US006792575B1

(12) **United States Patent**
Samaniego et al.

(10) **Patent No.: US 6,792,575 B1**
(45) **Date of Patent: Sep. 14, 2004**

(54) **AUTOMATED PROCESSING AND DELIVERY OF MEDIA TO WEB SERVERS**

(75) Inventors: **Christopher Samaniego**, San Francisco, CA (US); **Nelson H. "Rocky" Offner**, Kensington, CA (US); **Adrian D. Thewlis**, Sausalito, CA (US); **David R. Boyd**, San Francisco, CA (US)

(73) Assignee: **Equilibrium Technologies**, Sausalito, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/425,326**

(22) Filed: **Oct. 21, 1999**

(51) **Int. Cl.**[7] **G06F 17/00**; G06F 15/00; G06F 9/30

(52) **U.S. Cl.** **715/513**; 715/501.1; 715/517; 345/735; 709/203; 709/219

(58) **Field of Search** 715/513, 517, 715/501.1; 707/102; 345/735, 800, 629; 709/203, 219

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,088,052 A		2/1992	Spielman et al.	395/158
5,355,472 A		10/1994	Lewis	395/600
5,530,852 A		6/1996	Meske, Jr. et al.	395/600
5,701,451 A		12/1997	Rogers et al.	395/600
5,708,845 A		1/1998	Wistendahl et al.	395/806
5,710,918 A		1/1998	Lagarde et al.	395/610
5,737,619 A		4/1998	Judson	395/761
5,745,908 A		4/1998	Anderson et al.	707/513
5,793,964 A		8/1998	Rogers et al.	395/200.32
5,822,436 A		10/1998	Rhoads	380/54
5,845,084 A		12/1998	Cordell et al.	395/200.64
5,845,299 A		12/1998	Arora et al.	707/513
5,860,068 A		1/1999	Cook	705/26
5,860,073 A		1/1999	Ferrel et al.	707/522
5,861,881 A		1/1999	Freeman et al.	345/302
5,862,325 A		1/1999	Reed et al.	395/200.31
5,870,552 A	*	2/1999	Dozier et al.	709/219
5,880,740 A	*	3/1999	Halliday et al.	345/629

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	0747842 A1	12/1996	G06F/17/30
EP	782085	7/1997	G06F/17/30
EP	818907	1/1998	H04L/29/06
EP	0843276 A1	5/1998	G06K/9/20
EP	876034	11/1998	H04L/29/06
EP	833068	12/1998	G06F/17/30
EP	886409	12/1998	H04L/29/06
EP	895171	2/1999	G06F/17/60

OTHER PUBLICATIONS

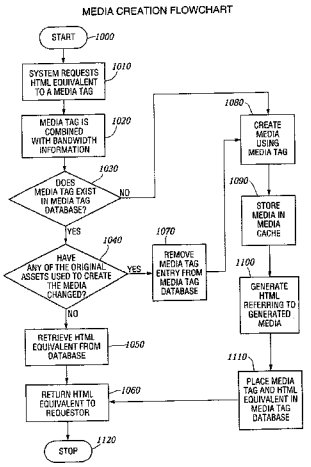
Sakaguchi et al., A browsing tool for multi–lingual documents for users without multi–lingual fonts, ACM International Conference On Digital Libraries, 1996, pp. 63–71.*

Primary Examiner—Sanjiv Shah
Assistant Examiner—William L. Bashore
(74) *Attorney, Agent, or Firm*—Glenn Patent Group; Michael A. Glenn

(57) **ABSTRACT**

A system using as input original media, an HTML document or browser language having proprietary tags, Web server traffic, and Web-client capabilities to generate an optimized Web media and HTML to refer to the generate media, and to automatically deploy the HTML and media to the Web server is provided. A Web authoring process is provided for facilitating creation of the media, assignment of a unique name to the media, and modification of the HTML document or browser language to contain a proprietary tag. Viewing capability is provided by the Web server passing the HTML or browser language, client browser capabilities, and current server traffic to the system, which parses the HTML or browser language searching for the proprietary tags. If a proprietary tag is found, the tag is processed to generate the Web media. Information is stored in the system database in case identical proprietary tags are processed.

9 Claims, 16 Drawing Sheets



U.S. PATENT DOCUMENTS

5,890,170 A * 3/1999 Sidana 715/501.1
5,895,476 A * 4/1999 Orr et al. 715/517
5,937,160 A * 8/1999 Davis et al. 709/203
6,009,436 A * 12/1999 Motoyama et al. 707/102
6,456,305 B1 * 9/2002 Qureshi et al. 345/800
6,563,517 B1 * 5/2003 Bhagwat et al. 345/735
6,591,280 B2 * 7/2003 Orr 715/513
6,623,529 B1 * 9/2003 Lakritz 715/536

* cited by examiner

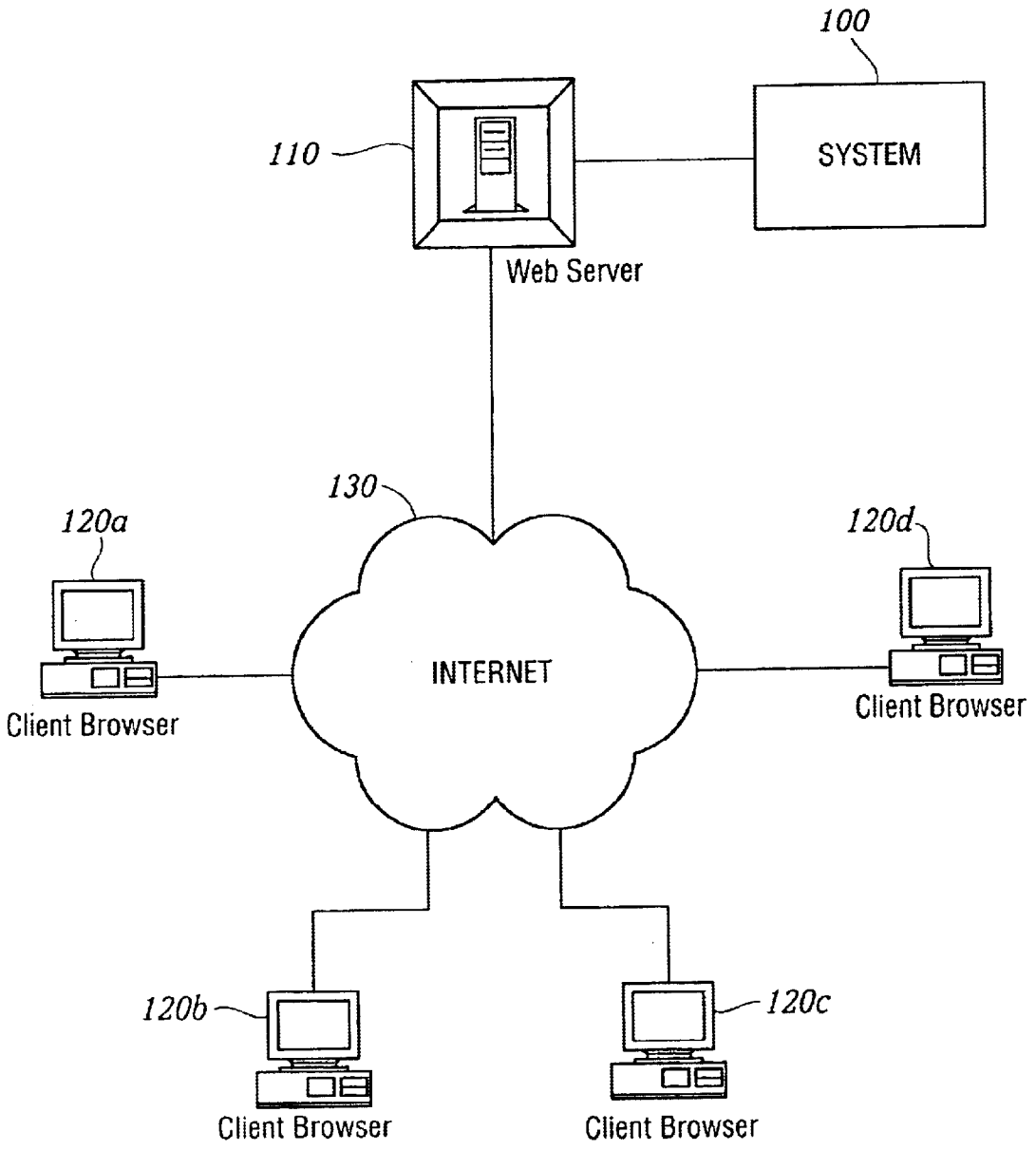

100
110
SYSTEM
Web Server
130
120a
120d
INTERNET
Client Browser
Client Browser
120b
120c
Client Browser
Client Browser

FIG. 1

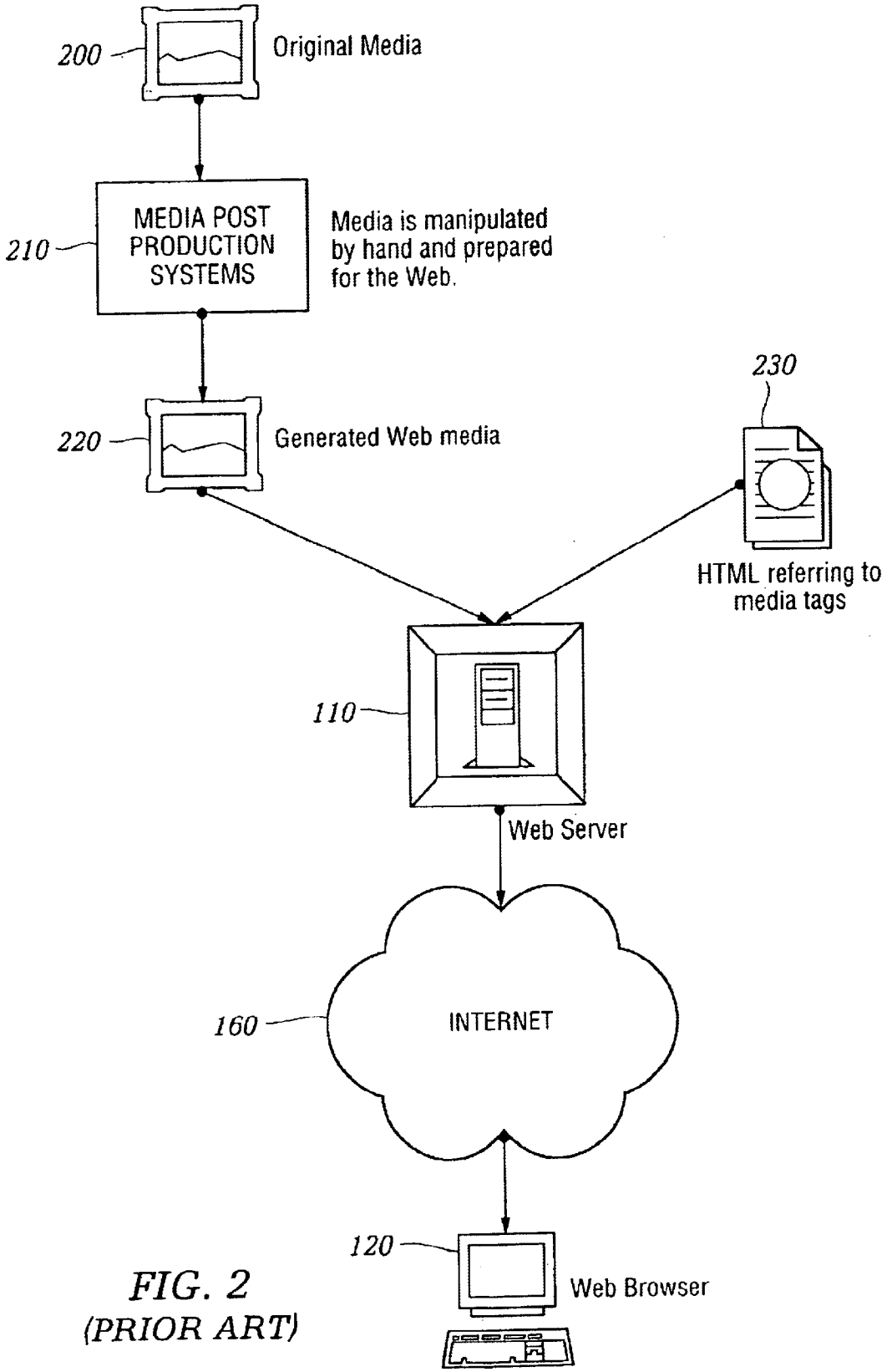


FIG. 2
(PRIOR ART)

