US 20020078093A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2002/0078093 A1

Samaniego et al. (43) Pub. Date: Jun. 20, 2002

(54) AUTOMATED MEDIA DELIVERY SYSTEM

(76) 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); **David C. Salmon**, San Rafael, CA (US); **Joshua N. Devan**, Kentfield, CA (US)

Correspondence Address:
GLENN PATENT GROUP
3475 EDISON WAY
SUITE L
MENLO PARK, CA 94025 (US)

(21) Appl. No.: **09/929,904**

(22) Filed: **Aug. 14, 2001**

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/425,326, filed on Oct. 21, 1999. Non-provisional of provisional application No. 60/226,043, filed on Aug. 16, 2000.

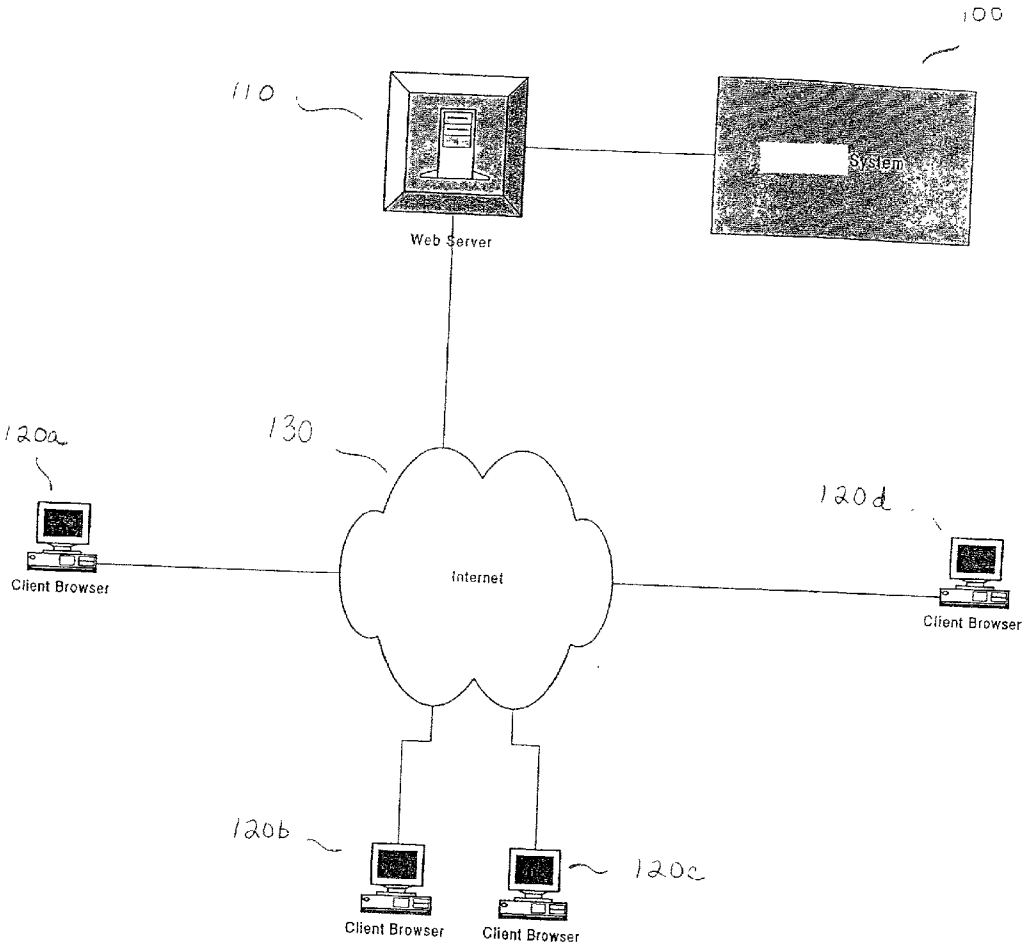
Publication Classification

(51) **Int. Cl.**[7] **G09G 5/12**

(52) **U.S. Cl.** **707/513**; 707/500.1

(57) **ABSTRACT**

An automatic graphics delivery system that operates in parallel with an existing Web site infrastructure is provided. The system streamlines the post-production process by automating the production of media through content generation procedures controlled by proprietary tags placed by an author within URLs embedded within Web documents.



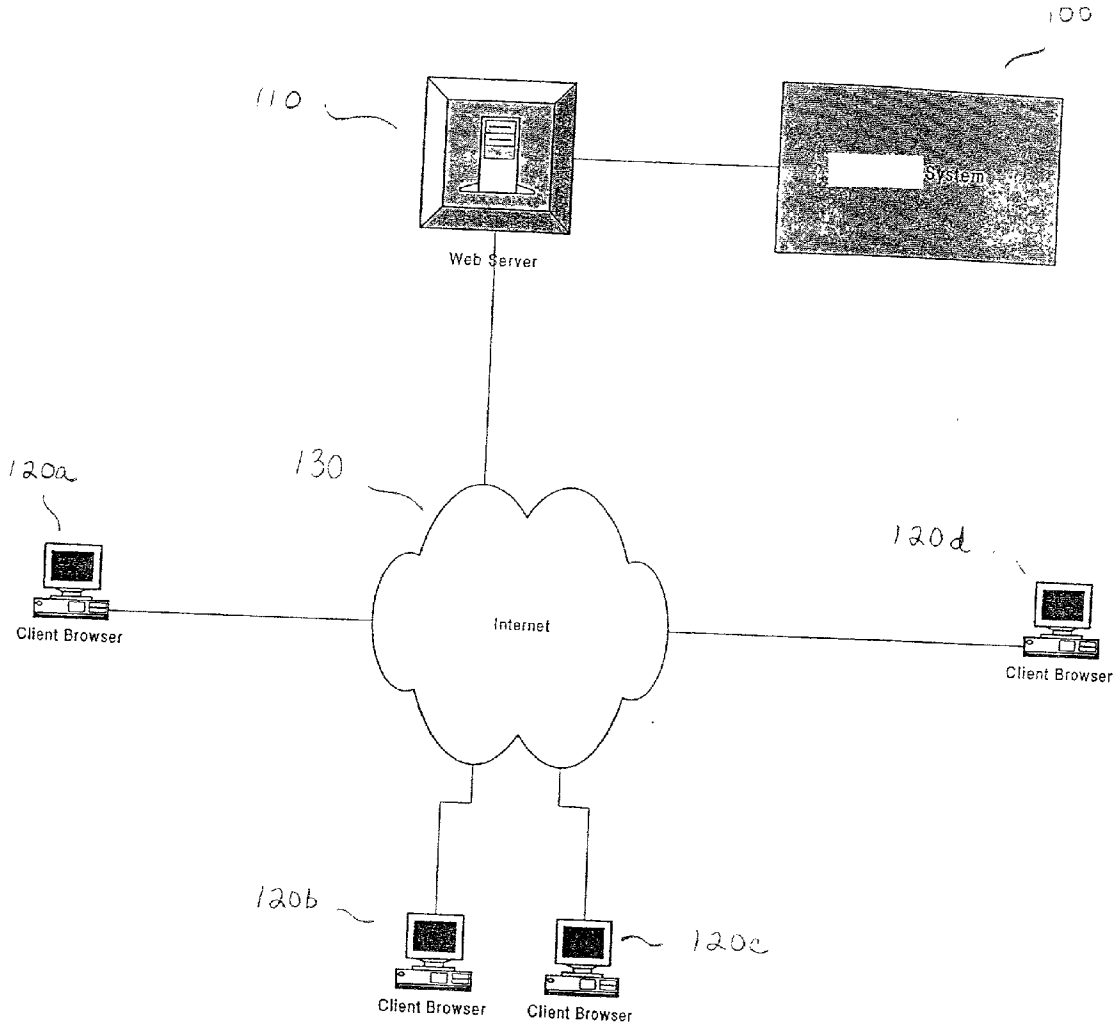

100
110
System
Web Server
120a
130
120d
Client Browser
Internet
Client Browser
120b
120c
Client Browser
Client Browser

Fig 1

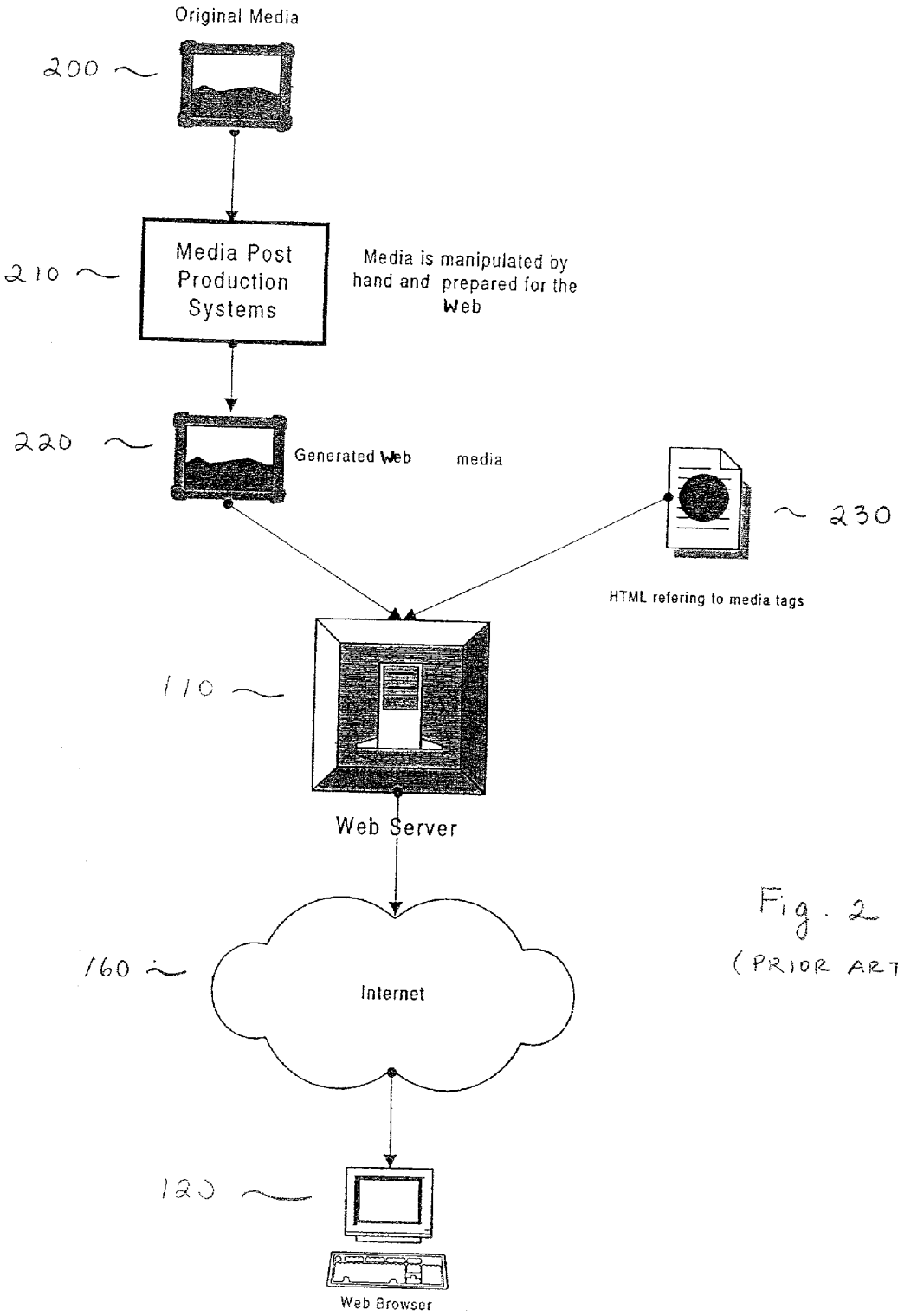


Fig. 2
(PRIOR ART)

