24.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,	HyperCard and Director 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	HyperCard and Director prior art
at least said portion of said distributed hypermedia document within said browser-controlled window,	
985-24.e: wherein said executable application is external to said file,	HyperCard and Director 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,	HyperCard and Director discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c. HyperCard and Director 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,	HyperCard and Director 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,	HyperCard and Director 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,	HyperCard and Director 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,	HyperCard and Director 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	HyperCard and Director discloses using type information to identify and locate an executable application external to the file. *See* evidence recited for 985-1.g.
985-24.l: wherein the executable application is automatically invoked by the browser, in response to the identifying of the embed text format.	HyperCard and Director discloses automatically invoking the executable application. *See* evidence recited for 985-1.h.

Claim Text from '985 Patent	HyperCard and Director prior art
985-25.a: The method of claim 24 where: the information to enable comprises text formats.	HyperCard and Director 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.	The text format tags used by HyperCard, while not HTML, are nonetheless tags that HyperCard can recognize to direct the way it lays out each card associated with each segment in the stack. HTML was known to the HyperCard developers but storing and reading binary data to and from the resource fork achieved an efficiency and speed not possible by parsing raw text. *See* evidence recited 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.	HyperCard and Director 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:	HyperCard and Director discloses computer code physically embodied on a medium. *See* evidence recited for 985-16.a. HyperCard and Director discloses a client workstation and a network server in a network environment. *See* evidence recited for 985-1.a. HyperCard and Director 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	HyperCard and Director discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e. HyperCard and Director discloses an object external to a file containing enabling information. *See* evidence recited for 985-1.f.

Claim Text from ’985 Patent	HyperCard and Director prior art
displayed in a browser-controlled window,	HyperCard and Director discloses that there is enabling of an end-user to directly interact with the object. *See* evidence recited for 985-24.b. HyperCard and Director discloses that the interaction with the object is at a first location in a hypermedia document. *See* evidence recited for 985-1.h. HyperCard and Director discloses that the object is displayed within a display area created at the first location.. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card

Claim Text from '985 Patent	HyperCard and Director prior art
	unlock screen with wipe right end presentMovie The object in this script is a file with the name "Cement Column," which is specified directly and is stored elsewhere as a separate file. The object is displayed in a location that corresponds to the first location (e.g. the segment of the stack corresponding to the card). (See e.g., [Sadowski11] at 72-74]).
985-28.c: wherein said network environment is a distributed hypermedia environment,	HyperCard and Director 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,	HyperCard and Director 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,	HyperCard and Director 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,	HyperCard and Director discloses a browser application executing on the client workstation. *See* evidence recited for 985-1.c. HyperCard and Director discloses parsing text formats. *See* evidence recited for 985-1.d.
985-28.g: wherein at least said portion of the document is displayed within the browser-controlled window,	HyperCard and Director discloses displaying at least a portion of the document within the browser-controlled window. *See* evidence recited for 985-1.e.
985-28.h: wherein an embed text format which corresponds to said first location in the document is identified by the browser,	HyperCard and Director 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.

Claim Text from ’985 Patent	HyperCard and Director prior art
985-28.i: wherein the embed text format specifies the location of at least a portion of said object external to the file,	HyperCard and Director 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,	HyperCard and Director 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	HyperCard and Director 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.	HyperCard and Director 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:	HyperCard and Director 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 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;	HyperCard and Director 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 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.

Claim Text from '985 Patent	HyperCard and Director prior art
985-36.c: executing the browser application on the client workstation, with the browser application:	HyperCard and Director 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;	HyperCard and Director 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;	HyperCard and Director 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;	HyperCard and Director discloses an object. *See, e.g.*, : HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, "The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates." A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be: on presentMovie global returnSound playMovie "Cement Column" lock screen

Claim Text from ’985 Patent	HyperCard and Director prior art
	play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name “Cement Column,” which is specified directly and is stored elsewhere as a separate file. (See e.g., [Sadowski11] at 72-74]). HyperCard and Director 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	HyperCard and Director discloses that the browser identifies and locates an executable application associated with the object. *See, e.g.*, HyperCard used a platform-specific file format that was associated with the old Macintosh operating system. Files having that format were said to have two forks; a data fork and a resource fork. From [Sadowski11] at 53, resources such as code, interface item definitions, icons, script code or text were objects that were stored in the resource fork of an application file. Thus, HyperCard stacks were stored in such a file. On the file system the stack is organized into segments; one segment for each card. Again, from [Sadowski11] at 71, “The placement of the content for each card is dictated by coordinates for that content found in that segment. When presented to the HyperCard application, the content of the segment is parsed, the various content objects are located with their precise coordinates, and then they are placed on the display in accordance with the coordinates.” A HyperTalk script stored in the resource fork of a segment of a HyperCard document is one type of embed text format and is discovered at a first location when HyperCard parses the segment associated with a card. (See [Sadowski11] at 53]). Such a script could be:

Claim Text from ’985 Patent	HyperCard and Director prior art
	on presentMovie global returnSound playMovie “Cement Column” lock screen play returnSound pop card unlock screen with wipe right end presentMovie The object in this script is a file with the name “Cement Column,” which is specified directly and is stored elsewhere as a separate file. In this case, the syntax "playMovie" is type information specifying that the object “Cement Column” is a playMovie XCMD for a Director movie object. (See e.g., [Sadowski11] at 72-74]). That type information is found in the embed text format. The thus-located XCMD is a compiled 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,	HyperCard and Director discloses identifying an embed text format. *See* evidence recited in 985-1.f. HyperCard and Director 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. HyperCard and Director 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. HyperCard and Director discloses that a hypermedia document is displayed in a browser window. *See, e.g.*, evidence recited for 985-1.e.

Claim Text from ’985 Patent	HyperCard and Director prior art
985-36.i: wherein the executable application is part of a distributed application, and	HyperCard and Director discloses a distributed application. *See, e.g.*, : HyperCard interoperated with distributed applications. For example, HyperCard provided for XCMDs that interoperated with applications executing on other computers or network-connected devices. For example, HyperCard can connect to “any other computer (like a bulletin board service, MCI Mail, or Dow Jones News Retrieval) that offers asynchronous modem access” through HyperTalk script control. (See Goodman at pp. 725 – 726. see also [Sadowski11] at 54, 55, 60, 74, 76.) As another example, “HyperCard is also actively used in business as a tool to design what are known as 'front ends' to information stored on IBM (and other) mainframe computers.” (See Goodman at p. 726.) To accomplish the connections to an IBM mainframe computer, external commands (XCMDs), also called Application Programming Interfaces (APIs), that link HyperCard to a 3270-style terminal (a terminal that is used to connect to IBM mainframe computers) are added to HyperCard. These are typically supplied by the 3270-style hardware manufacturers. (See, e.g., Goodman at p. 727.) Concentrix Technology, Inc. designed front ends to IBM’s PROFS using Avatar, DCA and Tri-data APIs (XCMDs). IBM’s PROFS is an electronic mail and group scheduling program that runs on IBM mainframe computers. (See, e.g., Goodman at p. 727 – 728.) These front ends together with IBM’s PROFS constitute a distributed application. Another example is described in [Powers], in which the author developed software to enable a Macintosh to communicate with an IBM mainframe computer from within HyperCard. As another example, “HyperCard is also used extensively in business for accessing Structured Query Language (SQL) databases, usually running on mainframes or minicomputers (but also on database servers on local area networks).” (See Goodman at p. 728.) The databases, e.g., Oracle and Sybase, provide XCMD toolkits for HyperCard users to allow

Claim Text from '985 Patent	HyperCard and Director prior art
	HyperCard stacks to access, retrieve and write data to the databases. "The HyperCard XCMDs extract the data, and regular HyperTalk scripting puts the data into fields or draws graphs based on that data." See, e.g., Goodman at pp. 727 – 728. The XCMD together with the application executing on the databases (connected through network) constitute a distributed application. As another example in which a HyperCard XCMD served as a front end to applications running on remote servers, [Morgan] discloses XCMDs that enable TCP-based client-server interactions. The XCMD together with the applications running on the remote servers constituted a distributed application. Morgan discloses two examples: Mini-atlas and Listmanager. "A connection is established using the TCPActiveOpen function, which establishes a connection with the remote socket (a connection between computer processors allowing them to communicate in a fast, reliable manner) and returns a connection ID." (See Morgan at 421.) "Alternatively, TCPPassiveOpen will allow a connection to be accepted on a particular socket." (See Morgan at 422.) "Mini-Atlas is a client for the Geographic Name Server. The Geographic Name Server contains brief information about most United States cities and geographic landmarks." (See Morgan at 422.) "Another, more interesting application is the ListManager, a front end to LISTSERV programs operating electronic lists such as PACSL, AUTOCAT, and LIBREF-L. ListManager automates the procedures necessary to search the archives of these lists by keyword Boolean queries, tum off mail from the list temporarily, retrieve a list of the list's participants, or retrieve files from the lists." (See Morgan at 423.) HyperCard and Director discloses that the executable application is part of a distributed application. *See, e.g.*, : HyperCard interoperated with executable applications that were part of distributed applications. For example, HyperCard provided for XCMDs