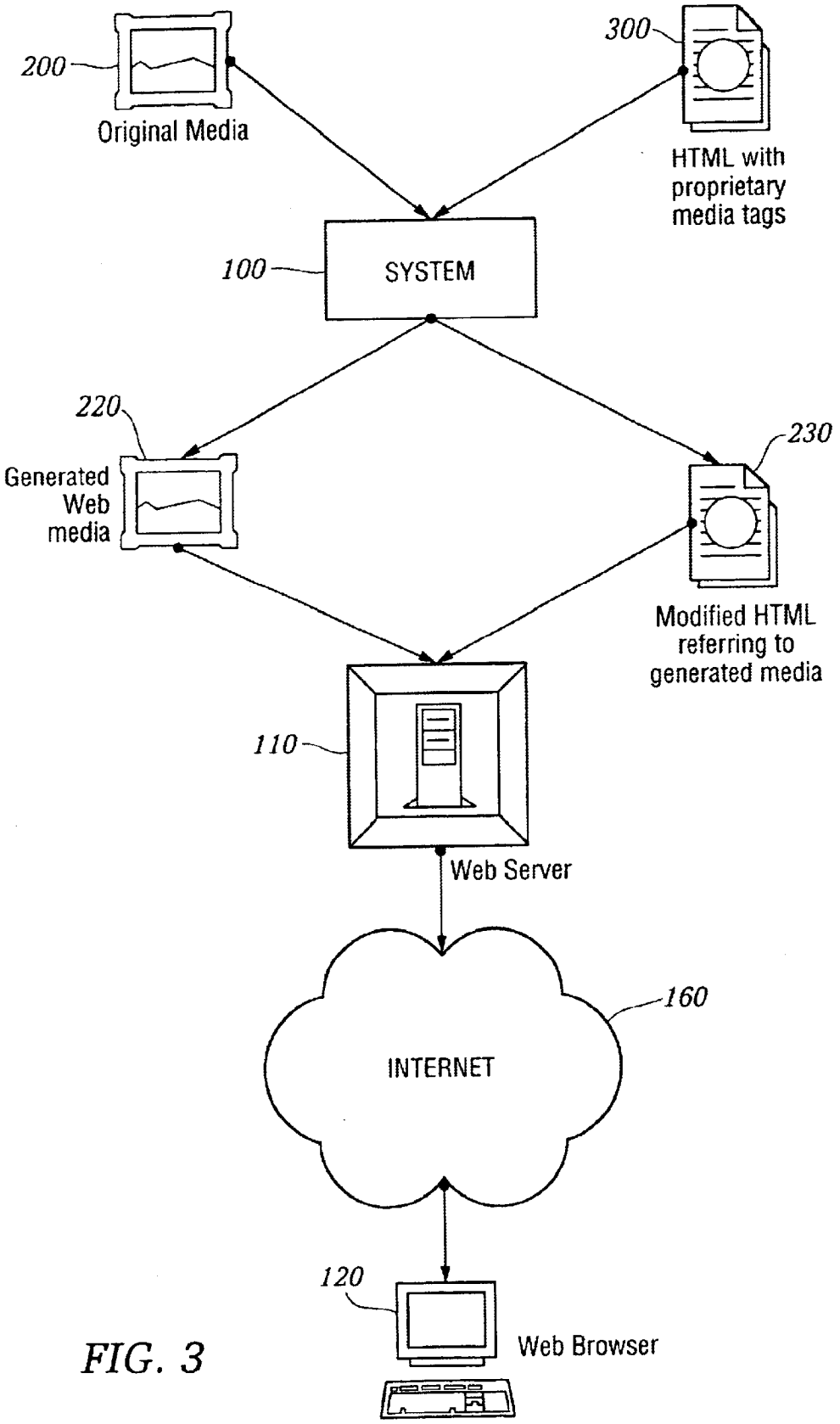

200
Original Media
300
HTML with proprietary media tags
100
SYSTEM
220
Generated Web media
230
Modified HTML referring to generated media
110
Web Server
160
INTERNET
120
Web Browser

FIG. 3

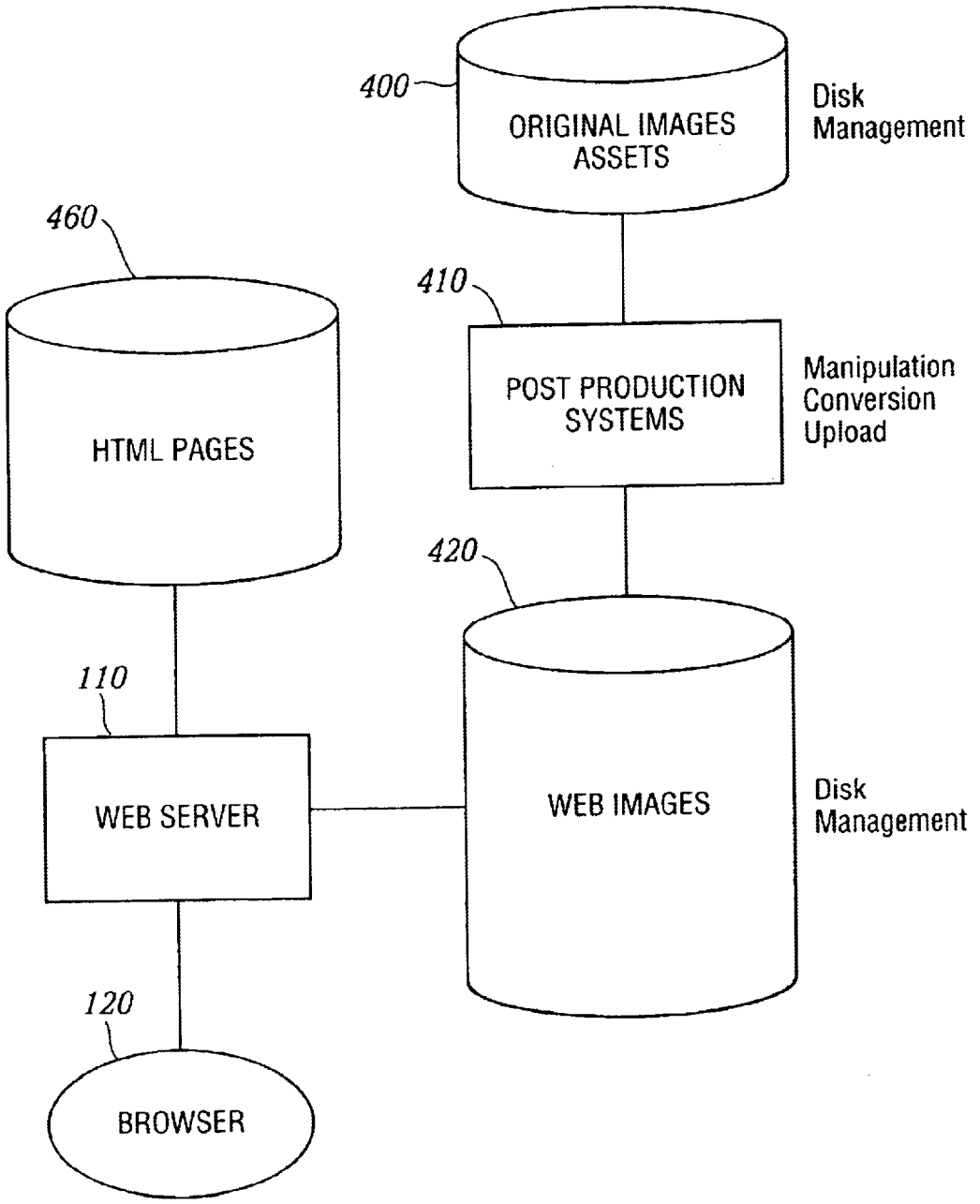


FIG. 4
(PRIOR ART)

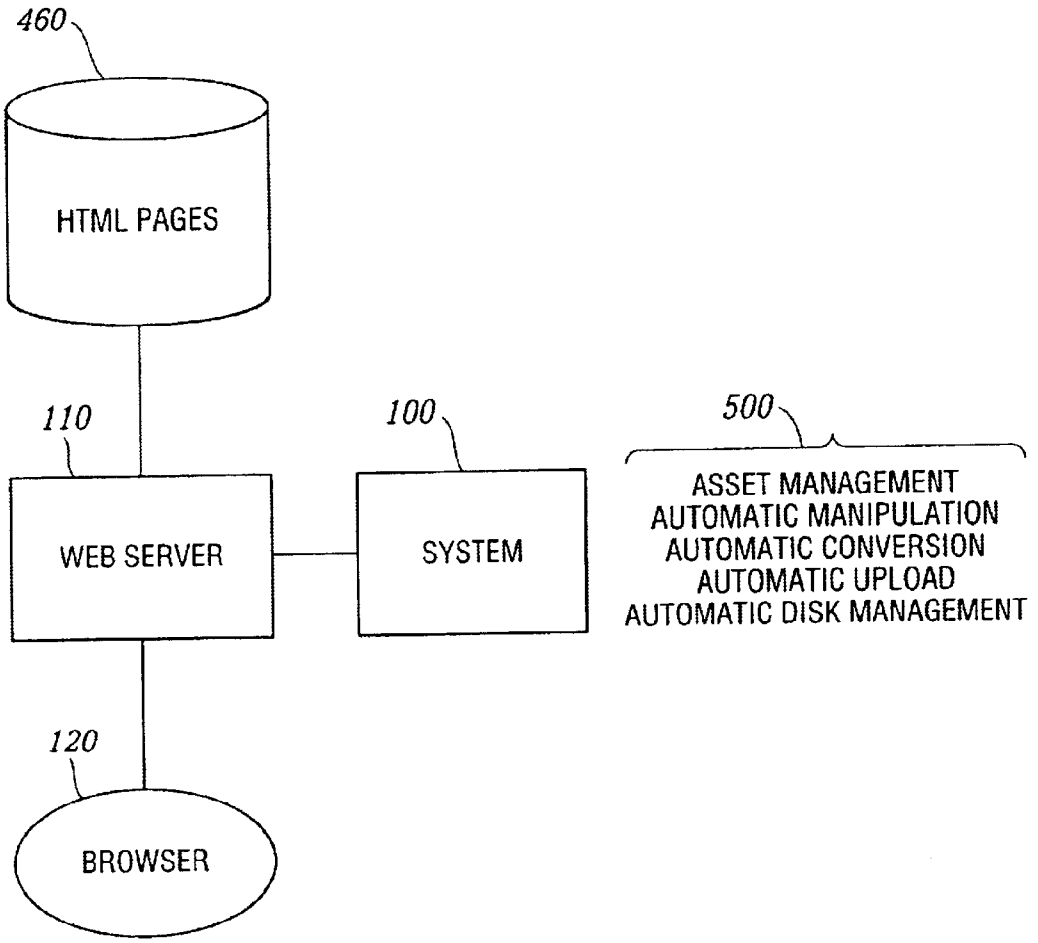

460
HTML PAGES
110
WEB SERVER
100
SYSTEM
500
ASSET MANAGEMENT
AUTOMATIC MANIPULATION
AUTOMATIC CONVERSION
AUTOMATIC UPLOAD
AUTOMATIC DISK MANAGEMENT
120
BROWSER