Fig. 3

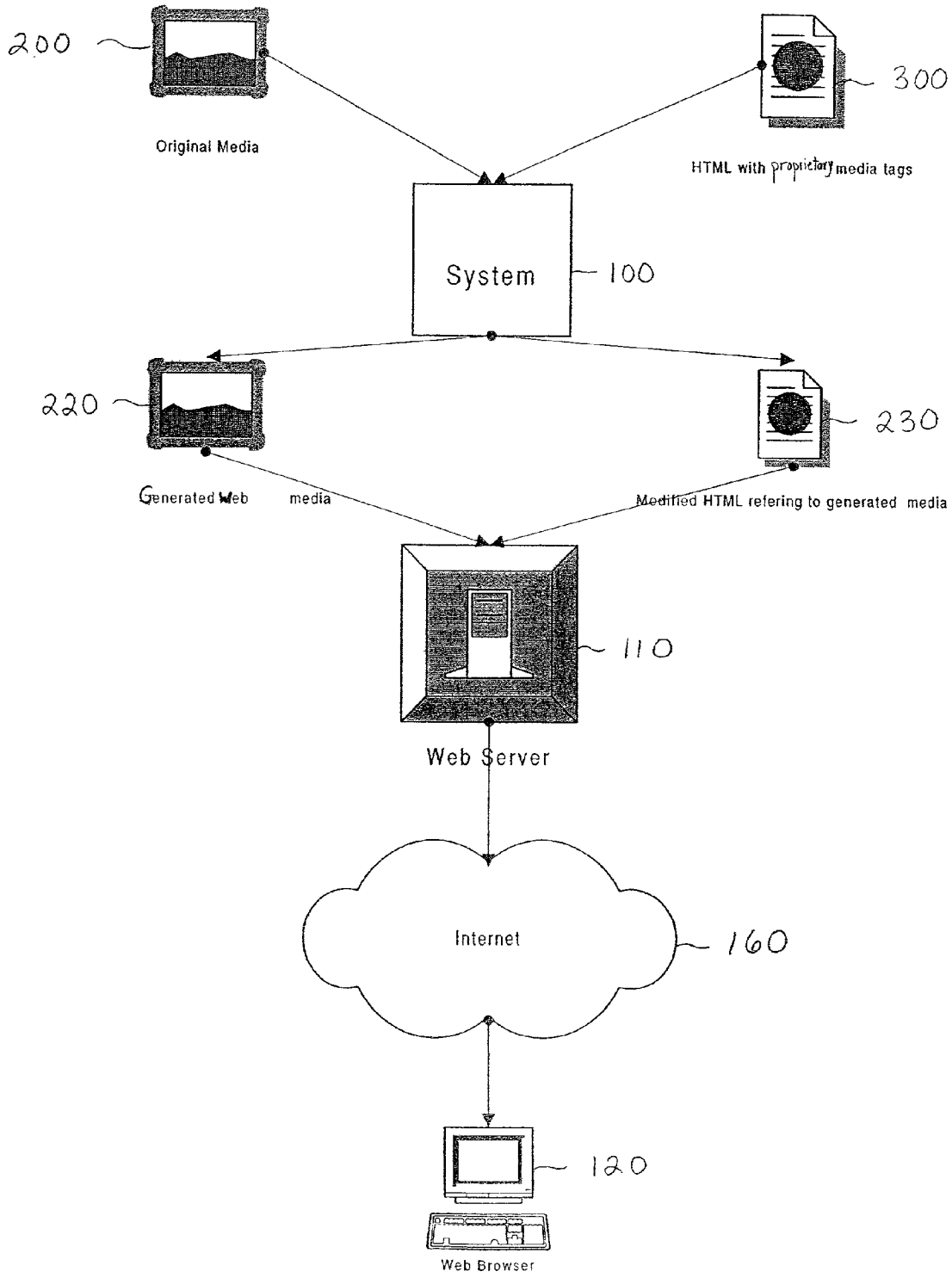

200
Original Media
300
HTML with proprietary media tags
System
100
220
Generated Web media
230
Modified HTML refering to generated media
110
Web Server
Internet
160
120
Web Browser

Fig. 4 (PRIOR ART)

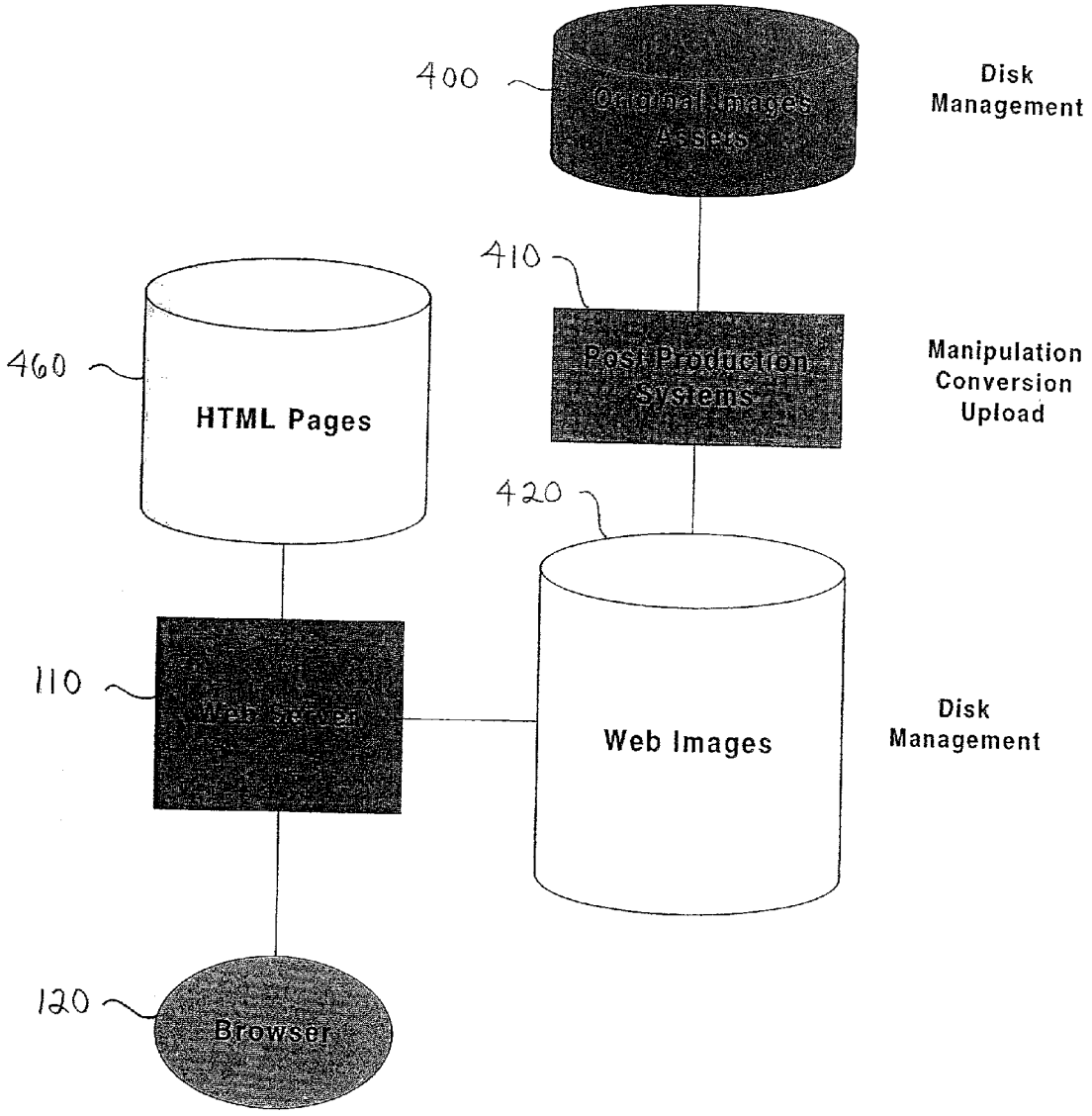
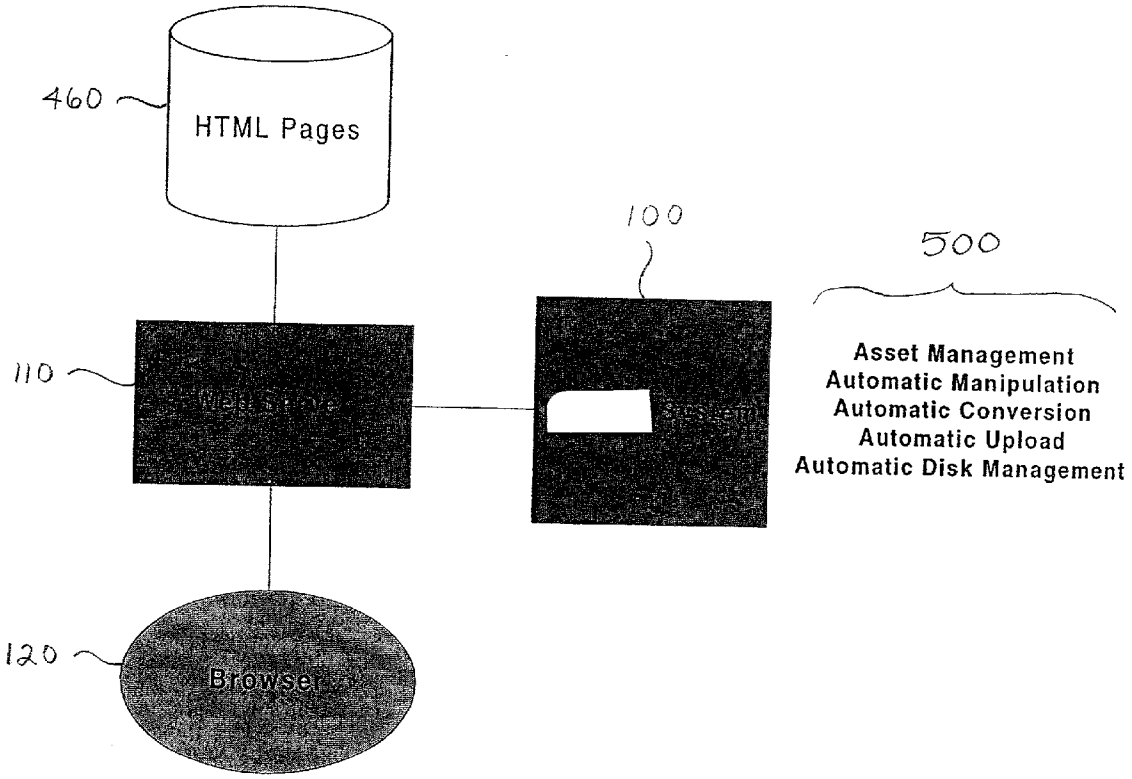


Fig. 5


460
HTML Pages
100
500
110
Web Server
System
Asset Management
Automatic Manipulation
Automatic Conversion
Automatic Upload
Automatic Disk Management
120
Browser