Claim Text from ’985 Patent	HyperCard and Director prior art
	that interoperated with applications executing on other computers or network-connected devices. Additionally, HyperCard can connect to “any other computer (like a bulletin board service, MCI Mail, or Dow Jones News Retrieval) that offers asynchronous modem access” through HyperTalk script control. (See Goodman at pp. 725 – 726; see also [Sadowski11] at 54, 55, 60, 74, 76.) As another example, “HyperCard is also actively used in business as a tool to design what are known as 'front ends' to information stored on IBM (and other) mainframe computers.” (See Goodman at p. 726.) To accomplish the connections to an IBM mainframe computer, external commands (XCMDs), also called Application Programming Interfaces (APIs), that link HyperCard to a 3270-style terminal (a terminal that is used to connect to IBM mainframe computers) are added to HyperCard. These are typically supplied by the 3270-style hardware manufacturers. (See, e.g., Goodman at p. 727.) Concentrix Technology, Inc. designed front ends to IBM’s PROFS using Avatar, DCA and Tri-data APIs (XCMDs). IBM’s PROFS is an electronic mail and group scheduling program that runs on IBM mainframe computers. (See, e.g., Goodman at p. 727 – 728.) These front ends together with IBM’s PROFS constitute a distributed application. Another example is described in [Powers], in which the author developed software to enable a Macintosh to communicate with an IBM mainframe computer from within HyperCard. As another example, “HyperCard is also used extensively in business for accessing Structured Query Language (SQL) databases, usually running on mainframes or minicomputers (but also on database servers on local area networks).” (See Goodman at p. 728.) The databases, e.g., Oracle and Sybase, provide XCMD toolkits for HyperCard users to allow HyperCard stacks to access, retrieve and write data to the databases. “The HyperCard XCMDs extract the data, and regular HyperTalk scripting puts the data into fields or draws graphs based on that data.” See, e.g.,

Claim Text from ’985 Patent	HyperCard and Director prior art
	Goodman at pp. 727 – 728. The XCMD together with the application executing on the databases (connected through network) constitute a distributed application. As another example in which a HyperCard XCMD served as a front end to applications running on remote servers, [Morgan] discloses XCMDs that enable TCP-based client-server interactions. The XCMD together with the applications running on the remote servers constituted a distributed application. Morgan discloses two examples: Mini-atlas and Listmanager. “A connection is established using the TCPActiveOpen function, which establishes a connection with the remote socket (a connection between computer processors allowing them to communicate in a fast, reliable manner) and returns a connection ID.” (See Morgan at 421.) “Alternatively, TCPPassiveOpen will allow a connection to be accepted on a particular socket.” (See Morgan at 422.) “Mini-Atlas is a client for the Geographic Name Server. The Geographic Name Server contains brief information about most United States cities and geographic landmarks.” (See Morgan at 422.) “Another, more interesting application is the ListManager, a front end to LISTSERV programs operating electronic lists such as PACSL, AUTOCAT, and LIBREF-L. ListManager automates the procedures necessary to search the archives of these lists by keyword Boolean queries, tum off mail from the list temporarily, retrieve a list of the list's participants, or retrieve files from the lists.” (See Morgan at 423.)
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.	HyperCard and Director discloses that the distributed application executes at least partially on a network server. *See, e.g.*, : HyperCard interoperated with distributed applications executing at least partially on a server. For example, HyperCard provided for XCMDs that interoperated with applications executing on other computers or network-connected devices. For example, HyperCard can connect to “any other computer (like a bulletin board service, MCI Mail, or Dow Jones News Retrieval) that