FIG. 5

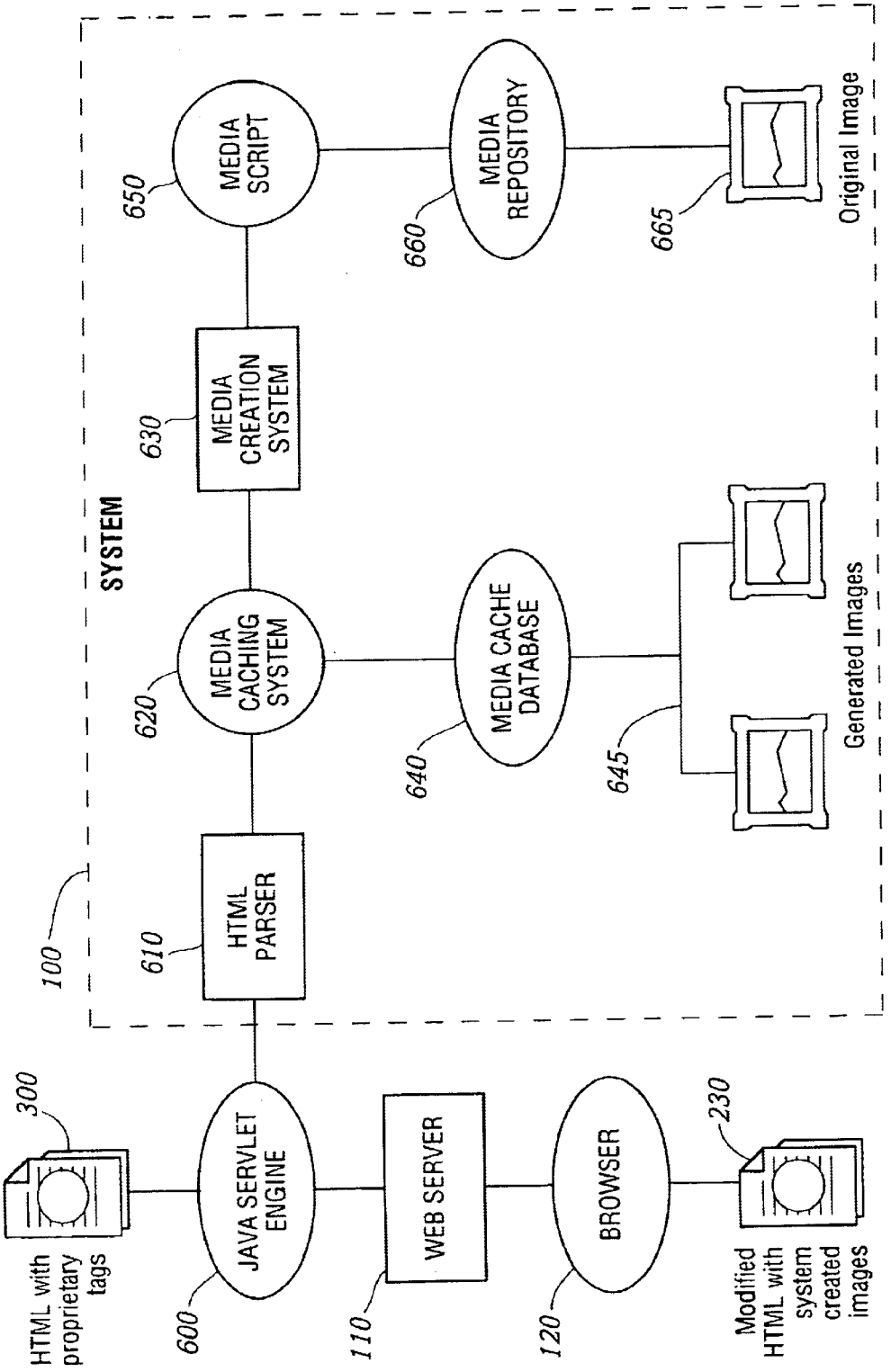

HTML with proprietary tags
300
JAVA SERVLET ENGINE
600
WEB SERVER
110
BROWSER
120
Modified HTML with system created images
230
100
SYSTEM
HTML PARSER
610
MEDIA CACHING SYSTEM
620
MEDIA CREATION SYSTEM
630
MEDIA SCRIPT
650
MEDIA CACHE DATABASE
640
MEDIA REPOSITORY
660
Generated Images
645
Original Image
665

FIG. 6

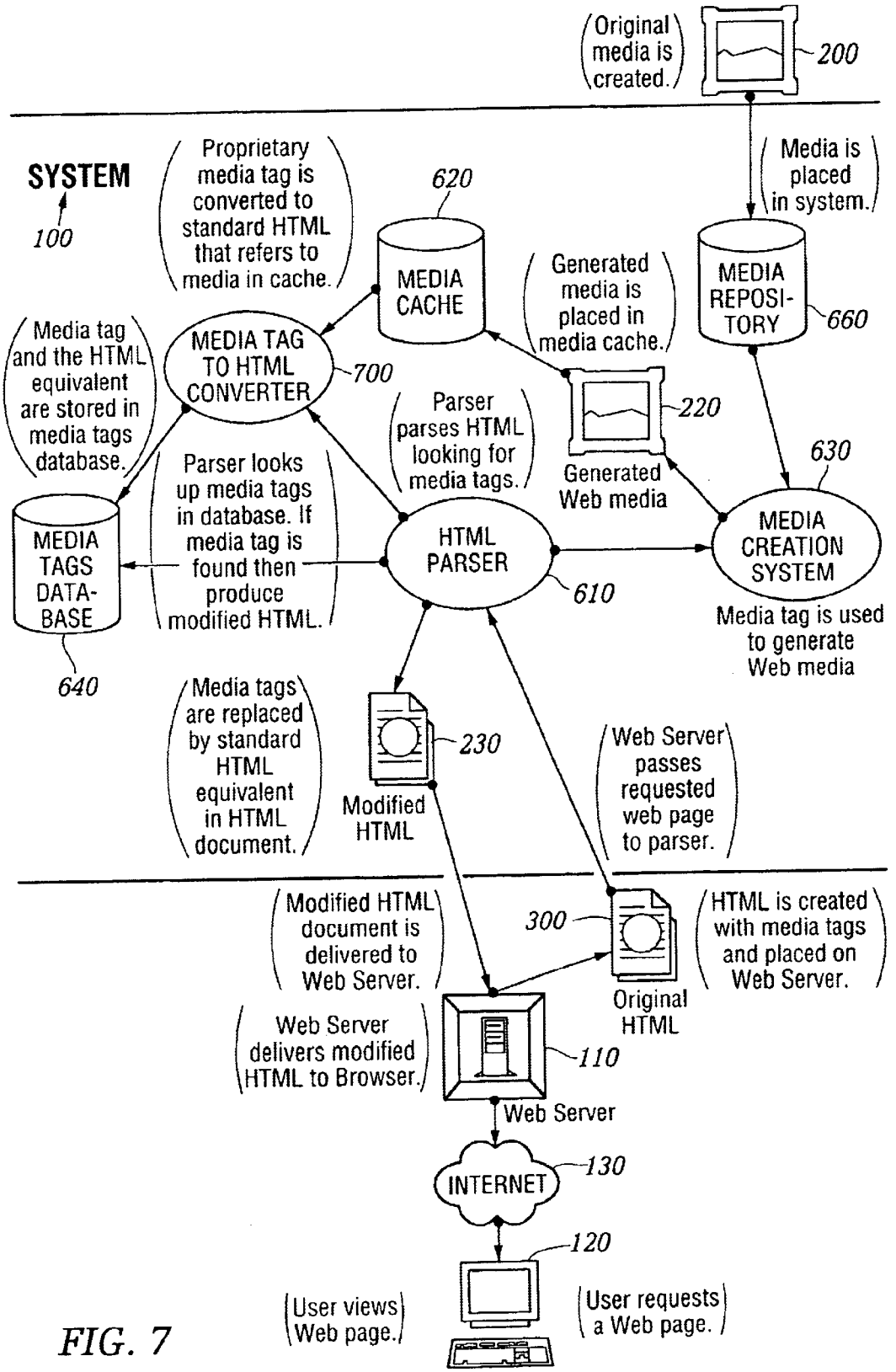

Original media is created.
200
SYSTEM
100
Proprietary media tag is converted to standard HTML that refers to media in cache.
620
MEDIA CACHE
Media is placed in system.
MEDIA REPOSI-TORY
660
Generated media is placed in media cache.
Media tag and the HTML equivalent are stored in media tags database.
MEDIA TAG TO HTML CONVERTER
700
Parser parses HTML looking for media tags.
220
Generated Web media
630
Parser looks up media tags in database. If media tag is found then produce modified HTML.
MEDIA TAGS DATA-BASE
640
HTML PARSER
610
MEDIA CREATION SYSTEM
Media tag is used to generate Web media
Media tags are replaced by standard HTML equivalent in HTML document.
230
Modified HTML
Web Server passes requested web page to parser.
Modified HTML document is delivered to Web Server.
300
Original HTML
HTML is created with media tags and placed on Web Server.
Web Server delivers modified HTML to Browser.
110
Web Server
INTERNET
130
120
User views Web page.
User requests a Web page.

FIG. 7

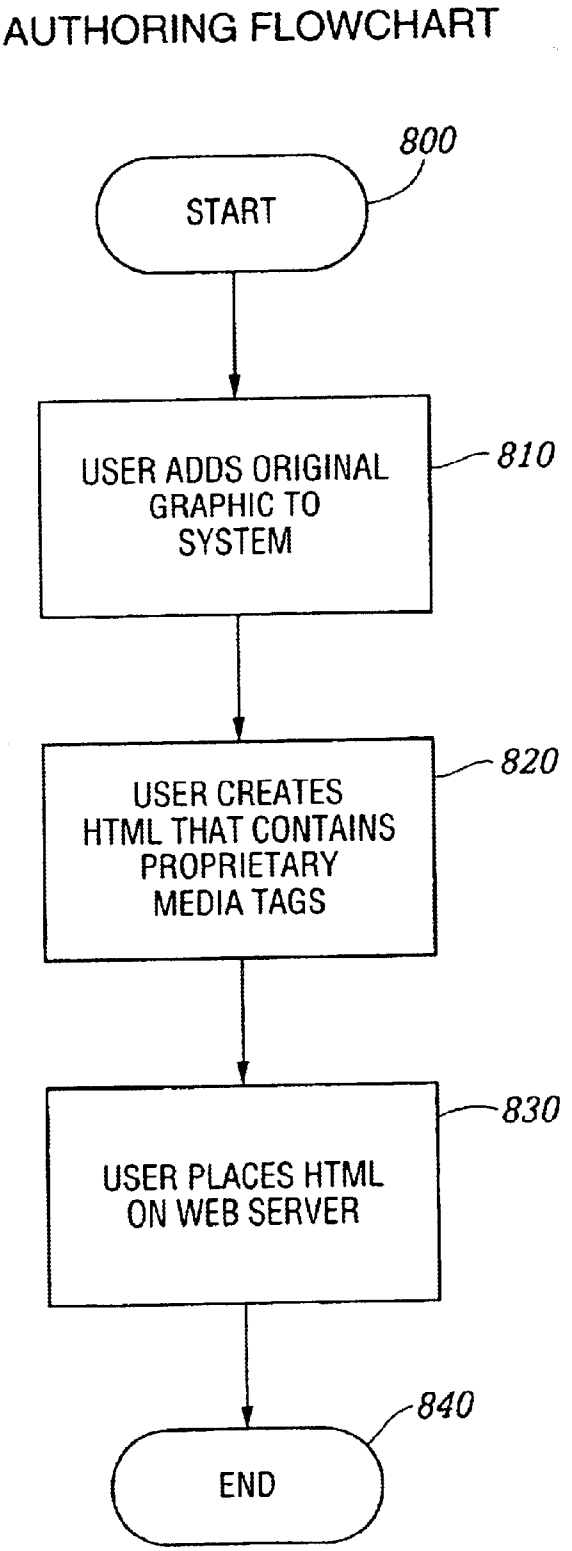

AUTHORING FLOWCHART
800
START
USER ADDS ORIGINAL GRAPHIC TO SYSTEM
810
USER CREATES HTML THAT CONTAINS PROPRIETARY MEDIA TAGS
820
USER PLACES HTML ON WEB SERVER
830
840
END

FIG. 8

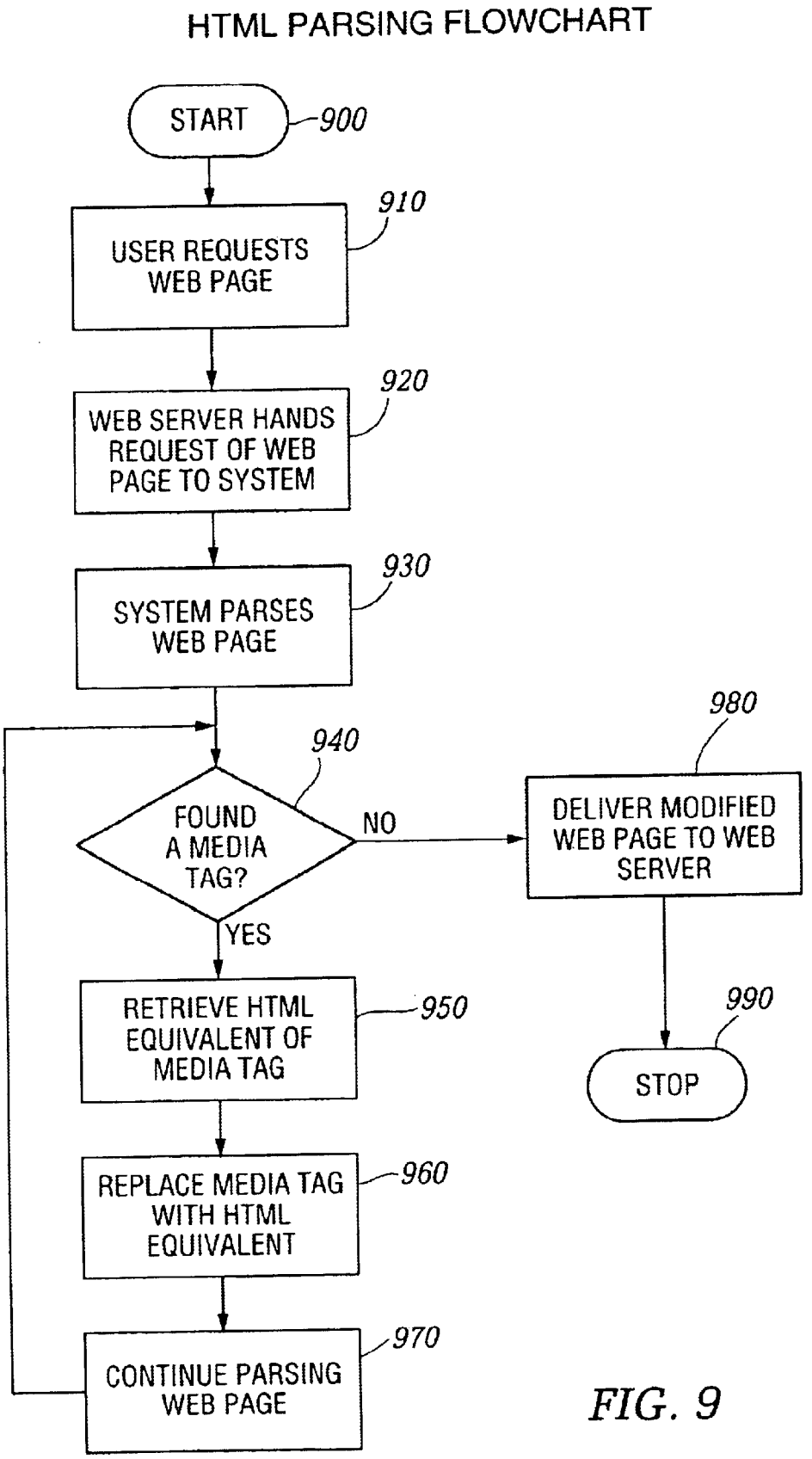

HTML PARSING FLOWCHART
START
900
USER REQUESTS WEB PAGE
910
WEB SERVER HANDS REQUEST OF WEB PAGE TO SYSTEM
920
SYSTEM PARSES WEB PAGE
930
FOUND A MEDIA TAG?
940
NO
YES
DELIVER MODIFIED WEB PAGE TO WEB SERVER
980
STOP
990
RETRIEVE HTML EQUIVALENT OF MEDIA TAG
950
REPLACE MEDIA TAG WITH HTML EQUIVALENT
960
CONTINUE PARSING WEB PAGE
970