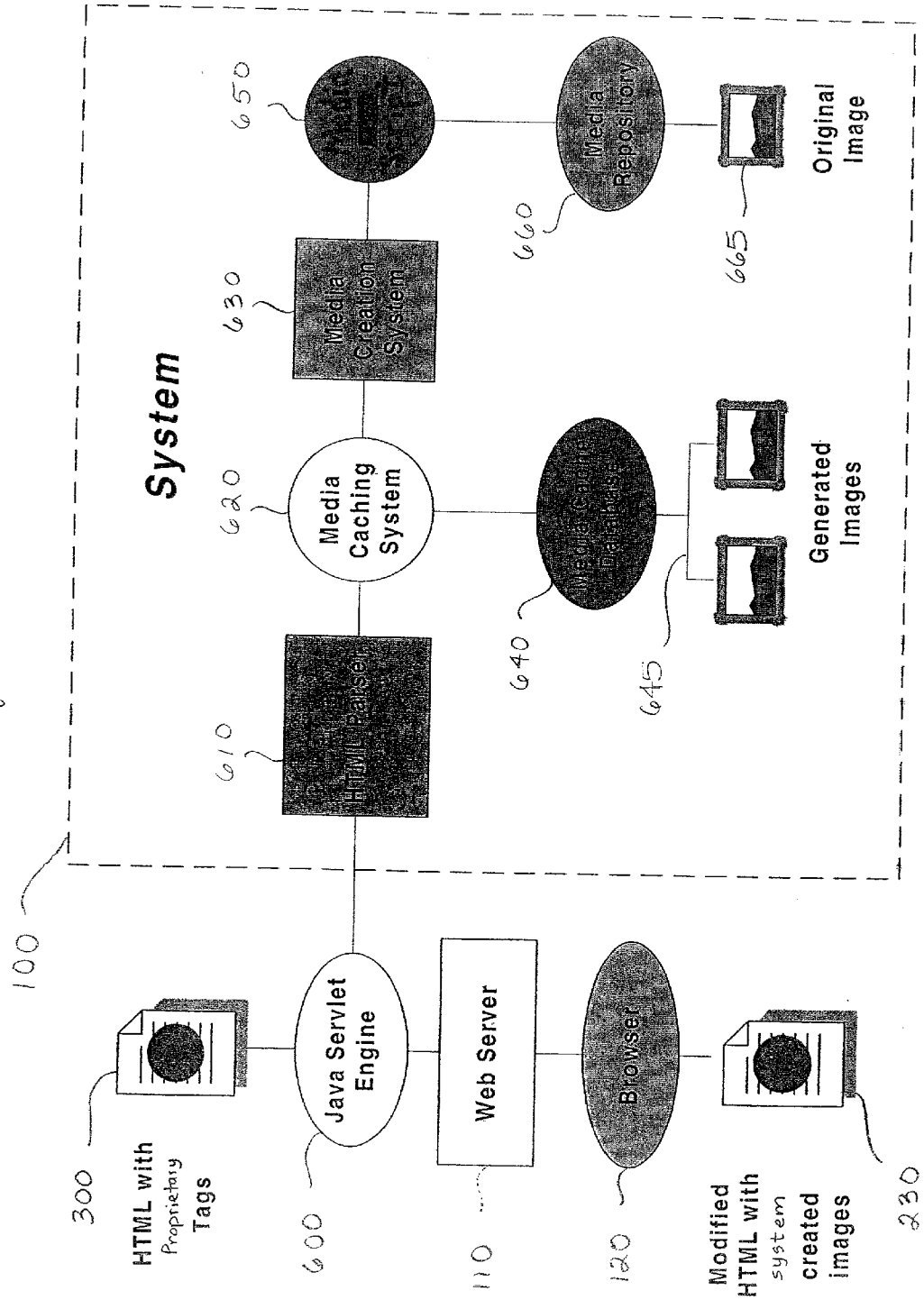

System
100
HTML Parser
610
Media Caching System
620
Media Creation System
630
650
Media Repository
660
Original Image
665
Generated Images
645
640
HTML with Proprietary Tags
300
Java Servlet Engine
600
Web Server
110
Browser
120
Modified HTML with system created images
230

Fig. 6

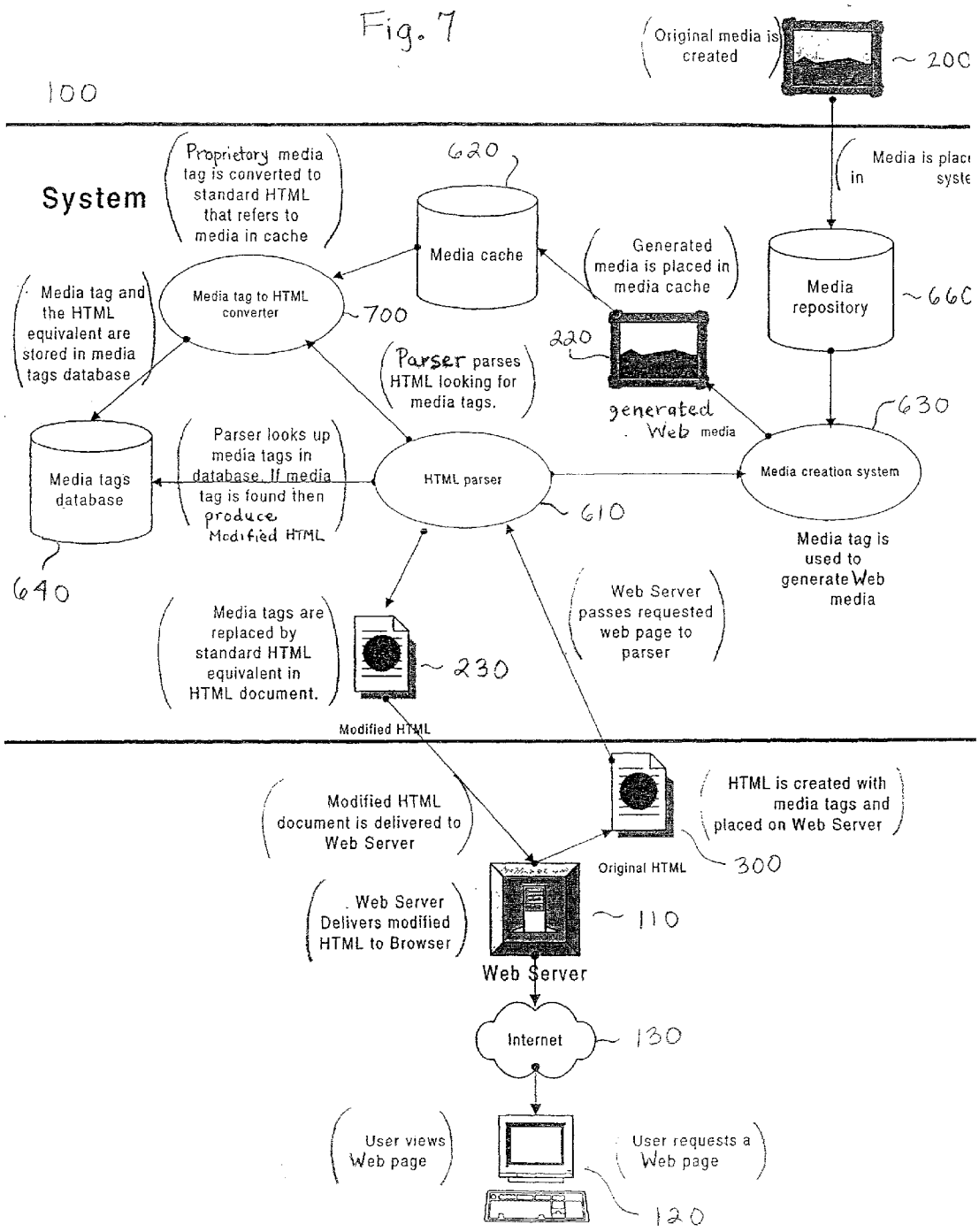


Fig. 7
100
Original media is created
200
System
Proprietary media tag is converted to standard HTML that refers to media in cache
620
Media cache
Generated media is placed in media cache
Media repository
660
Media tag and the HTML equivalent are stored in media tags database
Media tag to HTML converter
700
220
Parser parses HTML looking for media tags.
generated Web media
630
Media tags database
Parser looks up media tags in database. If media tag is found then produce Modified HTML
HTML parser
610
Media creation system
Media tag is used to generate Web media
640
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
HTML is created with media tags and placed on Web Server
Original HTML
300
Web Server Delivers modified HTML to Browser
110
Web Server
Internet
130
User views Web page
User requests a Web page
120