Claim Text from ’985 Patent	HyperCard and Director prior art
	offers asynchronous modem access” through HyperTalk script control. (See Goodman at pp. 725 – 726; see also [Sadowski11] at 54, 55, 60, 74, 76.) As another example, “HyperCard is also actively used in business as a tool to design what are known as 'front ends' to information stored on IBM (and other) mainframe computers.” (See Goodman at p. 726.) To accomplish the connections to an IBM mainframe computer, external commands (XCMDs), also called Application Programming Interfaces (APIs), that link HyperCard to a 3270-style terminal (a terminal that is used to connect to IBM mainframe computers) are added to HyperCard. These are typically supplied by the 3270-style hardware manufacturers. (See, e.g., Goodman at p. 727.) Concentrix Technology, Inc. designed front ends to IBM’s PROFS using Avatar, DCA and Tri-data APIs (XCMDs). IBM’s PROFS is an electronic mail and group scheduling program that runs on IBM mainframe computers. (See, e.g., Goodman at p. 727 – 728.) These front ends together with IBM’s PROFS constitute a distributed application. Another example is described in [Powers], in which the author developed software to enable a Macintosh to communicate with an IBM mainframe computer from within HyperCard As another example, “HyperCard is also used extensively in business for accessing Structured Query Language (SQL) databases, usually running on mainframes or minicomputers (but also on database servers on local area networks).” (See Goodman at p. 728.) The databases, e.g., Oracle and Sybase, provide XCMD toolkits for HyperCard users to allow HyperCard stacks to access, retrieve and write data to the databases. “The HyperCard XCMDs extract the data, and regular HyperTalk scripting puts the data into fields or draws graphs based on that data.” See, e.g., Goodman at pp. 727 – 728. The XCMD together with the application executing on the databases (connected through network) constitute a distributed application. As another example in which a HyperCard XCMD served as a front end to

Claim Text from ’985 Patent	HyperCard and Director prior art
	applications running on remote servers, [Morgan] discloses XCMDs that enable TCP-based client-server interactions. The XCMD together with the applications running on the remote servers constituted a distributed application. Morgan discloses two examples: Mini-atlas and Listmanager. “A connection is established using the TCPActiveOpen function, which establishes a connection with the remote socket (a connection between computer processors allowing them to communicate in a fast, reliable manner) and returns a connection ID.” (See Morgan at 421.) “Alternatively, TCPPassiveOpen will allow a connection to be accepted on a particular socket.” (See Morgan at 422.) “Mini-Atlas is a client for the Geographic Name Server. The Geographic Name Server contains brief information about most United States cities and geographic landmarks.” (See Morgan at 422.) “Another, more interesting application is the ListManager, a front end to LISTSERV programs operating electronic lists such as PACSL, AUTOCAT, and LIBREF-L. ListManager automates the procedures necessary to search the archives of these lists by keyword Boolean queries, tum off mail from the list temporarily, retrieve a list of the list's participants, or retrieve files from the lists.” (See Morgan at 423.)
985-37.a: The method of claim 36 where: the information to enable comprises text formats.	HyperCard and Director 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.	The text format tags used by HyperCard, while not HTML, are nonetheless tags that HyperCard can recognize to direct the way it lays out each card associated with each segment in the stack. HTML was known to the HyperCard developers but storing and reading binary data to and from the resource fork achieved an efficiency and speed not possible by parsing raw text. *See* evidence recited for 985-3.a.

Claim Text from '985 Patent	HyperCard and Director prior art
985-39.a: The method of claim 36 where: the information contained in the file received comprises at least one embed text format.	HyperCard and Director 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:	HyperCard and Director discloses digital information. *See* evidence recited for 985-20.a. HyperCard and Director 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:	HyperCard and Director discloses a client workstation. *See* evidence recited for 985-1.a. HyperCard and Director 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 distributed hypermedia document within a browser-controlled window;	HyperCard and Director 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.
985-40.d: execute, at said client workstation, a browser application, with the browser application:	HyperCard and Director 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;	HyperCard and Director discloses parsing text formats. *See* evidence recited for 985-1.d.
985-40.f: displaying, on said client workstation, at least a	HyperCard and Director 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	HyperCard and Director prior art
portion of the document within the browser-controlled window;	
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;	HyperCard and Director discloses an object. *See* evidence recited for 985-36.f. HyperCard and Director 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	HyperCard and Director 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,	HyperCard and Director discloses identifying an embed text format. *See* evidence recited in 985-1.f. HyperCard and Director 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. HyperCard and Director 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. HyperCard and Director 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	HyperCard and Director discloses that the executable application is part of a distributed application. *See* evidence recited in 985-36.i.

Claim Text from ’985 Patent	HyperCard and Director prior art
985-40.k: wherein at least a portion of the distributed application is for execution on the network server.	HyperCard and Director 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.	HyperCard and Director 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.	The text format tags used by HyperCard, while not HTML, are nonetheless tags that HyperCard can recognize to direct the way it lays out each card associated with each segment in the stack. HTML was known to the HyperCard developers but storing and reading binary data to and from the resource fork achieved an efficiency and speed not possible by parsing raw text. *See* evidence recited 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.	HyperCard and Director discloses that the enabling information in the file includes an embed text format. *See* evidence recited for 985-4.a.