FIG. 9

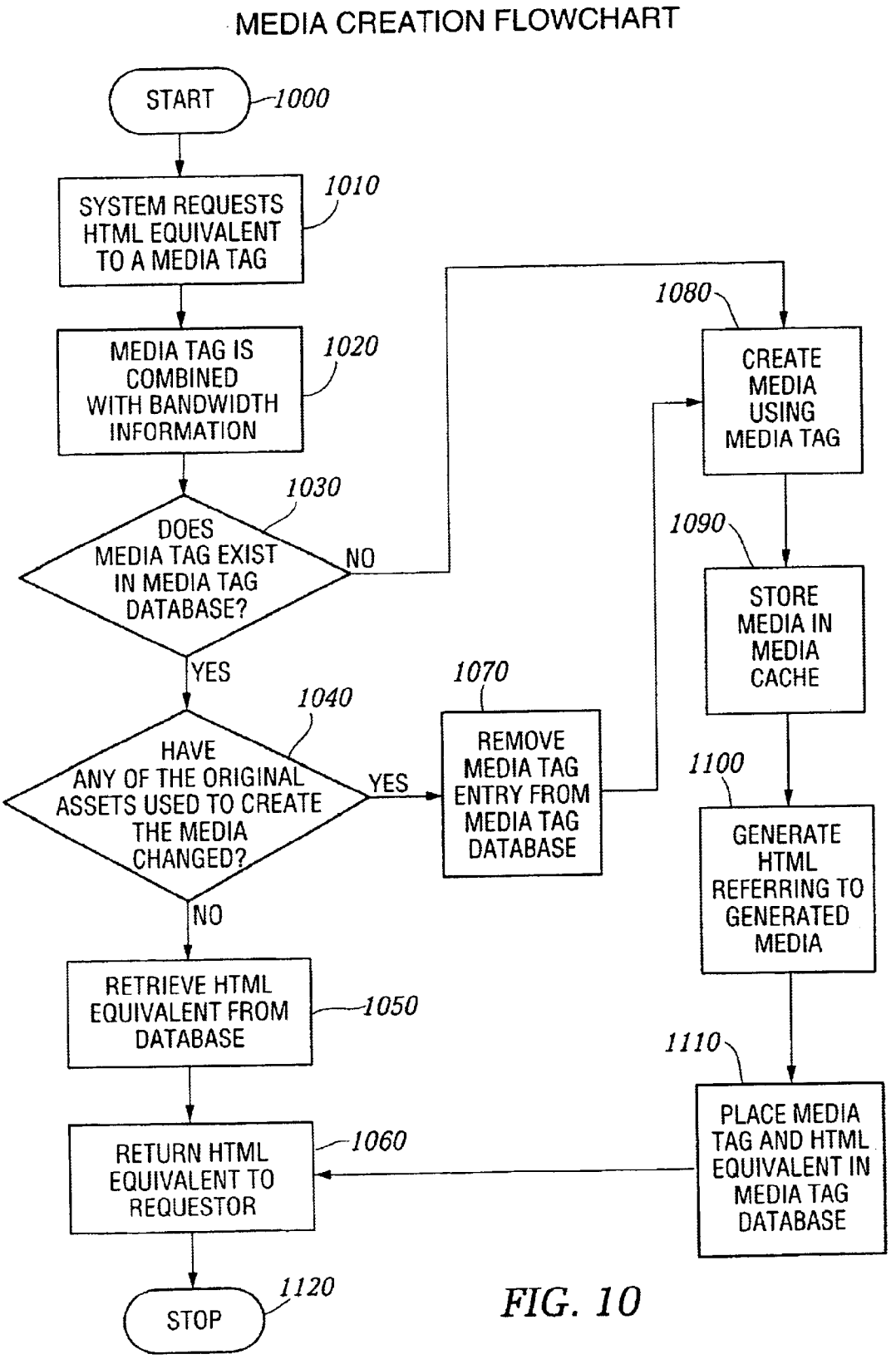

MEDIA CREATION FLOWCHART
START
1000
SYSTEM REQUESTS HTML EQUIVALENT TO A MEDIA TAG
1010
MEDIA TAG IS COMBINED WITH BANDWIDTH INFORMATION
1020
DOES MEDIA TAG EXIST IN MEDIA TAG DATABASE?
1030
NO
YES
HAVE ANY OF THE ORIGINAL ASSETS USED TO CREATE THE MEDIA CHANGED?
1040
YES
NO
REMOVE MEDIA TAG ENTRY FROM MEDIA TAG DATABASE
1070
RETRIEVE HTML EQUIVALENT FROM DATABASE
1050
RETURN HTML EQUIVALENT TO REQUESTOR
1060
STOP
1120
CREATE MEDIA USING MEDIA TAG
1080
STORE MEDIA IN MEDIA CACHE
1090
GENERATE HTML REFERRING TO GENERATED MEDIA
1100
PLACE MEDIA TAG AND HTML EQUIVALENT IN MEDIA TAG DATABASE
1110

FIG. 10

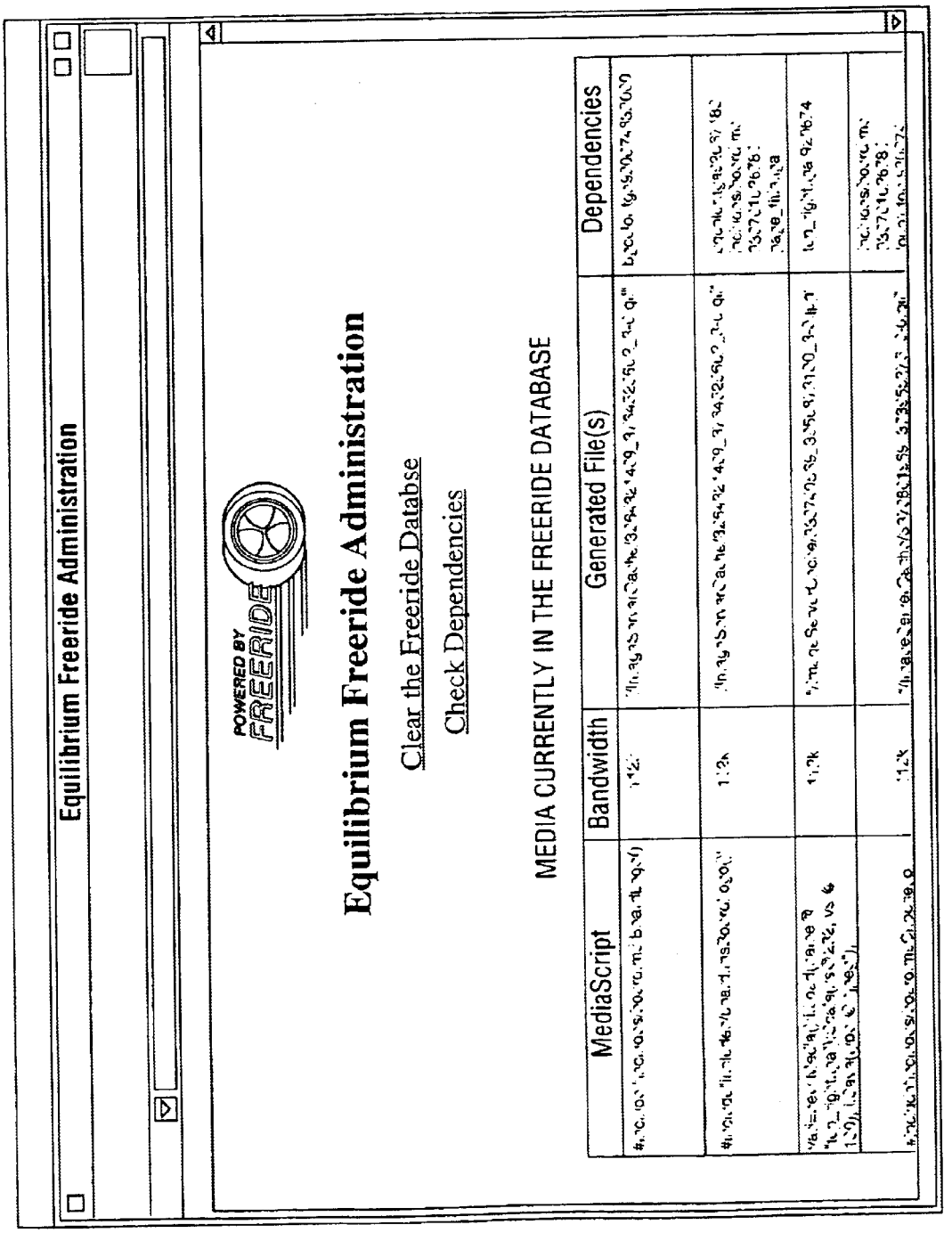

Equilibrium Freeride Administration
POWERED BY FREERIDE
Equilibrium Freeride Administration
Clear the Freeride Databse
Check Dependencies
MEDIA CURRENTLY IN THE FREERIDE DATABASE
MediaScript
Bandwidth
Generated File(s)
Dependencies

FIG. 11

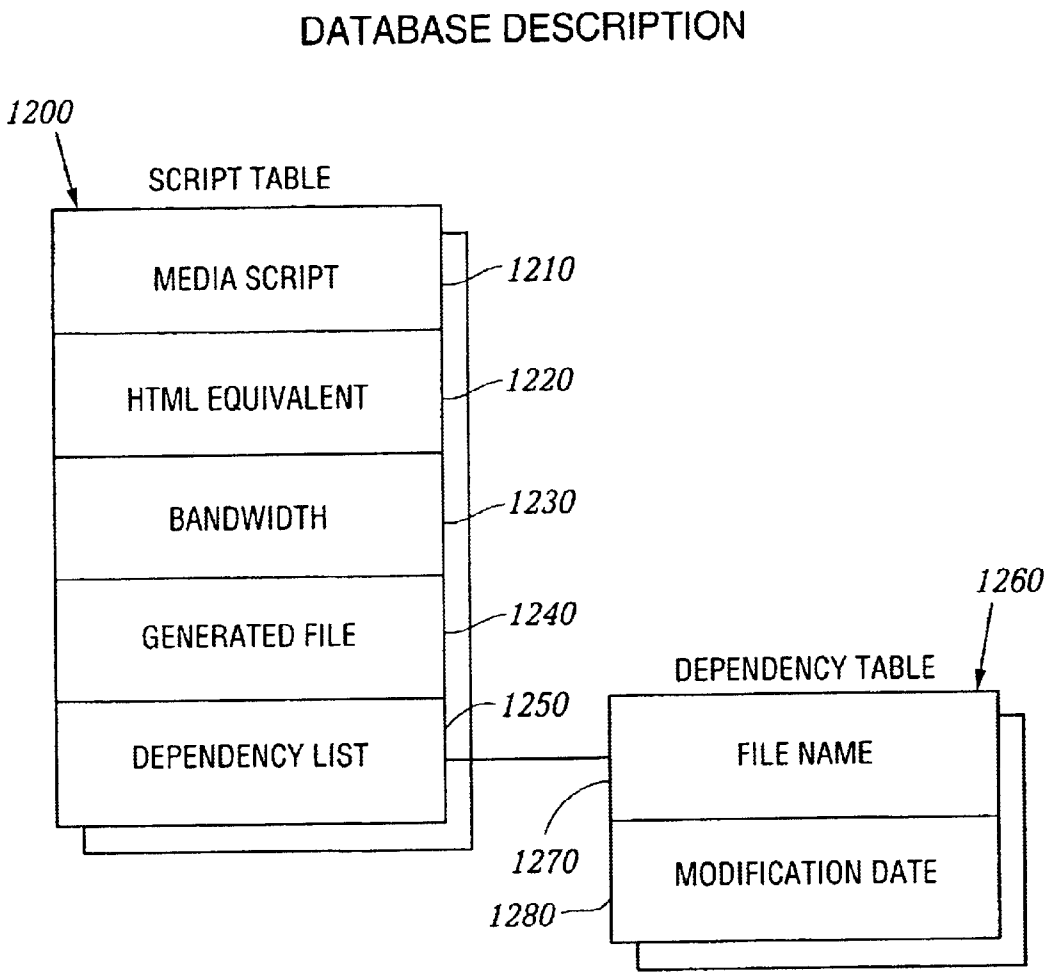

DATABASE DESCRIPTION
1200
SCRIPT TABLE
MEDIA SCRIPT
1210
HTML EQUIVALENT
1220
BANDWIDTH
1230
GENERATED FILE
1240
DEPENDENCY LIST
1250
1260
DEPENDENCY TABLE
FILE NAME
1270
MODIFICATION DATE
1280