Authoring Flowchart

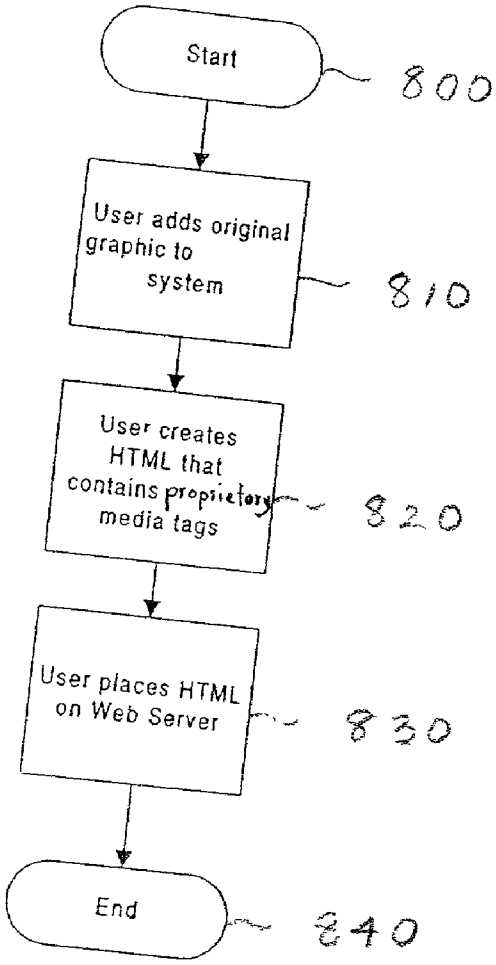


Fig. 8

HTML Parsing Flowchart

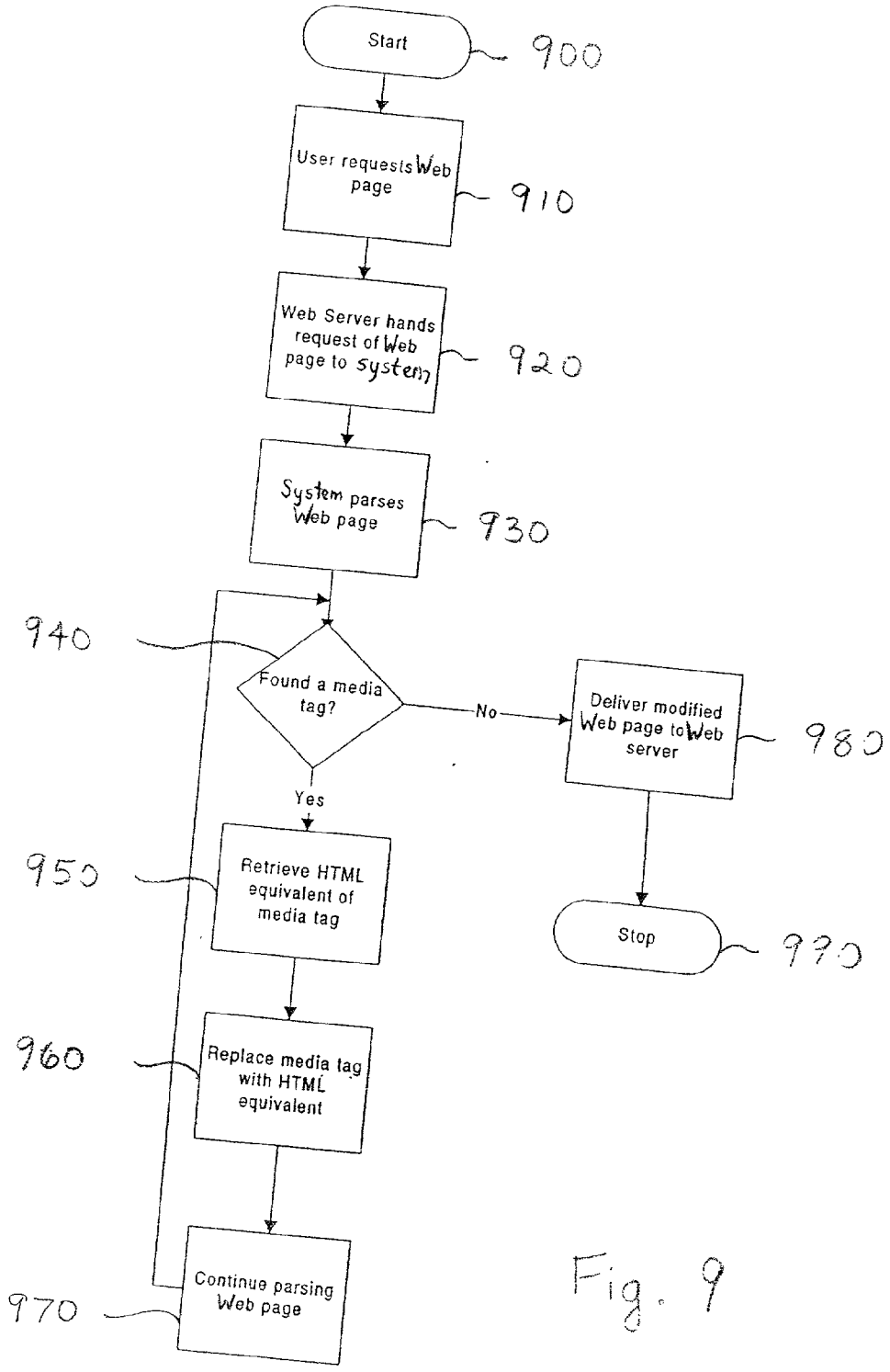


Fig. 9

Me 'ia Creation Flowchart

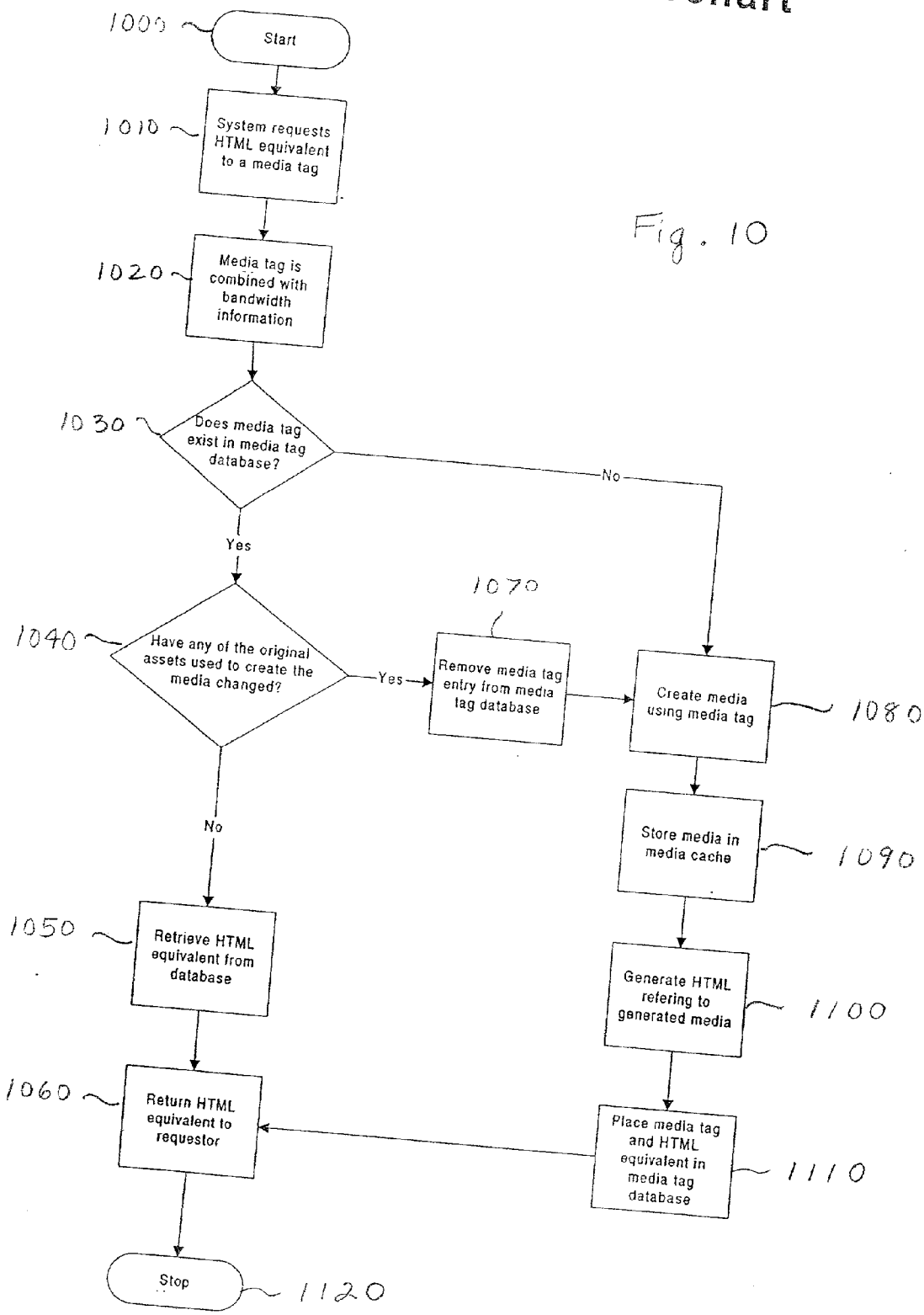


Fig. 10

Fig. 11

File Edit View Go Favorites Window Help

Equilibrium Freeride Administration

Internet Explorer 4.01

Back Refresh Home Search Mail Favorites Larger Smaller Preferences

Address: http://miles.equilibrium.com/Admin/Administration.asp

Live Home Page | Apple Computer | Apple Support | Apple Store | MSN | Office for Macintosh | Internet Explorer

Channels | Favorites | History | Search

POWERED BY FREERIDE

Equilibrium Freeride Administration

Clear the Freeride Database

Check Dependencies

Media Currently in the Freeride Database

MediaScript	Bandwidth	Generated File(s)	Dependencies
#include "includes/board.ms" BoardLogo();	112k	"/ImageServerCache/3264321469_3764525602_3-0.gif"	bgcolor.tga 920674832000
#include "includes/board.ms" BoardLogo();	112k	"/ImageServerCache/3264321469_3764525602_3-0.gif"	bgcolor.tga 920674832000 includes/board.ms 936761026781 page_title.tga 920674836000
var i = new Media(); i.Load(name @ "top_right.tga"); i.Scale(xs @ 232, ys @ 130); i.Save(type @ "jpeg");	112k	"/ImageServerCache/3337740569_3956878100_3-0.jpg"	top_right.tga 920674832000
#include "includes/board.ms" ChooseLogo(1);	112k	"/ImageServerCache/3978861969_3739552276_3-0.gif"	includes/board.ms 936761026781 logo1.tga 920674838000

Internet Zone

Database Description

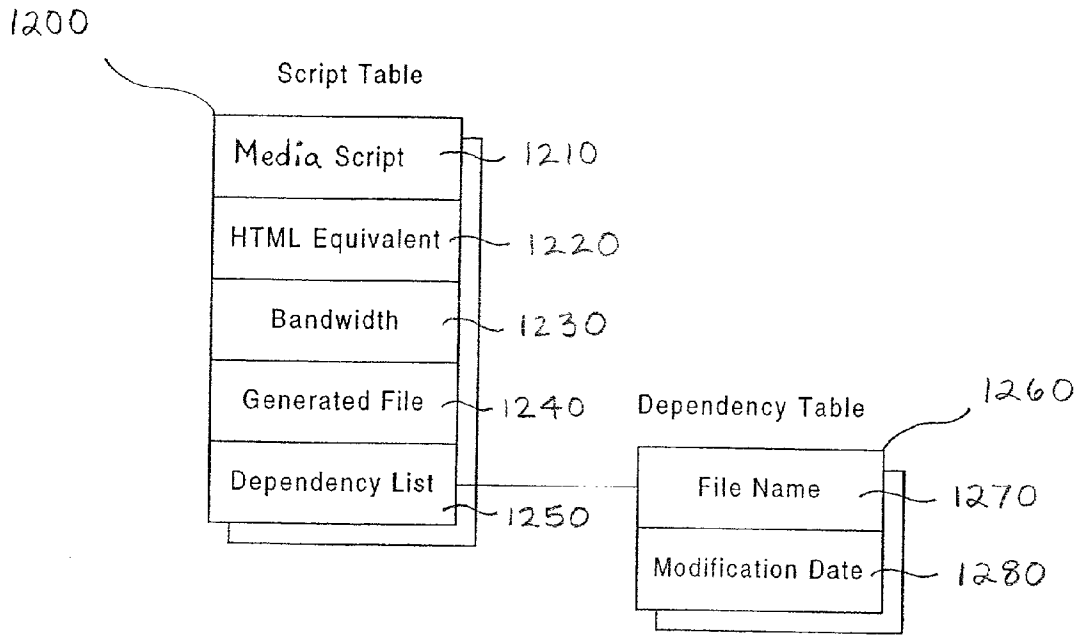
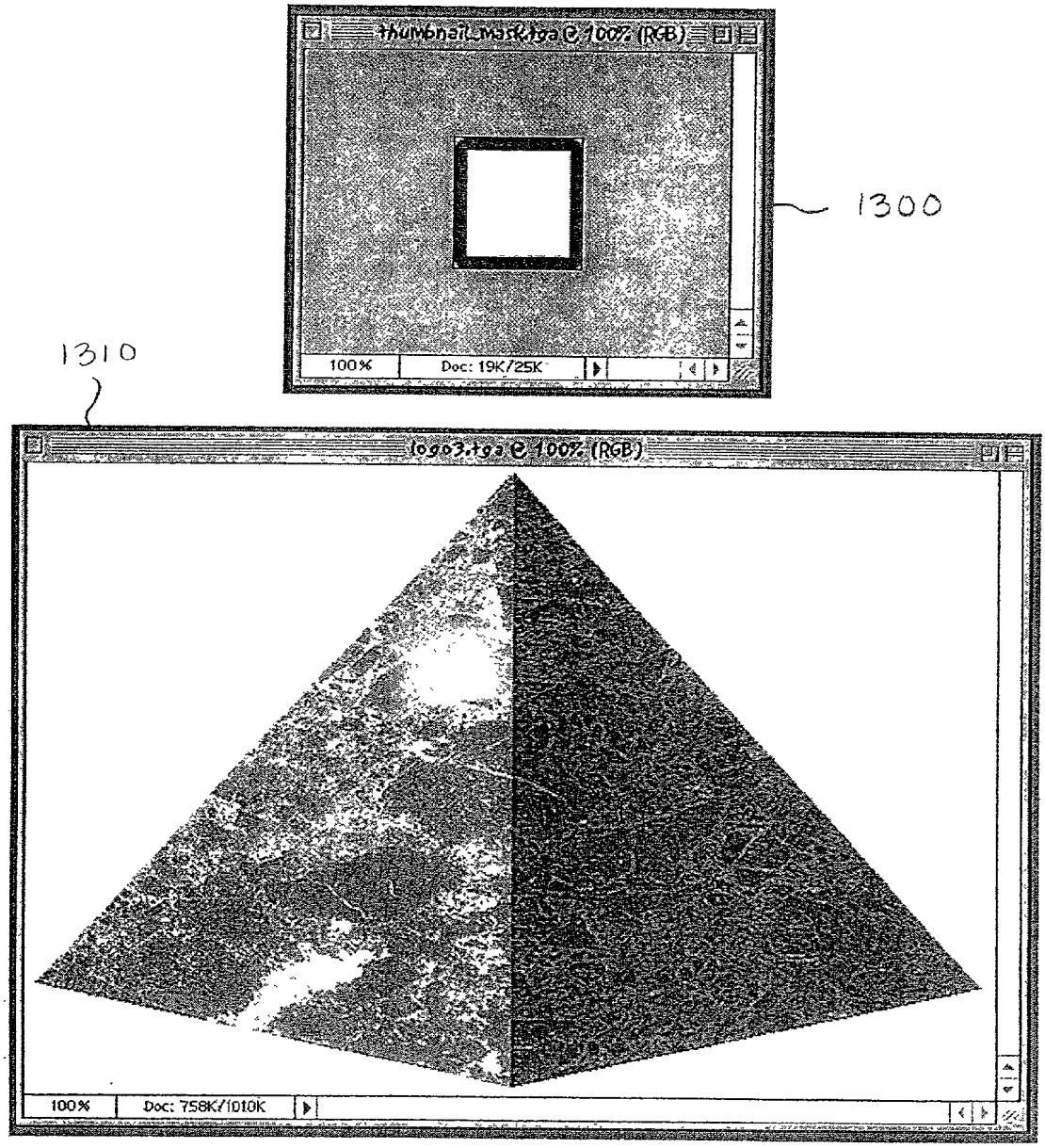


Fig. 12

Fig. 13

Original Images



HTML Document with Proprietary Tag

1400

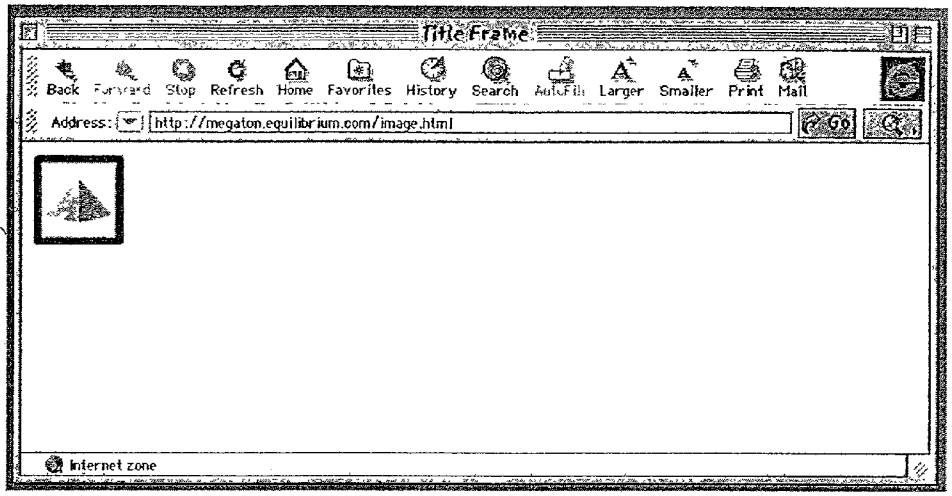
```
image.html
<html>
<head>
<title>
Title Frame
</title>3
</head>
<body>
<imgsrc= <freerideimage>var i = new Media(); i.Load(name @ logo3.tga);
i.Scale(ys @ 65, constrain  @ true, alg @ "smooth",); i.Crop(xs @ 80, ys @ 80,
padcolor @ 0xffffff); var i2 = new Media(); i2.Load(name @
thumbnail_mask,tga); i.Composite(source @ i2); i.Scale(xs @ 60, ys @ 60, alg
@ "smooth"); i.Reduce(); i.Save(type @ "gif"); </freerideimage>
heigth=60 width=60 border=0><br>
</body>
</html>
```

Fig. 14

Fig. 15

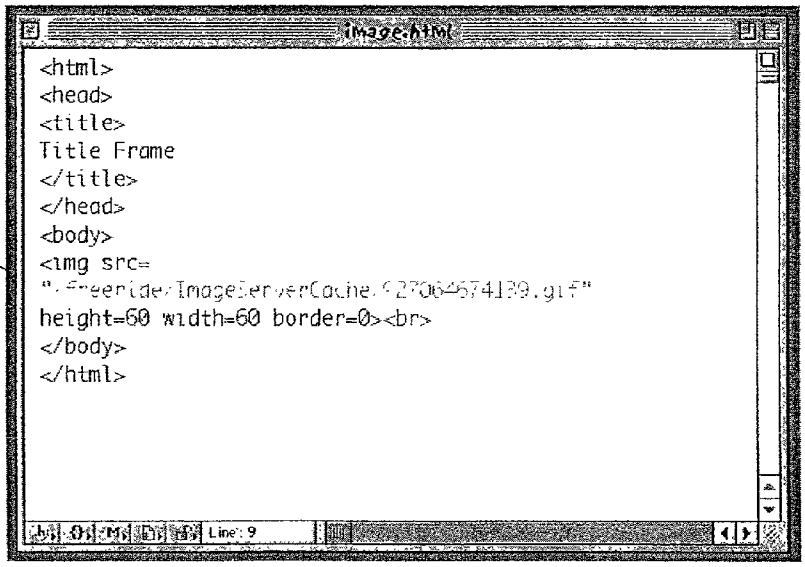
HTML document viewed in browser

1500



HTML document source

1510



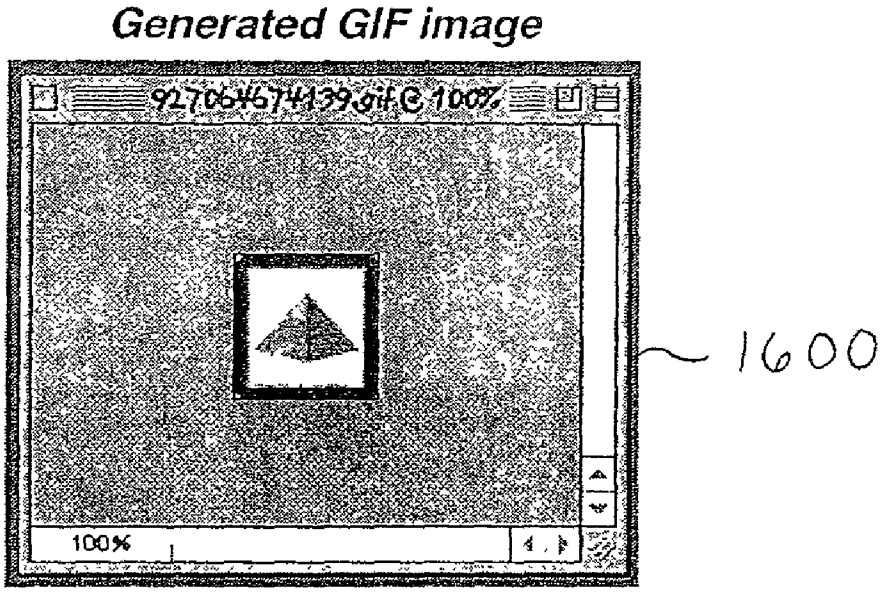


Fig. 16

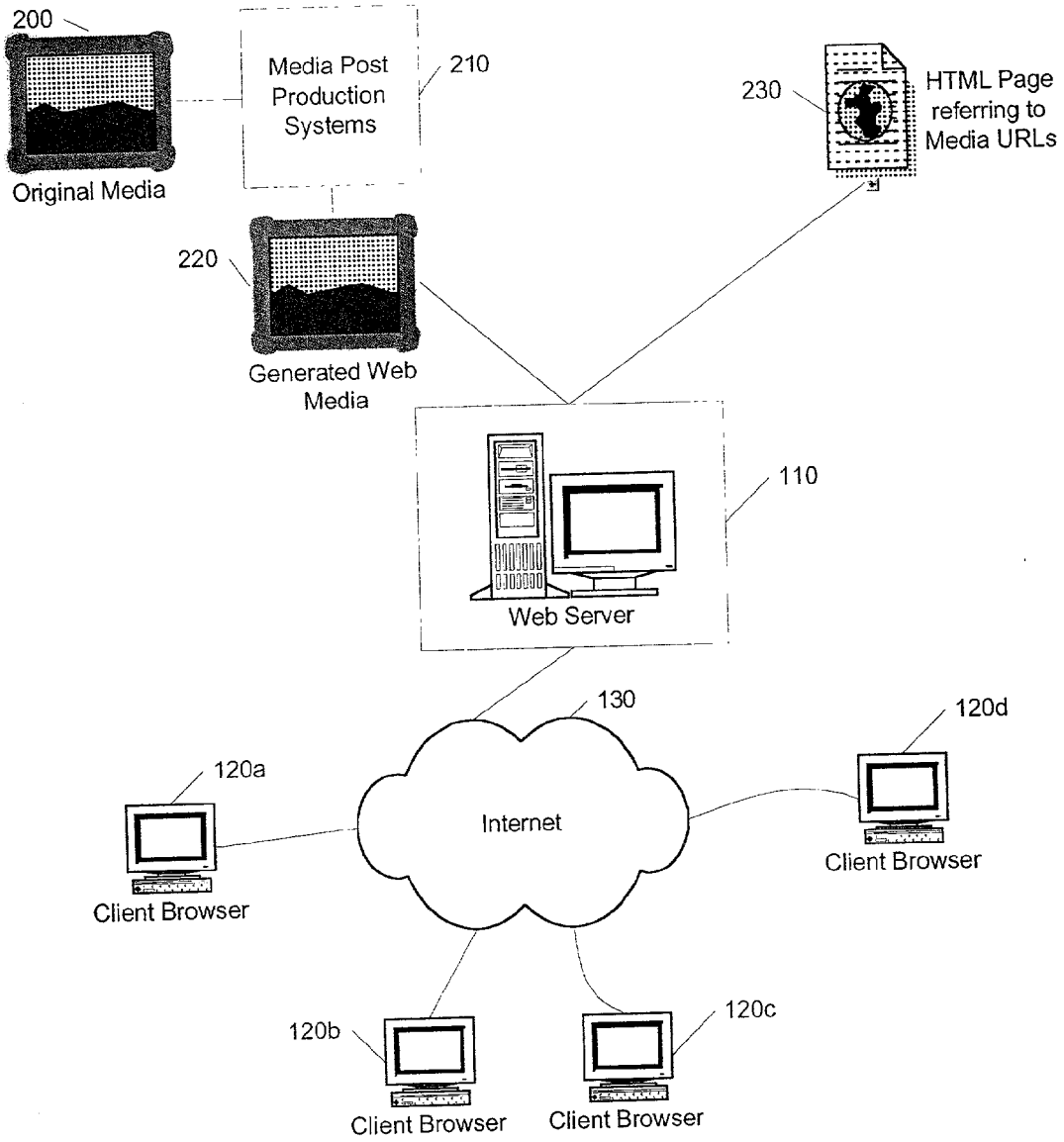

200
Original Media
Media Post Production Systems
210
230
HTML Page referring to Media URLs
220
Generated Web Media
110
Web Server
130
Internet
120a
Client Browser
120d
Client Browser
120b
Client Browser
120c
Client Browser