FIG. 12

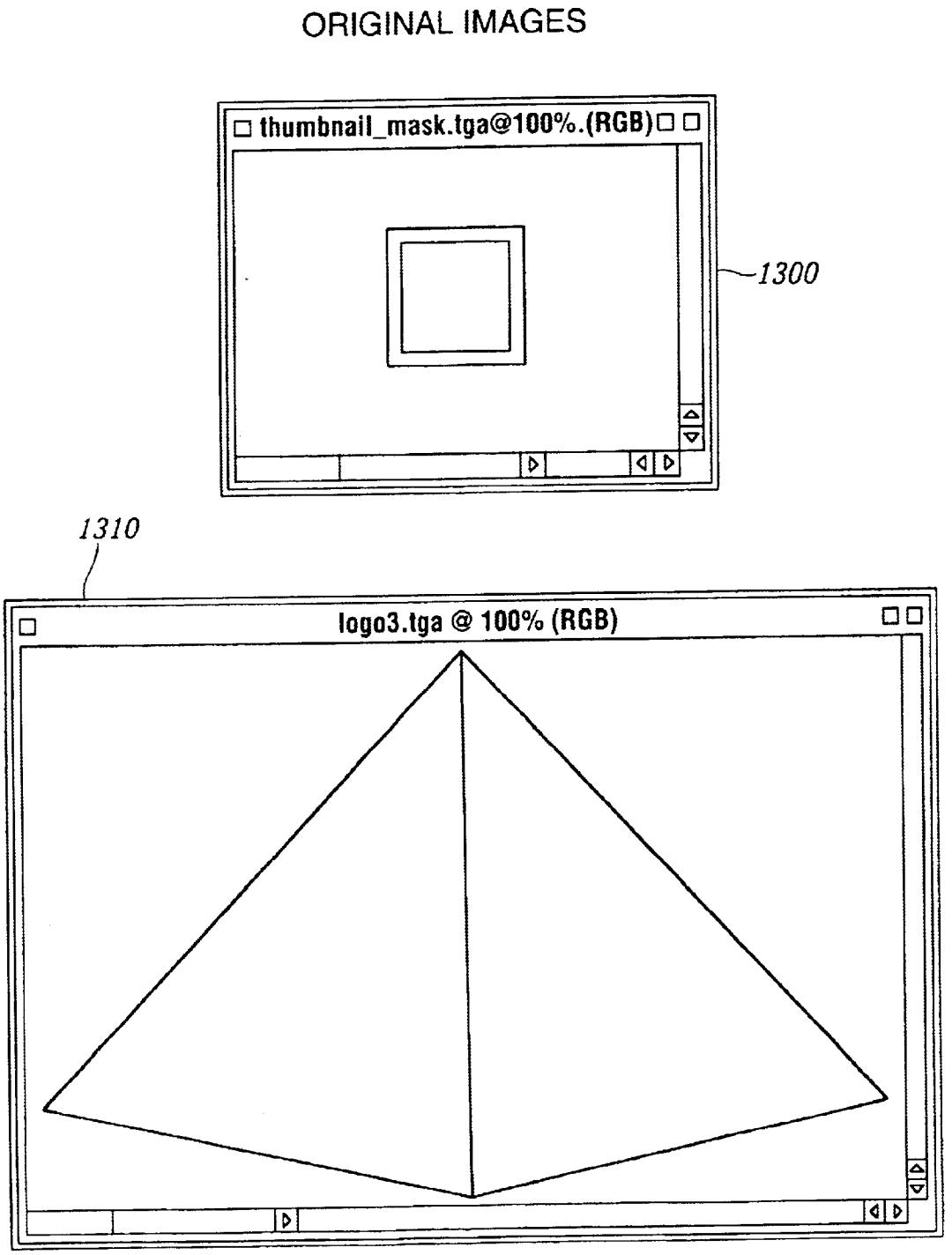

ORIGINAL IMAGES
thumbnail_mask.tga@100%.(RGB)
1300
1310
logo3.tga @ 100% (RGB)

FIG.13

HTML DOCUMENT WITH PROPRIETARY TAG

1400

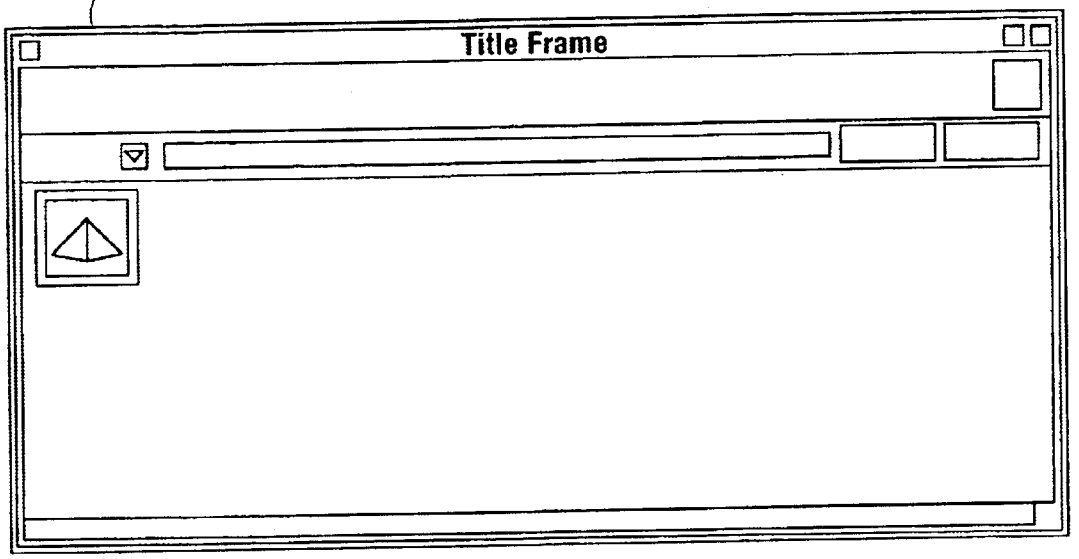
image.html

```
<html>
<head>
<title>
Title Frame
</title>3
</head>
<body>
<imgsrc= <freeride:image>var i = new Media(), i.Load(name @ logo3.tga);
i.Scale(xs @ 65, constrain @ true, alg @ "smooth"); i.Crop(xs@ 60,ys@60,
padcolor @ 0xffffff); var i2 = new Media(); i2.Load(name @
thumbnail_mask.tga), i.Composite(source @ i2); i.Scale(xs @ 60, ys@60,alg
@ "smooth"), i.Reduce(); iSave(type @ "gif"), </freeride:image>
height=60 width=60 border=0><br>
</body>
</html>
```

FIG.14

1500

HTML DOCUMENT VIEWED IN BROWSER



1510

HTML DOCUMENT SOURCE

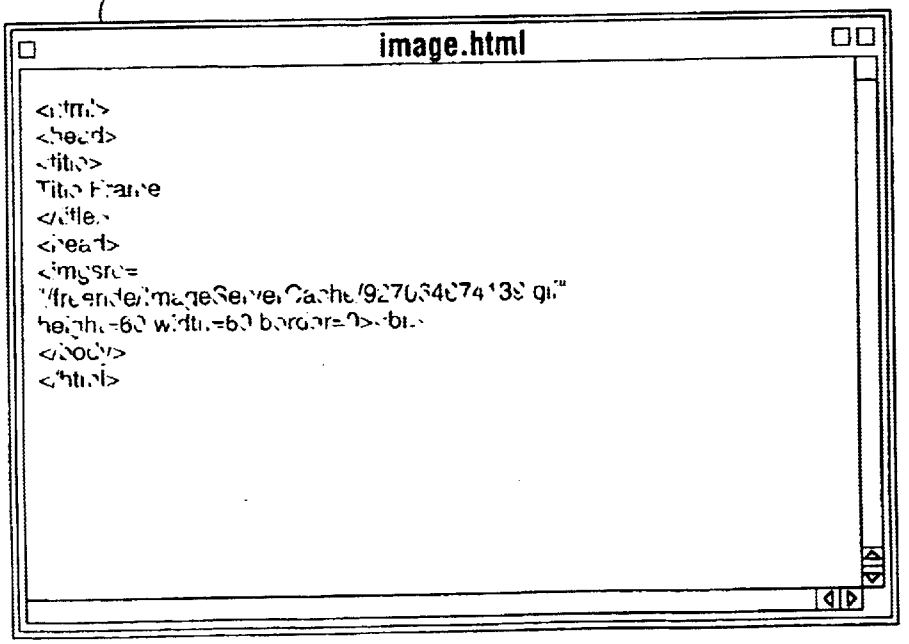


FIG.15

GENERATED GIF IMAGE

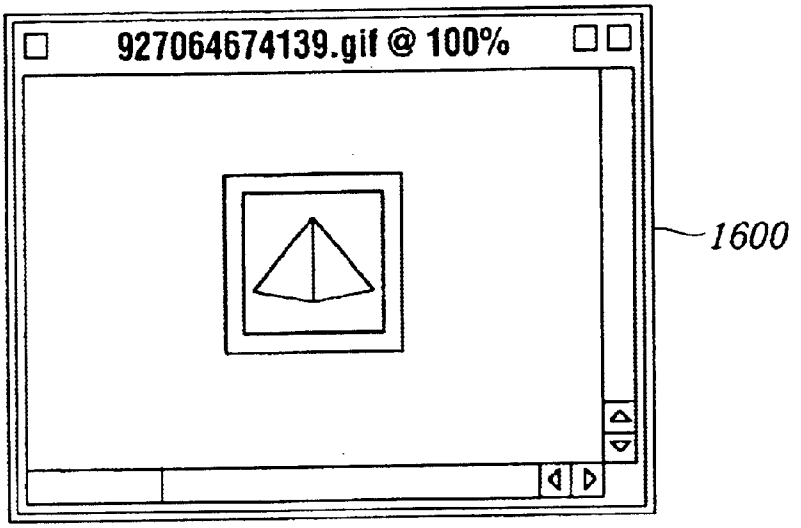


FIG.16

AUTOMATED PROCESSING AND DELIVERY OF MEDIA TO WEB SERVERS

BACKGROUND OF THE INVENTION

1. Technical Field

The invention relates to software systems. More particularly, the invention relates to an Internet server-based software system that provides delivery of automated graphics and other media to Web sites for access by an end user or consumer.

2. Description of the Prior Art

Most Web sites today are primarily handmade. From the guy publishing a simple online technology newsletter from his home, to the Fortune 1000 company's multi-tiered site with hundreds of pages of text, images, and animations, the Web developer and each of his HTML-coding and graphics-producing coworkers toil page by page and image by image. Thousands of established online companies employ hundreds of highly-skilled workers just to produce and maintain their Web sites. After all, the Web is now a major selling vehicle and marketing medium for many of these companies. The Web has even sprouted service industries such as, for example, public companies with multi-billion dollar valuations created just to consult and produce Web sites for others.

Most Web developers who use established WYSIWYG tools in the industry still must produce each page on their Web site one by one. The same rate applies to preparing and placing images, animations, and other visual assets. Each page represents its own set of issues ranging from whether to use GIF, JPEG, or PNG file formats, to finding the optimum bit depth for each image to ensure the fastest downloading through the different browsers of the consumer. The bottlenecked state of the customer's workflow to produce graphics for Web pages can be described as follows:

Current Workflow for Creating Web Graphics

Original Artwork/Asset Creation
  Use third-party point products
Asset Editing
  Scale/reduce/slice
Asset Format Conversion
  JPEG/GIF/PNG
Asset Staging
  Place in Web file system
  Edit HTML
Create/Modify HTML for particular page
Store HTML on Web server
View final pages
Repeat process for each version of each graphic on each page
Estimated time
  Two hours per page times the number of pages

Also, from a user's perspective, the current state of the art is to offer the consumer zooming and panning capabilities so that by clicking on an image the consumer can view more closely or from a different angle. On the horizon are pages with three-dimensional imagery that enable a user to move around a page that can look more like a room than a brochure. While interesting, these features are merely incremental improvements to a consumer's surfing experience.

D. C. A. Bulterman, *Models, Media, and Motion: Using the Web to Support Multimedia Documents*, Proceedings of 1997 International Conference on Multimedia Modeling, Singapore, 17–20 Nov. 1997 discloses "an effort underway by members of industry, research centers and user groups to define a standard document format that can be used in conjunction with time-based transport protocols over the Internet and intranets to support rich multimedia presentations. The paper outlines the goals of the W3C's Synchronized Multimedia working group and presents an initial description of the first version of the proposed multimedia document model and format."

Text and Graphics on UMI's ProQuest Direct. The Best (yet) of both Worlds, Online, vol. 21, no. 2, pp. 73–7, March–April 1997 discloses an information system that offers "periodical and newspaper content covering a wide range of business, news, and professional topics . . . letting the user search both text and graphics and build the product to suit. Articles can be retrieved in varying levels of detail: citation, abstracts, full text, and text with graphics. Images come in two flavors: Page Image, a virtual photocopy, and Text+Graphics, in which graphics are stored separately from the text and are manipulable as discrete items. . . . [The system] comes in two versions: Windows and Web."

John Mills Dudley, Network-Based Classified Information Systems, AU-A-53031/98 (Aug. 27, 1998) discloses a "system for automatically creating databases containing industry, service, product and, subject classification data, contact data, geographic location data (CCG-data) and links-to web pages from HTML, XML, or SGML encoded web pages posted on computer networks such as Internets or Intranets. . . . The . . . databases may be searched for references (URLs) to web pages by use of enquiries which reference one or more of the items of the CCG-data. Alternatively, enquiries referencing the CCG-data in the databases may supply contact data without web page references. Data duplication and coordination is reduced by including in the web page CCG-data display controls which are used by web browsers to format for display the same data that is used to automatically update the databases."

Cordell et al, Automatic Data Display Formatting with A Networking Application, U.S. Pat. No. 5,845,084 (Dec. 1, 1998) discloses a placeholder image mechanism. "When a data request is made, the data transfer rate is monitored. When the receive data transfer rate is slow, and the data contains an embedded graphical image of unknown dimensions, a small placeholder image is automatically displayed for the user instead of the actual data. The small placeholder image holds a place on a display device for the data or the embedded graphical image until the data or embedded graphical image is received. When embedded graphical image is received, the placeholder image is removed, and the display device is reformatted to display the embedded graphical image."

Jonathon R. T. Lewis, System For Substituting Tags For Non-Editable Data Sets In Hypertext Documents And Updating Web Files Containing Links Between Data Sets Corresponding To Changes Made To The Tags, U.S. Pat. No. 5,355,472 (Oct. 11, 1994) discloses a "hypertext data processing system wherein data sets participating in the hypertext document may be edited, the data processing system inserting tags into the data sets at locations corresponding to the hypertext links to create a file which is editable by an editor and the data processing system removing the tags, generating a revised data set and updating the link information after the editing process. Its main purpose is to preserve the linking hierarchy that may get lost when the individual data sets get modified."

Wistendahl et al, System for Mapping Hot Spots in Media Content Interactive Digital Media Program, U.S. Pat. No.

5,708,845 (Jan. 13, 1998) discloses a "system for allowing media content to be used in an interactive digital media (IDM) program [that] has Frame Data for the media content and object mapping data (N Data) representing the frame addresses and display location coordinates for objects appearing in the media content. The N Data are maintained separately from the Frame Data for the media content, so that the media content can be kept intact without embedded codes and can be played back on any system. The IDM program has established linkages connecting the objects mapped b y the N Data to other functions to be performed in conjunction with display of the media content. Selection of an object appearing in the media content with a pointer results in initiation of the interactive function. A broad base of existing non-interactive media content, such as movies, videos, advertising, and television programming can be converted to interactive digital media use. An authoring system for creating IDM programs has an object outlining tool and an object motion tracking tool for facilitating the generation of N Data. In a data storage disk, the Frame Data and the N Data are stored on separate sectors. In a network system, the object mapping data and IDM program are downloaded to a subscriber terminal and used in conjunction with presentation of the media content."

Rogers et al, Method for Fulfilling Requests of A Web Browser, U.S. Pat. No. 5,701,451 (Dec. 23, 1997) and Lagarde et al, Method for Distributed Task Fulfillment of Web Browser Requests, U.S. Pat. No. 5,710,918 (Jan. 20, 1998) disclose essentially "improvements which achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and presenting that information to the Web client in a text or graphics display at a location specified by the request."

Tyan et al, HTML Generator, European Patent Application No. EP 0843276 (May 20, 1998) discloses "generating an HTML file based on an input bitmap image, and is particularly directed to automatic generation of an HTML file, based on a scanned-in document image, with the HTML file in turn being used to generate a Web page that accurately reproduces the layout of the original input bitmap image."

TrueSpectra has a patent pending for the technology employed in its two products, IrisAccelerate and IrisTransactive. These products are designed for zooming and panning and simple image transformations and conversions, respectively. They support **10** file formats and allow developers to add new file formats via their SDK. They do not require the use of Flashpix for images. However, their documentation points out that performance is dependent on the Flashpix format. The system would be very slow if a non-Flashpix format was used.

TrueSpectra allows the image quality and compression to be set for JPEGs only. The compression setting is set on the server and all images are delivered at the same setting.

TrueSpectra has a simple caching mechanism. Images in the cache can be cleared out automatically at certain times and it does not have any dependency features for image propagation. The Web server needs to be brought down in order to update any original assets.

TrueSpectra does not require plug-ins to operate features such as zooming/panning or compositing, but suggests using their plug-ins for better performance. The alternative to plug-ins is using their Javascript or active server page technology. These technologies are used by many Web sites to provide interactivity, but not all Web browsers work correctly with these technologies.

In their latest version, TrueSpectra fixed their software so that communication to the server will not have to be through a certain port. Using TrueSpectra without a port requires the system to use Javascript or Active Server Pages. This requirement is not always possible so firewall issues remain.

TrueSpectra relies on Flashpix as its native file format and does not support media types such as multi-GIFs and sound formats. Flashpix files are typically larger than most file formats. Access to files is faster for zooming and panning, but appears to be quite slow.

The key to IrisTransactive is the compositing subsystem. It requires three things to build a shopping solution using image composition.

1) The original images must be created. It is suggested that the image be converted to Flashpix for better performance.
2) All of the individual images must be described in XML using the image composer program. The program allows the editor to specify anchor points, layer attributes, and layer names. The resulting file is between 5 k and 50 k.
3) The Web designer must place HTML referring to the XML in the Web site. By specifying parameters to the XML, the Web designer can turn on or off layers.

The herein above process for compositing images enables Web designers to create shopping sites. However, a lot of overhead is the result. The XML documents add 5 k–50 k to a Web site. The compositing commands that are embedded in the HTML are difficult to understand. And, because the compositing feature requires several steps to implement, it is not suitable for every image on a Web site. The process seems to be designed for the specific purpose of shopping.

The disclosed prior art fail to provide systems and mhethodologies that result in a quantum leap in the speed with which they can modify and add images, video, and sound to sites, in the volume of data they can publish internally and externally, and in the quality of the output. The development of such an automated media delivery system would constitute a major technological advance.

It would be advantageous to empower an end user with flexibility and control by providing interactive page capabilities.

It would be advantageous from an end user's perspective to generate Web pages that contain active graphics. For example, clicking on a Corvette image will cause a simple menu to pop up suggesting alternative colors and sizes in which to see the car. Clicking on portions of the image, such as a fender, can call up a close-in view of the fender.

It would be advantageous to provide an automated graphics delivery system that becomes part of the Web site infrastructure and operates as part of the Web page transaction and that thereby provides a less expensive and less time-consuming process.

It would be advantageous to provide a system for automated processing and delivery of media (images, video, and sound) to a Web server whereby it eliminates the laborious post-production and conversion work that must be done before a media asset can be delivered on a Web server.

It would be advantageous to create a dynamic Web site, wherein images are generated on demand from original assets, wherein only the original assets need to be updated, and wherein updated changes propagate throughout the site.

It would be advantageous to provide a system that generates media based on current Web server traffic thereby optimizing throughput of the media through the Web server.

It would be advantageous to provide a system that generates media that is optimized for the Web client, wherein client connection speed determines optimum quality and file size.

It would be advantageous to provide a system that generates media, whereby the media is automatically uploaded.

It would be advantageous to provide a system that automatically caches generated media so identical requests can be handled without regeneration of images.

It would be advantageous to provide a system that resides behind the Web server, thereby eliminating security issues.

It would be advantageous to provide a system wherein the client browser does not require a plug-in.

It would be advantageous to provide a system wherein the system does not require any changes to a Web server.

It would be advantageous to provide a system wherein the system manages the Web server media cache.

It would be advantageous to provide a system wherein the Web media is generated only if requested by a client browser.

SUMMARY OF THE INVENTION

An automated media delivery system that becomes part of the Web site infrastructure and operates as part of the Web page transaction is provided. The claimed invention streamlines the post-production process by automating the production of a media through proprietary HTML tags embedded in Web documents. The author simply places the original media in the system and adds proprietary HTML tags to HTML or other browser language. The system automatically processes the proprietary HTML tags and produces the media for the Web client. It also replaces the proprietary HTML tags with standard HTML tags so to be processed correctly by the HTML client.

This invention takes as input the client connection, server traffic, and proprietary HTML tags in order to generate the optimized media for the client. The need for the Web author to create different versions of a Web site is reduced because the need is automatically handled by the claimed invention. The generated media is cached so that further requests for the same media require little overhead.

Because the invention takes the original media and proprietary HTML tags as inputs for generating the Web media, it is possible to modify one or both and have the system automatically update the media on all of the associated Web pages.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a schematic diagram showing the placement of the system within a current Web infrastructure according to the invention;

FIG. **2** is a schematic diagram showing how a typical Web site delivers an HTML document and its graphics to a Web browser according to the prior art;

FIG. **3** is a schematic diagram showing delivery of an HTML document and media to a Web browser according to the invention;

FIG. **4** is a schematic diagram showing the components involved in Web site administration according to the prior art;

FIG. **5** is a schematic diagram showing the components of the system involved in Web site administration according to the invention;

FIG. **6** is a simple overview showing the components of the system according to the invention;

FIG. **7** is a schematic diagram showing the process flow of a proprietary enabled page delivered to a Web browser according to the invention;

FIG. **8** is a flow chart showing an authoring process according to the invention;

FIG. **9** is a flow chart showing an HTML parsing process according to the invention;

FIG. **10** is a flow chart showing a media creation process according to the invention;

FIG. **11** is a screen shot showing an administration tool according to the invention;

FIG. **12** displays a structure of a database record used for the system according to the invention;

FIG. **13** shows original media to be processed according to the invention;

FIG. **14** shows a portion on an HTML document with a proprietary tag according to the invention;

FIG. **15** shows an HTML document and an HTML document source according to the invention; and

FIG. **16** shows a generated GIF image according to the invention.

DETAILED DESCRIPTION OF THE INVENTION

An automated graphics delivery system that becomes part of the Web site infrastructure and operates as part of the Web page transaction is provided. The claimed invention streamlines the post-production process by automating the production of a media through proprietary HTML tags embedded in Web documents. The author simply places the original media in the system and adds proprietary HTML tags to HTML documents. The system automatically processes the proprietary HTML tags and produces the media for the Web client. It also replaces the proprietary HTML tags with standard HTML tags so it can be processed correctly by the HTML client.

This invention takes as input the client connection, server traffic, and proprietary HTML tags in order to generate the optimized media for the client. The need for the Web author to create different versions of a Web site is reduced because the need is automatically handled by the claimed invention. The generated media is cached so that further requests for the same media require little overhead.

FIG. **1** is a schematic diagram showing the placement of the system within a current Web infrastructure according to a preferred embodiment of the invention. The system **100** is attached to a Web server **110**, which is connected to multiple client browsers **120**(*a–d*) via the Internet **130**.

FIG. **2** is a schematic diagram showing how a typical Web site delivers an HTML document and its graphics to a Web browser according to the prior art. An original media **200** is passed to post-production systems **210**, wherein the media **200** is manipulated by hand and prepared for the Web. The result is a Web media **220**. The Web media **220** and an associated HTML document **230** referring to the media **220** by media tags are input to a Web server **110** for a Web browser **120** to view via the Internet **130**.

FIG. **3** is a schematic diagram showing delivery of an HTML document and media to a Web browser according to a preferred embodiment of the invention. An original media **200** and an HTML document embedded with proprietary media tags **300** are input into the system **100**. The system **100** generates a Web-safe media **220** and a modified HTML document **230** that refers to the Web media, and automatically loads them onto the Web server **110** for view by a Web browser **120** via the Internet **160**.