Fig. 17

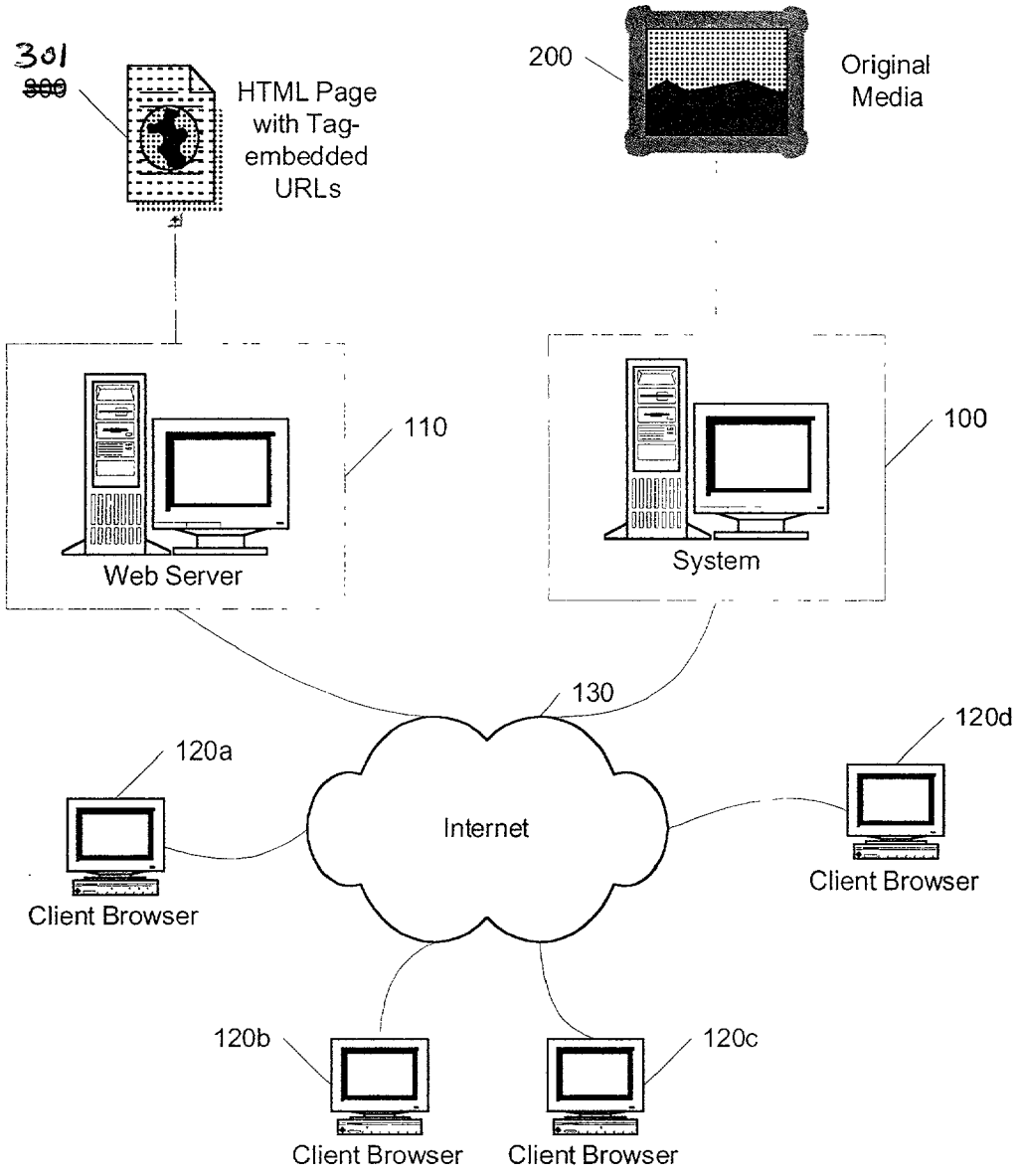

301
~~300~~
HTML Page with Tag-embedded URLs
200
Original Media
110
Web Server
100
System
130
Internet
120a
Client Browser
120d
Client Browser
120b
Client Browser
120c
Client Browser

Fig. 18

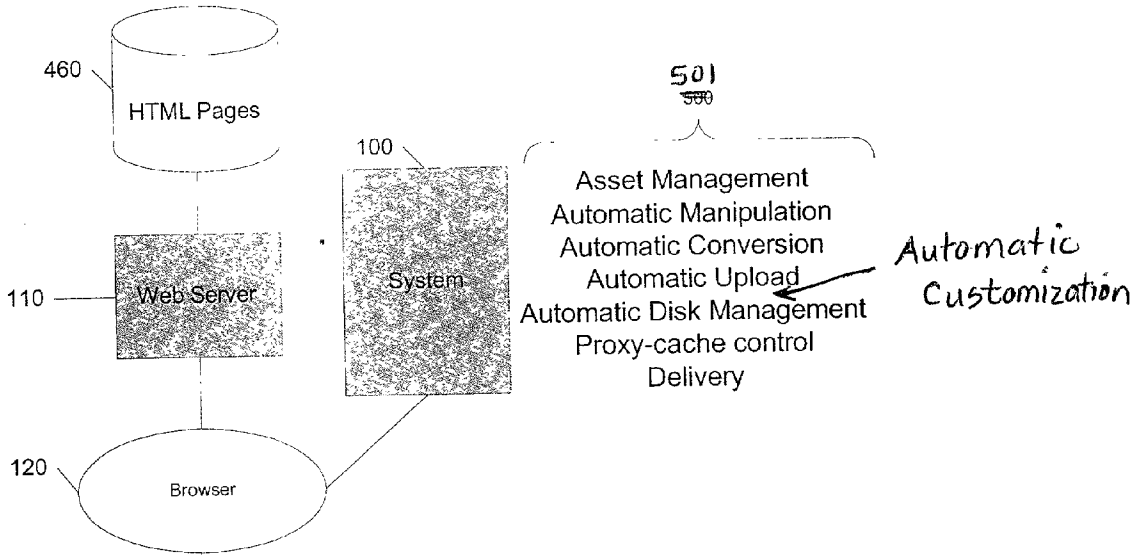

460
HTML Pages
100
System
110
Web Server
120
Browser
501
~~500~~
Asset Management
Automatic Manipulation
Automatic Conversion
Automatic Upload
Automatic Disk Management
Proxy-cache control
Delivery
Automatic Customization

Fig. 19

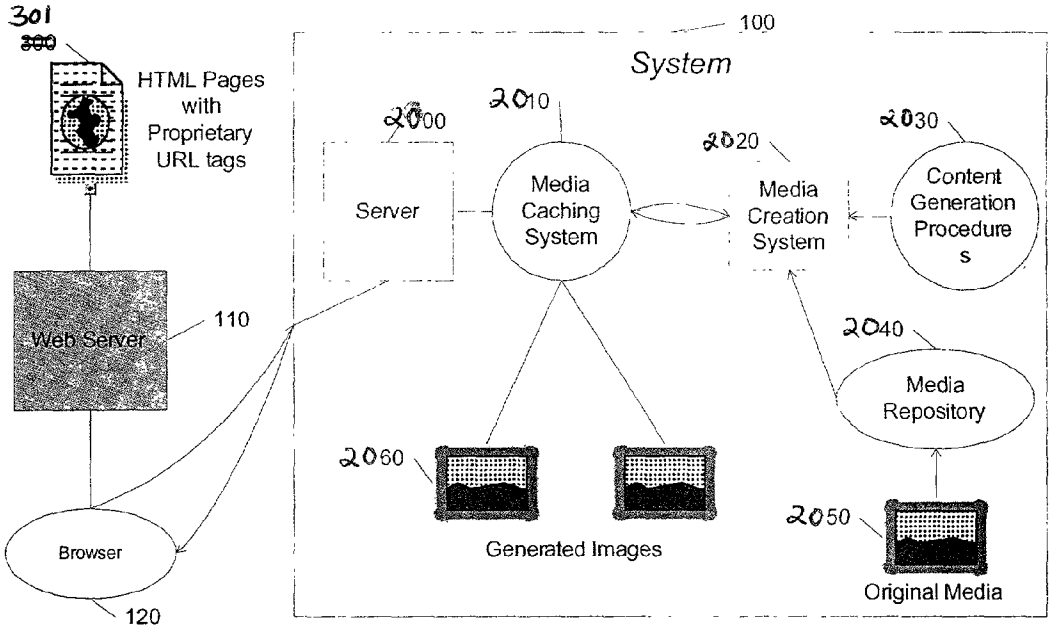
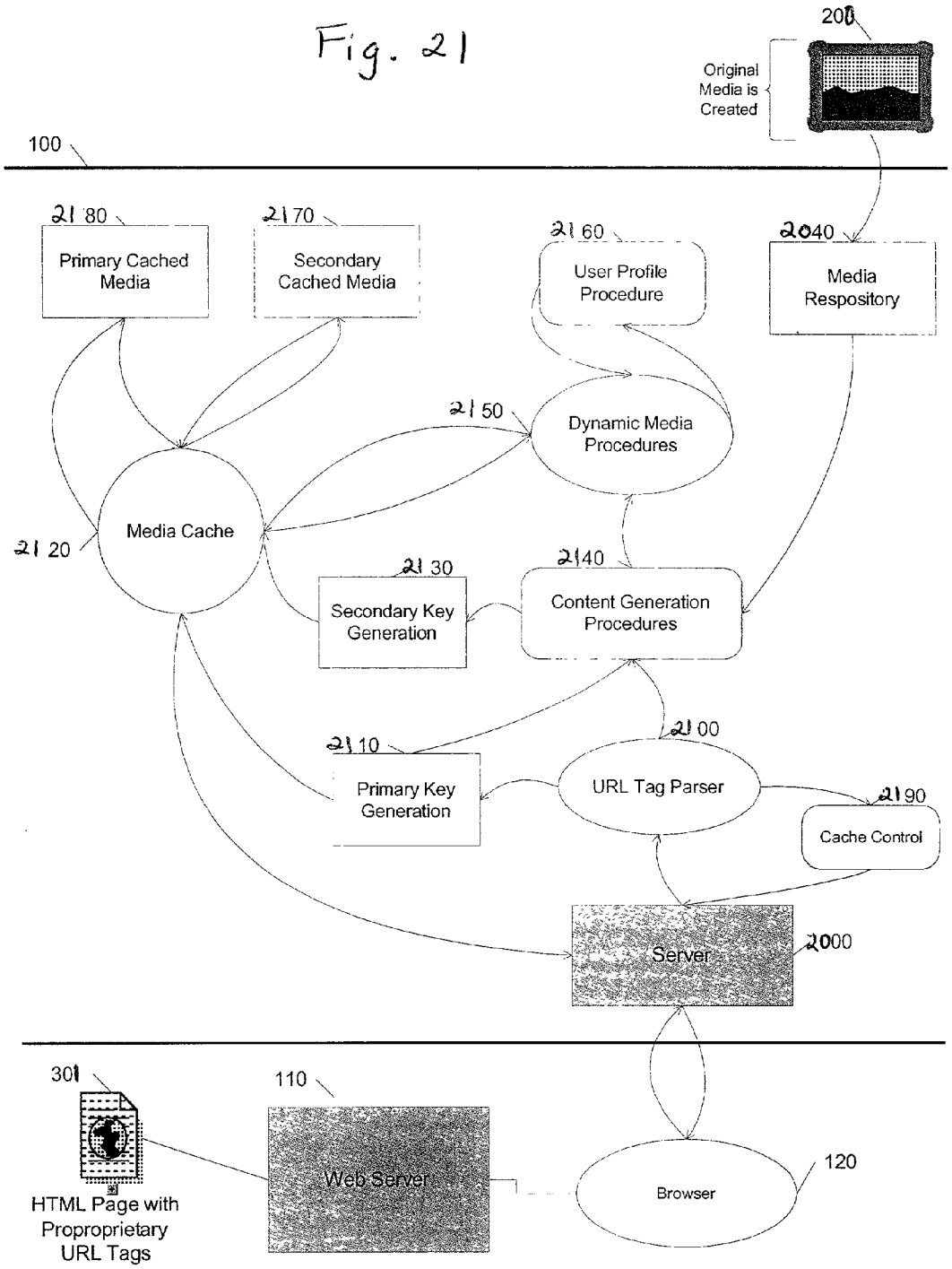

301
HTML Pages with Proprietary URL tags
Web Server
110
Browser
120
100
System
2000
Server
2010
Media Caching System
2020
Media Creation System
2030
Content Generation Procedures
2040
Media Repository
2060
Generated Images
2050
Original Media

Fig. 20

Fig. 21


200
Original Media is Created
100
2180
Primary Cached Media
2170
Secondary Cached Media
2160
User Profile Procedure
2040
Media Respository
2150
Dynamic Media Procedures
2120
Media Cache
2130
Secondary Key Generation
2140
Content Generation Procedures
2110
Primary Key Generation
2100
URL Tag Parser
2190
Cache Control
Server
2000
301
HTML Page with Proproprietary URL Tags
110
Web Server
Browser
120