FIG. **4** is a schematic diagram showing components involved in Web site administration according to the prior art. Original media assets **400** are original images, video, or sound that have not been prepared for the Web. Web sites

usually need to manage the placement of media on the network for easy retrieval by Web designers. Post-production systems **410** vary from Web site to Web site. Post-production systems **410** are usually custom procedures that Web designers use to convert an original media, such as an image, to one that can be displayed on the Web. Post-production systems **410** also upload finished images to Web image systems. Web images **420** are Web versions of the original images. Web images **420** are ready for retrieval by the Web server **110** to be delivered to a Web browser **120**. Any image to be modified or updated must pass through the herein above three components before it can be delivered to the Web browser **120**. HTML pages **460** have references to Web images **420**.

FIG. **5** is a schematic diagram showing the components involved in Web site administration according to a preferred embodiment of the invention. Web site administration is simplified using the claimed invention. Asset management, automatic image manipulation, automatic image conversion, automatic image upload, and automatic disk management **500** are provided by the claimed invention.

FIG. **6** is a simple overview showing the components of the system according to a preferred embodiment of the invention. HTML with proprietary tags **300** is the original HTML document that is embedded with proprietary tags which describe how the images are to be manipulated for the Web. Java servlet engine **600** is a third-party product that allows the system **100** to interface with the Web server **110** and execute Java servlet code. The Web server **10** is third-party software that delivers Web pages to a Browser **120**. The Browser **120** views Web pages that are sent from the Web server **110**. Modified HTML with system created images **230** are a final result of the system. Modified HTML **230** is a standard HTML document without proprietary embedded tags and with standard Web graphics.

The System.

A preferred embodiment of the system **100** is provided.

HTML parsing subsystem **610** parses through an HTML document and searches for proprietary tags. If it finds a proprietary tag it hands it to a media caching subsystem **620** for further processing. The media caching subsystem **620** returns a standard HTML tag. The HTML parsing subsystem **610** then replaces the proprietary tag it found with the returned tag. The parsing subsystem **610** then continues searching for a next proprietary tag, repeating the process herein above. The process is finished when no more proprietary tags can be found.

The media caching subsystem **620** determines if an image has been created for the requested proprietary tag. If the image has already been created and the files that built that image have not been modified, the media caching subsystem **620** returns an HTML tag that refers to a previously-generated image. If the image has not been created, the media caching subsystem **620** hands the HTML tag to a media creation subsystem **630**. The media creation subsystem **630** returns an image to the media caching subsystem **620**. The media caching subsystem **620** adds the created image and the HTML tag to a media cache database **640**.

The media cache database **640** contains references to the created images **645**. In a preferred embodiment, the references are the script used to create the image, the names of the images used to create the image, the dates of those files, and the HTML that represents the created image. The media caching subsystem **620** performs lookups in this database to determine if the image has been created. If the image has not been created the media caching subsystem **620** calls upon the media creation subsystem **630** to create the image and then store the results in the media cache database **640**.

The media creation subsystem **630** takes a proprietary tag from the media caching subsystem **620** and generates an image. The image is generated by deciphering the tag and handing it to the media processing engine **650**. After the image is created, the media creation subsystem returns the name of the newly created image to the media caching subsystem **620**.

The media processing engine **650** interprets the proprietary tag and generates the image. The media processing engine **650** looks up images in a media repository to obtain the location of the original file.

The media repository **660** contains original images **665** used in the system **100**.

FIG. **7** is a schematic diagram showing the process flow of a proprietary enabled page delivered to a Web browser according to a preferred embodiment of the invention. An original media **200** is created. The media **200** is placed into the system **100** in the media repository **660**. Similarly, an HTML document with proprietary tags **300** is created and placed on a Web server **110**. A user requests a Web page from a Web browser **120**. The Web server **110** passes the requested page to an HTML parser **610**. The HTML parser **610** parses HTML looking for media tags. The parser **610** looks, up media tags in a media tags database **640**. If the media tag is found, then the system **100** produces a modified HTML document **230**. Otherwise, the media creation subsystem **630** uses the media tag to generate a Web media **220**. The generated Web media **220** is placed in a media cache subsystem **620**. The proprietary media tag is converted by a converter **700** to a standard HTML tag that refers to the generated media **220** in cache. The media tag and the HTML equivalent are stored in the media tags database **640**. Media tags are replaced by standard HTML equivalent to provide a modified HTML document **230**. The modified HTML document **230** is delivered to the Web server **110**. The Web server **100** delivers the modified HTML document **230** to the browser **120** via the Internet for a user to view.

FIG. **8** is a flow chart showing an authoring process according to a preferred embodiment of the invention. The process starts **(800)** when a user adds an original graphic to the system **(810)**. The user then creates an HTML document that contains proprietary media tags **(820)**. The user then places the HTML document on a Web server **(830)** and ends the authoring process **(840)**.

FIG. **9** is a flow chart showing an HTML parsing process according to a preferred embodiment of the invention. The process starts **(900)** when a consumer requests a Web page **(910)**. A Web server hands the request of the Web page to the system **(920)**. The system parses the, Web page **(930)**. The system looks for a media tag **(940)**. If found, the system retrieves the HTML equivalent of the media tag **(950)** and replaces the media tag with the HTML equivalent tag **(960)**. The system continues parsing the Web page for tags **(970)** by returning to step **(940)**. When no more tags are found, the system delivers the modified Web page to the Web server **(980)** and therein ends the process **(990)**.

FIG. **10** is a flow chart showing a media creation process according to a preferred embodiment of the invention. The process starts **(1000)** when the system requests an HTML equivalent to a proprietary media tag **(1010)**. The Media tag is combined with bandwidth information **(1020)**. The subsystem checks if the media tag already exists in the media tag database **(1030)**. If it does, the subsystem checks if any of the original assets used to create the media have been changed **(1040)**. If not, then the subsystem retrieves the HTML equivalent tag from the database **(1050)** and returns the HTML equivalent tag to the requesting system **(1060)**. If

any of the original assets used to create the media have been changed **(1040)**, then the subsystem removes the media tag entry from the media database **(1070)** and creates the media using the media tag **(1080)**. The subsystem then stores the media in a media cache **(1090)**. The subsystem generates the HTML referring to the generated media **(1100)** and places the media tag and the HTML equivalent in the media tag database **(1110)**. The HTML equivalent is returned to the requesting system **(1060)** and the process stops **(1120)**.

The differences between using HTML and the proprietary tags disclosed herein are noted. HTML allows Web designers to create Web page layouts. HTML offers some control of the images. HTML allows the Web designer to set the height and width of an image. However, all of the other image operations disclosed herein are supported by the claimed invention and are not supported by HTML.

Table A herein below provides the claimed proprietary tags according to a preferred embodiment of the invention. The use of the term "freeride" refers to an internal code name for the invention.

TABLE A

Tags
Generate image <freerideimage> mediascript </freerideimage> Generate a standard Web image. Generate thumbnail image linked to full image <freerideimagethumbnail> mediascript <xs=size ys=size/freerideimagethumbnail> Generate a thumbnail of specified size and link it to the full size version. Generate zoom and pan image <freerideimagezoom> mediascript </freerideimagezoom> Generate a zoomable/panable image. Security <freerideimagesecure> </freerideimagesecure> Specifies that all images found between these tags are secured images and the system will determine access before generating.

Table B herein below provides the claimed script commands according to a preferred embodiment of the invention. Additional commands may be added as needed.

TABLE B

Media processing script commands
Add Noise Noise_AddNoise([amount=<value 1 . . . 999>] [gaussian] [grayscale]) This command adds noise to the image. Adjust HSB AdjustHsb([hue @ <value ±255>] [saturation @ <value ±255>] [brightness @ <value ±255>]) This command allows the HSB of an image to be altered. This can be applied to images of all supported bit-depths. Adjust RGB AdjustRgb([brightness @ <value ±255>] [contrast @ <value ±255>] [red @ <value ±255>] [green @ <value ±255>] [blue @ <value ±255>] [noclip @ <true, false>] [invert @ <true, false>]) This command allows the contrast, brightness, and color balance of an image to be altered. Blur Blur(radius @ <value 0.30>) This command applies a simple blur filter on the image. Blur Convolve Blu_Blur( ) This command commands perform a simple 3×3 convolution for blurring. Blur Convolve More Blur_MoreBlur( )

TABLE B-continued

Media processing script commands
This command commands perform a stronger 3×3 convolution for blurring. Blur Gaussian Blur_GaussianBlur([radius=<value 0.1 . . . 250>]) This command applies a Gaussian blur to the image. Blur Motion Blur_MotionBlur([distance=<value 1 . . . 250>] [angle=<degrees>]) This command applies motion blurring to the image using the specified distance and angle. Brush Composite Composite(source @ {<User-Defined Media Object name>} [x @ <pixel>] [y @ <pixel>] [onto] [opacity @ <value 0 . . . 255>] [color @ <color in hexadecimal>] [colorize @ <true, false>] [saturation @ <value 0 . . . 255>]) This command composites the specified "brush" (foreground) image onto the current "target" (background) image. Colorize Colorize(color @ <color in hexadecimal> [saturation @ <value 0 . . . 255>]) This command changes the hue of the pixels in the image to the specified color. Convert Convert(rtype @ <bit-depth> {dither @ <value 0 . . . 10>]) This command converts the image to the specified type/bit-depth. Convolve Convolve(Filter @ <filtername>) This command applies a basic convolution filter to the image. In a user interface driven system, the filters could be stored in files and edited/created by the user. Crop/Resize Canvas Crop([xs @ {<pixels>, <percentage + "%">}] [ys @ {<pixels>, <percentage + "%">}] [xo @ <left pixel>] [yo @ <top pixel>] [padcolor @ <color in hexadecimal>] [padindex @ <value 0 . . . 255>]) This command crops the media to a specified size. Discard Discard( ) This command removes the designated Media Object from memory. Drop Shadow DropShadow([dx @ <pixels>] [dy @ <pixels>] [color @ <color in hexadecimal>] [opacity @ <value 0 . . . 255>] [blur @ <value 0 . . . 30>] [enlarge @ <true, false>]) This command adds a drop shadow to the image based on its alpha channel. Equal Equal(source @ {<User-Defined Media Object name>}) This command compares the current media with the one specified. If the media are different in any way, an error value is returned. Equalize Equalize([brightness @ <−1, 0 . . . 20>] [saturation @ <−1, 0 . . . 20>]) This command equalizes the relevant components of the media. Equalization takes the used range of a component and expands it to fill the available range. Export Channel ExportGun(Channel @ <channelname>) This command exports a single channel of the source as a grayscale image. Find Edges Stylize_FindEdges([threshold=<value 0 . . . 255>] [grayscale] [mono] [invert]) This command finds the edges of the image based on the specified threshold value. Fix Alpha FixAlpha( ) This command adjusts the RGB components of an image relative to its alpha channel. Flip Flip(<horizontal, vertical> @ <true, false>) This command flips the media vertically or horizontally. Frame Add FrameAdd(Source @ <filename>) This command adds the given frame(s) to the specified Media Object. Glow/Halo