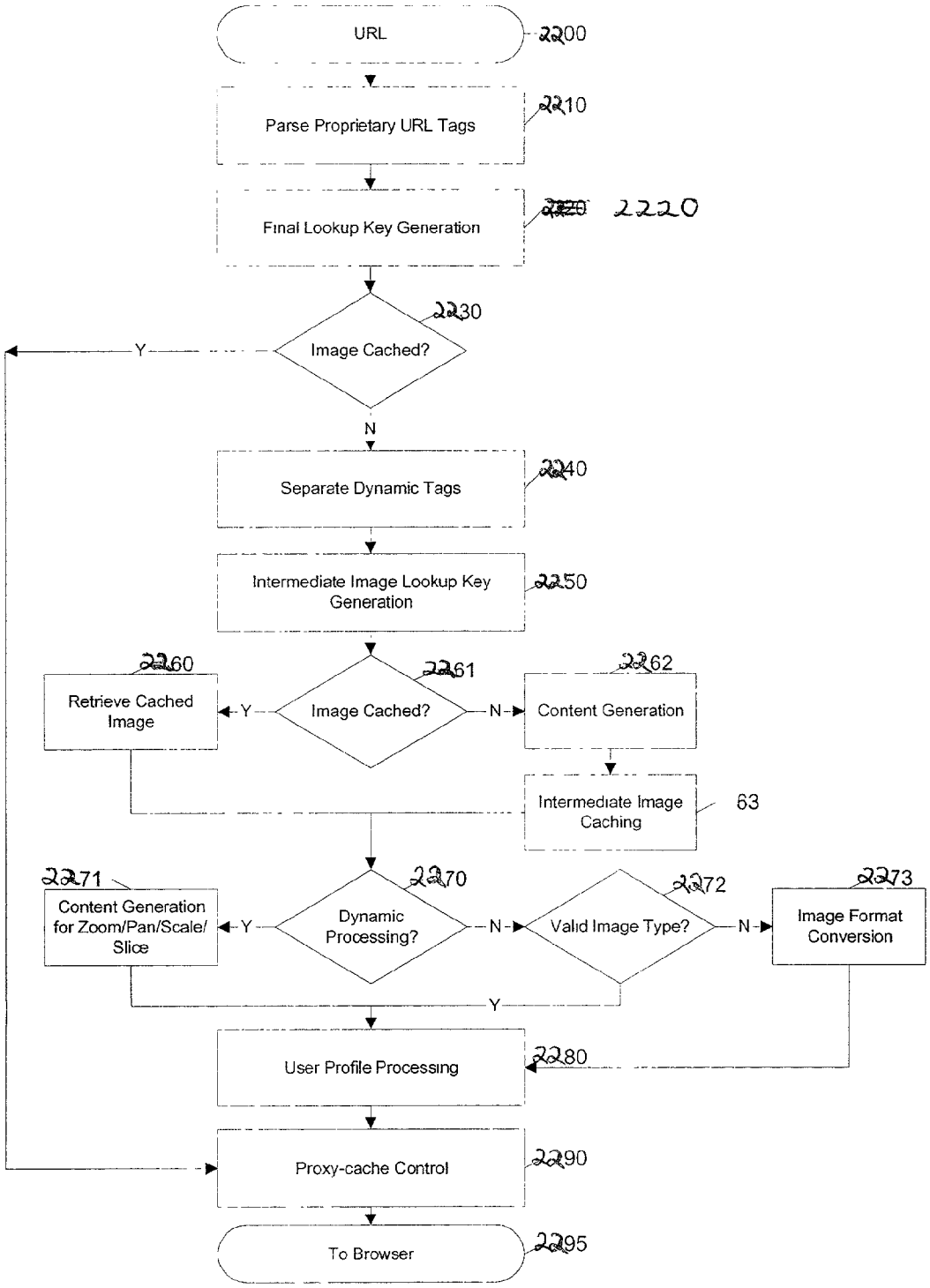

URL
2200
Parse Proprietary URL Tags
2210
Final Lookup Key Generation
2220
Image Cached?
2230
Y
N
Separate Dynamic Tags
2240
Intermediate Image Lookup Key Generation
2250
Retrieve Cached Image
2260
Image Cached?
2261
Content Generation
2262
Intermediate Image Caching
63
Content Generation for Zoom/Pan/Scale/ Slice
2271
Dynamic Processing?
2270
Valid Image Type?
2272
Image Format Conversion
2273
User Profile Processing
2280
Proxy-cache Control
2290
To Browser
2295

Fig. 22

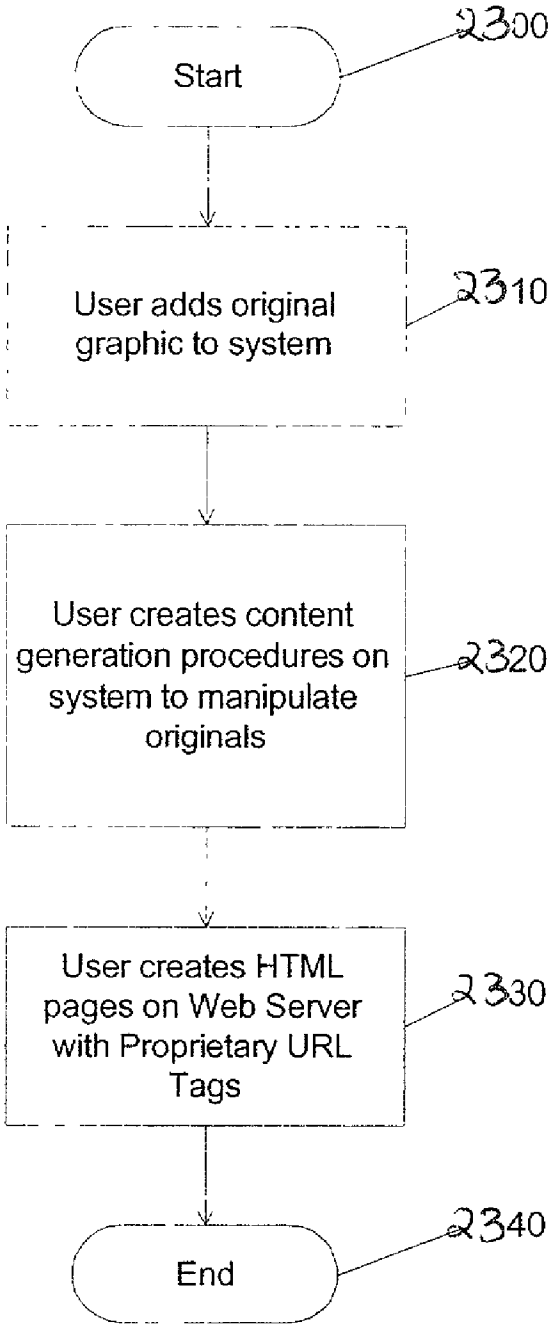

Start
2300
User adds original graphic to system
2310
User creates content generation procedures on system to manipulate originals
2320
User creates HTML pages on Web Server with Proprietary URL Tags
2330
End
2340

Fig. 23

AUTOMATED MEDIA DELIVERY SYSTEM

BACKGROUND OF THE INVENTION

[0001] 1. Technical Field

[0002] 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.

[0003] 2. Description of the Prior Art

[0004] 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.

[0005] 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:

[0006] Current Workflow for Creating Web Graphics

[0007] Original Artwork/Asset Creation

[0008] Use third-party point products

[0009] Asset Editing

[0010] Scale/reduce/slice

[0011] Asset Format Conversion

[0012] JPEG/GIF/PNG

[0013] Asset Staging

[0014] Place in Web file system

[0015] Edit HTML

[0016] Create/Modify HTML for particular page

[0017] Store HTML on Web server

[0018] View final pages

[0019] Repeat process for each version of each graphic on each page

[0020] Estimated time

[0021] Two hours per page times the number of pages

[0022] 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.

[0023] D. C. A. Bulterman, Models, Media, and Motion: Using the Web to Support Multimedia Documents, Proceedings of 1997 International Conference on Multimedia Modeling, Singapore, Nov. 17-20, 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."

[0024] 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."

[0025] 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."

[0026] 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."

[0027] 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."

[0028] 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 by 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."

[0029] 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."

[0030] 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."

[0031] 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.

[0032] 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.

[0033] 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.

[0034] TrueSpectra does not require plug-ins to operate features such as zooming/panning or compositing. 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.

[0035] 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.

[0036] The key to IrisTransactive is the compositing subsystem. It requires three things to build a shopping solution using image composition.

[0037] 1) The original images must be created. It is suggested that the image be converted to Flashpix for better performance.

[0038] 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.

[0039] 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.

[0040] 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.

[0041] MediaBin(TM) is limited to activities behind the firewall automating only the "post-creative busywork." In addition, MediaBin requires the use of an application server to function through a web interface. Thus images may not be directly added to any existing web page.

[0042] Macromedia's Generator operates by embedding variables in their proprietary Flash format. Therefore the

actual imaging operations are somewhat limited and cannot be controlled directly from a web page request.

[0043] MGI Software sells point solutions that require end-users to download a viewer to process a proprietary image format.

[0044] PictureIQ offers a server-side image-processing appliance that provides a limited set of Photoshop functionalities. This appliance runs on the web-page server, processes information embedded in the web page, and rewrites the web page with image data.

[0045] The disclosed prior art fail to provide systems and methodologies 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.

[0046] It would be advantageous to empower an end user with flexibility and control by providing interactive page capabilities.

[0047] 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.

[0048] 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.

[0049] 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.

[0050] 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.

[0051] 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.

[0052] 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.

[0053] It would be advantageous to provide a system that generates media, whereby the media is automatically uploaded.

[0054] It would be advantageous to provide a system that automatically caches generated media so identical requests can be handled without regeneration of images.

[0055] It would be advantageous to provide a system that resides behind the Web server, thereby eliminating security issues.

[0056] It would be advantageous to provide a system wherein the client browser does not require a plug-in.

[0057] It would be advantageous to provide a system wherein the system does not require any changes to a Web server.

[0058] It would be advantageous to provide a system wherein the system manages the Web server media cache.

[0059] It would be advantageous to provide a system wherein the Web media is generated only if requested by a client browser.

[0060] It would be advantageous for a system to reduce the need for a Web author to create different versions of a Web site, the system automatically handling image content.

[0061] It would be advantageous to provide dynamic imaging capabilities, have a more complete set of image processing functionality, and be controlled directly through an image URL.

[0062] It would be advantageous to provide an end-to-end solution requiring only a standard browser that is completely controllable using the proprietary tags contained within a simple image link in the web page.

[0063] It would be advantageous to run an image application as a separate server controlled directly by single image requests to that server, such that any web server, even one that is only sending static HTML can access imaging features.

SUMMARY OF THE INVENTION

[0064] An automatic graphics delivery system that operates in parallel with an existing Web site infrastructure is provided. The system streamlines the post-production process by automating the production of media through content generation procedures controlled by proprietary tags placed within URLs embedded within Web documents. The author simply places the original media in the system, and adds proprietary tags to the URLs for accessing that media. The system automatically processes the URL encoded tags and automatically produces derivative media for the web site from the original media.

[0065] The system takes as input the client connection, server traffic, content generation procedures, and proprietary tags placed within the URL to generate optimized media for the client. The need for the Web author to create different versions of a Web site is reduced because the image content of the site is automatically handled by the system. In addition, generated media is cached such that further requests for the same media require little overhead.

[0066] Because the invention takes the original media, content generation procedures, and proprietary URL tags as inputs for generating the Web media, it is possible to modify any of these inputs and have the system automatically update the media on the associated Web pages.