TABLE B-continued

Media processing script commands
Glow(Size @ <value 0 . . . 30> [halo @ <value 0 . . . size>] [color @ <color in hexadecimal>] [opacity @ <value 0 . . . 255>] [blur @ <value 0 . . . 30>] [enlarge @ <true, false>]) This command produces a glow or halo around the image based on the image's alpha.
High Pass
Other_HighPass([radius=<value 0.1 . . . 250>]) This command replaces each pixel with the difference between the original pixel and a Gaussian blurred version of the image.
Import Channel
ImportGun(channel @ <channel name> source @ {<User-Defined Media Object name>} [rtype @ <bit-depth>]) This command imports the specified source image (treated as a grayscale) and replaces the selected channel in the original.
Load
Load(Name @ <filename> [type @ <typename>] [transform @ <true, false>]) This command loads a media from the specified file.
Maximum
Other_Maximum([radius=<value 1 . . . 10>]) This command scans the area specified by the radius surrounding each pixel, and then replaces the pixel with the brightest pixel found.
Minimum
Other_Minimum([radius=<value 1 . . . 10>]) This command scans the area specified by the radius surrounding each pixel, and then replaces the pixel with the darkest pixel found.
Normalize
Normalize([clip @ <value 0 . . . 20>]) This command expands the volume of the sample to the maximum possible.
Pixellate Mosaic
Pixellate_Mosaic([size=<value 2 . . . 64>]) This command converts the image to squares of the specified size, where each square contains the average color for that part of the image.
Pixellate Fragment
Pixellate_Fragment([radius=<value 1 . . . 16>]) This command produces four copies of the image displaced in each direction (up, down, left, right) by the specified radius distance and then averages them together.
Quad Warp
Quadwarp([tlx=<position>] [tly=<position>] [trx=<position>] [try=<position>] [blx=<position>] [bly=<position>] [brx=<position>] [bry=<position>] [smooth]) This command takes the corners of the source image and moves them to the specified locations, producing a warped effect on the image.
Reduce to Palette
Reduce([colors @ <num colors>] [netscape @ >true, false>] [b&w @ <true, false>] [dither @ <value 0 . . . 10>] [dithertop @ <value 0 . . . 10>] [notbackcolor] [pad @ <true, false>]) This command applies a specified or generated palette to the image.
Rotate
Rotate(Angle @ <value 0 . . . 359> [smooth @ <true, false>] [enlarge @ <true, false>] [xs @ <pixels>] [ys @ <pixels>]) This command rotates the media by the specified angle in degrees.
Rotate 3D
Rotate3d([anglex @ <angle ±89>] [angley @ <angle ±89>] [distance @ <value>]) This command rotates the image in 3D about either the x-axis or y-axis.
Save
Save([type @ <image-type>]) This command saves a media to the specified file.
Scale
Scale([xs @ {<pixels>, <percentage + "%">}] [ys @ {<pixels>, <percentage + "%">}] [constrain @ <true, false>] [alg @ {"fast", "smooth", "outline"}] [x1 @ <pixels>] [y1 @ <pixels>] [x2 @ <pixels>] [y2 @ <pixels>]) This command scales the image to the specified size.
Select
Selection([source @ <User-Defined media Object>}] [remove @ <true, false>] [invert @ <true, false>] [backcolor] [color=<color>] [index=<value>] [opacity @ <value 0 . . . 255>]) This command manages the selected region for the current Media Object.
Set Color
SetColor([backcolor @ <color in hexadecimal>] [forecolor @ <color in hexadecimal>] [backindex @ <value 0 . . . 255>] [foreindex @ <value 0 . . . 255>] [transparency @ ("on", "off")]) This command allows the background color, foreground color, and transparency state of an image to be set.
Set Resolution
SetResolution([dpi @ <value>] [xdpi @ <value>] [ydpi @ <value>]) This command changes the DPI of the image in memory.
Sharpen
Sharpen_Sharpen( ) This command sharpens the image by enhancing the high-frequency component of the image.
Sharpen More
Sharpen_SharpenMore( ) This command sharpens the image by enhancing the high-frequency component of the image, but is stronger than the standard sharpening.
Stylize Diffuse
Stylize_Diffuse([radius=<value 0 . . . >] [lighten] [darken] This command diffuses the image by randomizing the pixels within a given pixel radius.
Stylize Embose
Stylize_Emboss([height=<value 1 . . . 10>] [angle=<degrees>] [amount=<percentage 1 . . . 500>]) This command converts the image to an embossed version.
Text Drawing
DrawText(Text @ <string> Font @ <font file> [size @ <value>] [color @ <color in hexadecimal>] [smooth @ <true, false>] [<left, right, top, bottom> @ <true, false>] [x @ <pixel>] [y @ <pixel>] [wrap @ <pixel-width>] [justify @ {left, center, right}] [angle @ <angle >]) This command composites the specified text string onto the image.
Text Making
MakeText(text @ <string> font @ <font file> [path @ <path to font directory>] [size @ <value 1 . . . 4095>] [color @ <color in hexadecimal>] [smooth @ <true, false>] [wrap @ <pixel-width>] [justify @ {left, center, right}] [angle @ <angle>]) This command creates a new image that includes only the specified text.
Trace Contour
Stylize_TraceContour([level=<value 0 . . . 255>] [upper] [invert]) This command traces the contour of the image at the specified level (for each gun).
Unsharpen Mask
Sharpen_UnsharpMask([amount=<percentage 1 . . . 500>] [radius=<value 0.1 . . . 250>] [threshold=<value 0 . . . 255>]) This command enhances the edges and detail of an image by exaggerating differences between the image and a gaussian blurred version of the same image.
Zoom
Zoom([xs @ <pixels>] [ys @ <pixels>] [scale @ <value>] [x @ <left pixel>] [y @ <top pixel>]) This command zooms in on a specified portion of the media and fits it to the specified size. This constitutes a crop followed by a scale.

Table C herein below provides a list of features provided by a preferred embodiment of the invention. It is noted that the list of features included in Table C is by no means complete. In other embodiments, the list of features is expanded or reduced as needed.

TABLE C

System Feature List
Reads and writes various file formats
BMP, GIF, JPG, PNG, TIF, PICT, TGA, PSD, FPX
Supports many image processing operations
Dynamically creates Web images from original assets
Dynamically creates thumbnail images
Dynamically creates images that can be panned and zoomed without browser plug-ins or special file formats
Automatically propagates changes of original assets throughout a Web site
Uses an intelligent caching mechanism
Clean up image cache on demand
Eliminates orphaned image files
Optimizes Web server cache by providing most recent images
Renders TrueType fonts on the server instead of browser
Uses intelligent scaling of line drawings
Allows Web designers to manipulate images with proprietary tags
Preserves original image assets
Optimizes Web server traffic by adiusting the bandwidth of graphics
Optimizes images for client connection speed
Allows clients to specify the quality of images on a Web site
Allows Web designers to dynamically create images by manipulating proprietary tags in their applications (server or client side)

FIG. **11** is a screen shot showing an administration tool according to a preferred embodiment of the invention. Specifically, FIG. **11** shows an administration page that contains cached images of generated scripts. The use of the term "freeride" refers to an internal code name for the invention.

FIG. **12** displays a structure of a database record used for the system according to a preferred embodiment of the invention. A Script Table **1200** has 5 columns, Media Script **1210**, HTML Equivalent **1220**, Bandwidth **1230**, Generated File **1240**, and Dependency List **1250**. A Dependency Table **1260** has two columns, File Name **1270** and Modification Date **1280**.

Snowboard Store Example

Background.

The snowboard store highlights several features of the claimed system. The snowboard store is an imaginary store that allows a user to configure his or her snowboard. The store consists of five logos, five board colors, and four boards. The consumer dicks on the buttons to change the snowboard represented in the middle of the screen. When the consumer has configured the snowboard they the snowboard can be purchased by selecting a buy button.

Prior Art Method.

To create the snowboard site today, the Web designer must render all possible combinations of the board. The number of combinations is five logos × five board colors × four boards=100. The designer also must render all the buttons. The creation process is very tedious and involves a lot of production work. Typically, most Web sites do not even attempt such an endeavor. Also, other issues must be addressed, such as, for example, updating the Web site and scripting. For example, updating a single logo involves updating a minimum of 20 images.

The prior art method sustains a graphic intensive site that requires management of at least 100 images. Updates to the Web site are time-consuming and prone to human error.

The Claimed Method.

A preferred embodiment of the method scripts the image creation process in HTML to create a dynamic Web site. There is no need to create over 100 images. The claimed system generates images on demand. The Web site only needs to create original assets. The scripting process involves writing the proprietary scripts. In the current example herein, scripting buttons is very simple. Once one button is created, simply copy and paste the HTML to create another button or many buttons. Only the name of the image to be overlaid on the button must be changed. The Webmaster then creates a simple program that reads what object a user has clicked on and generates a proprietary tag. The tag is then sent to the claimed system to generate a center image.

The claimed method allows the creation of all 100 combinations automatically. When the Web site receives an updated image, only the original image needs to be updated. Any change to the original image automatically propagates throughout the system. The Web site is easier to manage. Testing of the Web site is easier because there is no need to test all 100 combinations. A small subset of combinations will guarantee adequate coverage.

Processing of an Image Tag Example (FIG. **13–16**).

FIG. **13** shows two original images **1300** and **1310** to be processed according to a preferred embodiment of the invention.

FIG. **14** shows a portion on an HTML document with a proprietary tag **1400**, <freerideimage></freerideimage> according to a preferred embodiment of the invention. The use of the term "freeride" refers to an internal code name for the invention.

FIG. **15** shows an HTML document **1500** as viewed in a browser and an HTML document source **1510**, according to a preferred embodiment of the invention. The use of the term "freeride" refers to an internal code name for the invention.

FIG. **16** shows a generated GIF image **1600** according to a preferred embodiment of the invention.

Accordingly, although the invention has been described in detail with reference to a particular preferred embodiment, persons possessing ordinary skill in the art to which this invention pertains will appreciate that various modifications and enhancements may be made without departing from the spirit and scope of the claims that follow.

What is claimed is:

1. A process for delivering an original media, comprising the steps of:

placing said original media in a network system;

creating an HTML document or browser language having proprietary media tags and placing said HTML document or browser language onto a Web server;

actuating a user request from a customer browser for a Web page by having said Web server pass said requested Web page to an HTML parser on said network system;

parsing said HTML document or browser language on said parser by looking for said proprietary media tags;

looking up said proprietary media tags in a media tags database on said network system;

if at least one media tag of said proprietary media tags is not found,

generating a Web media by using said at least one media tag; placing said

generated Web media in a media cache on said network system;

converting said at least one media tag to at least one standard HTML equivalent tag that refers to said media in said cache; and

storing said at least one media tag and said at least one standard HTML equivalent tag in said database;

modifying said HTML document or browser language by replacing said at least one media tag by said at least one standard HTML equivalent tag;

delivering said modified HTML document or browser language from said network system to said Web server; and

delivering said modified HTML document or browser language from said Web server to said customer browser for said customer to view.

2. A network system for delivering an original media, comprising:

means for placing said original media in said network system;

means for creating HTML or browser language having proprietary media tags and placing said HTML or browser language onto a Web server;

means for actuating a request from a customer browser for a Web page by having said Web server pass said requested Web page to an HTML parser on said network system;

means for parsing said HTML document on said parser by looking for said proprietary media tags;

means for looking up said proprietary media tags in a media tags database on said network system;

if at least one media tag of said proprietary media tags is not found,

means for generating a Web media by using said at least one media tag;

placing said generated Web media in a media cache on said network system;

means for converting said at least one media tag to at least one standard HTML equivalent tag that refers to said media in said cache; and

means for storing said at least one media tag and said at least one standard HTML equivalent tag in said database;

means for generating HTML or browser language by replacing said at least one media tag by said at least one standard HTML equivalent tag;

means for delivering said generated HTML or browser language from said network system to said Web server; and

means for delivering said modified generated HTML or browser language from said Web server to said customer browser for said customer to view.

3. An media delivery system using a Java servlet engine, a Web server, and a customer browser for generating HTML or browser language, said generated HTML or browser language having a proprietary image, from an original HTML document or browser language embedded with a proprietary tag, comprising:

an HTML parser subsystem adapted to search for a proprietary tag in said original HTML or browser language and to replace said proprietary tag with a standard HTML tag;

a media caching subsystem adapted to determine a status of existing or needs to be created of said proprietary image for said proprietary tag, and to send said standard HTML tag to said media creation subsystem;

a media cache database adapted to store data associated with said proprietary image;

a media creation subsystem adapted to decipher said proprietary tag;

a media processing engine adapted to receive from said media creation subsystem said deciphered proprietary tag and to interpret said tag to generate said proprietary media; and

a media repository adapted to store at least one original media associated with said proprietary media;

wherein said media cache database further comprises:

a script table having a media processing script column, an HTML Equivalent column, a Bandwidth column, a Generated File column, and a Dependency List column; and

a dependency table having a File Name column and a Modification Date column.

4. The system of claim **3**, wherein said original HTML or browser language is embedded with a plurality of proprietary tags, said proprietary tags associated with a plurality of proprietary media and associated with a plurality of original media.

5. The system of claim **3**, wherein said stored data associated with said proprietary media further comprises:

a script used to create said proprietary media, said script having an associated date;

a name of said at least one original media used to create said proprietary media, said name having an associated date; and

said generated HTML or browser language.

6. The system of claim **5**, wherein said media caching subsystem further comprises:

means for determining if said associated script data and said associated original media date have been modified.

7. The system of claim **3**, wherein said HTML parser subsystem further comprises:

means for sending said proprietary tag to said media caching subsystem for further processing;

means for receiving from said media caching subsystem said standard HTML tag;

means for searching for a next proprietary tag to replace said next proprietary tag with a next standard HTML tag; and

means for determining that no more proprietary tags exist in said original HTML document or browser language.

8. The system of claim **3**, wherein said media caching subsystem further comprises:

means for determining if said proprietary media has been modified.

9. A process for creating a media associated with an HTML equivalent tag and associated with a media tag for a system, comprising the steps of:

receiving a request from said system for said HTML equivalent tag;

combining said media tag with bandwidth information;

determining if a media tag entry associated with said media tag exists in a media tag database;

if said media tag entry exists in said media database,

determining if at least one original asset used to create said media has changed;

if said at least one original asset used to create said media has not changed,

retrieving said HTML equivalent tag from said database; and

return said HTML equivalent tag to said system;

if said at least one original asset used to create said media has changed,

removing said media tag entry from said database;

create said media using said media tag;

storing said media in a media cache;

generating an HTML document referring to said generated media;

placing said media tag and said HTML equivalent tag in said media database; and

return said HTML equivalent tag to said system;

if said media tag entry does not exist in said media database,

create said media using said media tag;

storing said media in a media cache;

generating HTML or browser language referring to said generated media;

placing said media tag and said HTML equivalent tag in said media database; and

return said HTML equivalent tag to said system.

* * * * *