BRIEF DESCRIPTION OF THE DRAWINGS

[0067] **FIG. 1** is a schematic diagram showing the placement of the system within a current Web infrastructure according to the invention;

[0068] **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;

[0069] **FIG. 3** is a schematic diagram showing delivery of an HTML document and media to a Web browser according to the invention;

[0070] **FIG. 4** is a schematic diagram showing the components involved in Web site administration according to the prior art;

[0071] **FIG. 5** is a schematic diagram showing the components of the system involved in Web site administration according to the invention;

[0072] **FIG. 6** is a simple overview showing the components of the system according to the invention;

[0073] **FIG. 7** is a schematic diagram showing the process flow of a proprietary enabled page delivered to a Web browser according to the invention;

[0074] **FIG. 8** is a flow chart showing an authoring process according to the invention;

[0075] **FIG. 9** is a flow chart showing an HTML parsing process according to the invention;

[0076] **FIG. 10** is a flow chart showing a media creation process according to the invention;

[0077] **FIG. 11** is a screen shot showing an administration tool according to the invention;

[0078] **FIG. 12** displays a structure of a database record used for the system according to the invention;

[0079] **FIG. 13** shows original media to be processed according to the invention;

[0080] **FIG. 14** shows a portion on an HTML document with a proprietary tag according to the invention;

[0081] **FIG. 15** shows an HTML document and an HTML document source according to the invention;

[0082] **FIG. 16** shows a generated GIF image according to the invention;

[0083] **FIG. 17** is a schematic diagram of an image system within a typical Web infrastructure according to the invention;

[0084] **FIG. 18** is a schematic diagram showing delivery of an HTML document and original media according to the invention;

[0085] **FIG. 19** is a schematic diagram showing components of Web site administration according to a preferred embodiment of the invention;

[0086] **FIG. 20** is a simple overview showing components of the image system according to a preferred embodiment of the invention;

[0087] **FIG. 21** is a schematic diagram showing process flow of a proprietary enabled page delivered to a Web browser according to a preferred embodiment of the invention;

[0088] **FIG. 22** shows a flowchart of a content generation procedure according to a preferred embodiment of the invention; and

[0089] **FIG. 23** is a flow chart showing an authoring process according to a preferred embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0090] An automatic graphics delivery system that operates in parallel with an existing Web site infrastructure is provided. The system streamlines the post-production process by automating the production of media through content generation procedures controlled by proprietary tags placed within URLs embedded within Web documents. The author simply places the original media in the system, and adds proprietary tags to the URLs for accessing that media. The system automatically processes the URL encoded tags and automatically produces derivative media for the web site from the original media.

[0091] The system takes as input the client connection, server traffic, content generation procedures, and proprietary tags placed within the URL to generate optimized media for the client. The need for the Web author to create different versions of a Web site is reduced because the image content of the site is automatically handled by the system. In addition, the generated media is cached so that further requests for the same media require little overhead.

[0092] Because the invention takes the original media, content generation procedures, and proprietary URL tags as inputs for generating the Web media, it is possible to modify any of these inputs and have the system automatically update the media on the associated Web pages.

[0093] A detailed description of such automatic media delivery system operating in parallel with existing Web site infrastructure is found below in the section under the heading as such.

[0094] **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**.

[0095] **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**.

[0096] **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**.

[0097] **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**.

[**0098**] **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.

[**0099**] **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 **110** 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.

[**0100**] The System.

[**0101**] A preferred embodiment of the system **100** is provided.

[**0102**] 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.

[**0103**] 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**.

[**0104**] 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**.

[**0105**] 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**.

[**0106**] 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.

[**0107**] The media repository **660** contains original images **665** used in the system **100**.

[**0108**] **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.

[**0109**] **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**).

[**0110**] **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**).

[**0111**] **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**).

[**0112**] 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.

[**0113**] 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.

[**0114**] 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
Blur_Blur( )
This command commands perform a simple 3×3 convolution for blurring.
Blur Convolve More
Blur_MoreBlur( )
This command commands perform a stronger 3×3 convolution for blurring.
Blur Gaussian
Blur_GaussianBlur([radius=<value 0.1..250>])

TABLE B-continued

Media processing script commands
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
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

TABLE B-continued

Media processing script commands
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.

TABLE B-continued

Media processing script commands
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.

[0115] 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; and 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 adjusting the bandwidth of graphics; Optimizes images for client connection speed; Allows clients to specify the quality of images on a Web site; and Allows Web designers to dynamically create images by manipulating proprietary tags in their applications (server or client side).

[0116] **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.

[0117] **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**.

[0118] Snowboard Store Example.

[0119] Background.

[0120] 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 clicks 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.

[0121] Prior Art Method.

[0122] 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.

[**0123**] 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.

[**0124**] The Claimed Method.

[**0125**] 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.

[**0126**] 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.

[**0127**] Processing of an Image Tag Example (**FIG. 13-16**).

[**0128**] **FIG. 13** shows two original images **1300** and **1310** to be processed according to a preferred embodiment of the invention.

[**0129**] **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.

[**0130**] **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.

[**0131**] **FIG. 16** shows a generated GIF image **1600** according to a preferred embodiment of the invention.

[**0132**] Automatic media delivery system operating in parallel with existing Web site infrastructure.

[**0133**] It should be noted that the words, media, graphics, and images are used herein interchangeably.

[**0134**] An automatic graphics delivery system that operates in parallel with an existing Web site infrastructure is provided. The system streamlines the post-production process by automating the production of media through content generation procedures controlled by proprietary tags placed within URLs embedded within Web documents. The author simply places the original media in the system, and adds proprietary tags to the URLs for accessing that media. The system automatically processes the URL encoded tags and automatically produces derivative media for the web site from the original media.

[**0135**] The system takes as input the client connection, server traffic, content generation procedures, and proprietary tags placed within the URL to generate optimized media for the client. The need for the Web author to create different versions of a Web site is reduced because the image content of the site is automatically handled by the system. In addition, the generated media is cached so that further requests for the same media require little overhead.

[**0136**] Because the invention takes the original media, content generation procedures, and proprietary URL tags as inputs for generating the Web media, it is possible to modify any of these inputs and have the system automatically update the media on the associated Web pages.

[**0137**] A preferred embodiment of the invention is described with reference to **FIG. 17**. **FIG. 17** is a schematic diagram of an image system within a typical Web infrastructure according to the invention. The image system **100** is placed in parallel to an existing Web server **110**. The image system **100** may be on-site or off-site to the Web infrastructure. Multiple client browsers **120***a*-**120***d* communicate with both the Web server **110** and the image server **100** via the Internet **130**.

[**0138**] The delivery of an HTML document and media according to a preferred embodiment of the invention is described with reference to **FIG. 18**. Resource locators (URLs) are placed within HTML documents **301** accessible to the Web server **110**. These URLs direct browsers to generate requests for media to the system **100**. The system processes such URLs by interpreting the proprietary tags, executing the indicated image generation procedures on the original media **200**, and returning derivative Web-safe media to client browsers **120***a*-**120***d* via the Internet **130**. Additionally, such generated media is cached on the image server **100** and, therefore need not be regenerated for subsequent requests.

[**0139**] Web site administration according to another preferred embodiment of the invention is described with reference to **FIG. 19**. **FIG. 19** is a schematic diagram showing the components of Web site administration according to a preferred embodiment of the invention, whereby Web site administration is simplified. The preferred embodiment provides, but is not limited to the following services: asset management, automatic image manipulation, automatic image conversion, automatic image upload, automatic image customization based on browser characteristics, automatic disk management, automatic control of proxy caching, and image delivery **501**.

[**0140**] **FIG. 20** is a simple overview showing components of the system according to a preferred embodiment of the invention. HTML pages with proprietary URL tags **301** describe how referenced media therein is to be manipulated for Web. Browsers **120** send such tags to the image system **100** as media requests. A server **2000** within the image system **100** receives the media requests, decodes the URL tags, and retrieves any media that already exists in the media caching system **2010**. Non-existent media is subsequently generated by a media creation system **2020** using original media **2050** stored in a media repository **2040** and using content generation procedures **2030**.

[0141] The Image System.

[0142] Following is a detailed description of the preferred embodiment of the invention with reference to **FIG. 21** below.

[0143] The system receives a request for media through a URL containing proprietary tags for controlling image generation. The system parses this URL to determine the content generation procedure to execute, input to the content generation procedure, post-processing directives for, for example, zoom/pan/slice, browser properties, and any cache control directives. Such data is handed to a media caching subsystem that returns the requested image if found. If the image is not found, the information is handed to the media generation subsystem that executes the specified content generation procedure to produce a derivative image. The media generation subsystem returns one or more images to the media caching system for subsequent reuse.

[0144] The media caching subsystem is a mechanism for associating final or intermediate derivative media with the procedure, input, and user characteristics used to generate said media, specified through proprietary tags within the requested URL. This system may be implemented using a database, file system, or any other mechanism having capability to track such associations.

[0145] The media generation subsystem executes a primary content generation procedure to produce a derivative image whose identifier is provided to the media caching subsystem. This derivative image is composed of one or more original images acquired from the media repository. This media is then passed to the dynamic image content system, if necessary, to generate a subsequent derivative media suitably modified for the needs of zooming, panning, or slice. The resulting media is passed to the user profile system where it is again modified to account for any specific user browser characteristics specified using the proprietary URL tags. This media is then returned to the browser, along with any cache control directives encoded within the URL, and its identifier is passed to the media caching system for subsequent retrieval.

[0146] The dynamic content system operates on intermediate derivative images to generate image subsets or scalings used by Web site designers to implement zooming in on an image, panning across an image, slicing an image into parts, and the like for special Web page effects. The input to this system is cached by the media caching system such that the intermediate image need not be regenerated.

[0147] The user profile system operates on the final image about to be returned to the browser and may modify the image to account for individual needs of Web site users. The designer of a site is able to implement freely custom post-processing of images to meet the specific needs of their clients.

[0148] **FIG. 21** 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. Original media **200** is created and placed into the system **100** in a media repository **2040**. A content generation procedure **2140** is created with instructions on how the media is to be transformed to create the desired Web page content. An HTML page **301** is created for the Web site comprising the system **100**, the page containing one or more URLs directing a browser **120** to request the specified content generation procedure **2140** from the system **100** using input parameters specified with proprietary tags encoded within the URL. The browser **120** requests the Web page **301** from the Web site **110**. Upon receipt of the page **301**, the browser contacts the system **100** requesting media specified in the URL. The system parses the URL **2100** to determine the content generation procedure **2140** to execute, any corresponding input parameters to be used by such procedure, any dynamic content processing **2150** to be performed by dynamic media procedures, any user profile information **2160** to be used to modify the resulting image, and any cache control HTTP headers **2190** the site instructs to accompany the resulting image.

[0149] The parser generates a unique primary lookup key **2110** for the specified resulting media. If the key corresponds to an existing generated media **2180**, such media is returned immediately to the browser **120** through a media cache **2120**, and the transaction is complete. Otherwise, a media generation occurs. In the case of media generation requiring dynamic content processing, a unique secondary lookup key corresponding to intermediate media is generated **2130**. If intermediate media **2170** corresponding to this key is found, such media is passed directly to the dynamic media content system **2150** having dynamic media procedures, wherein appropriate action is taken to generate the required derivative from the intermediate media data. A unique key is generated for the derivative **2130** and passed to the media caching system **2120**. If the media caching system finds no such intermediate image, such intermediate image is generated according to instructions specified by the content generation procedure, cached by the media cache system **2120** as a secondary cached media **2170**, and passed to the dynamic media system **2150**. Again, appropriate action is taken to generate the required derivative from the intermediate image data.

[0150] The resulting image after any dynamic media processing is complete, is checked to ensure that the image is in a valid Web image format. If not, the image is automatically converted into a valid format.

[0151] The final media is passed to a user profile system **2160** wherein browser characteristics specified through proprietary tags within the URL are inspected, and appropriate modification to the media is performed, based on such characteristics. The resulting image is handed to the media cache system **2120** for caching and returned to the browser **120**.

[0152] **FIG. 22** shows a flowchart of the content generation procedure according to a preferred embodiment of the invention. A URL containing proprietary tags **(2200)** is parsed **(2210)** to determine the content generation procedure to execute, any dynamic modifications to the media, user profile characteristics, and proxy-cache control. A unique final lookup key is generated for the media **(2220)** and the media cache is checked **(2230)**. If the indicated media exists, control passes to proxy-cache control **(2290)** and the media is delivered to the browser **(2295)**. Otherwise, dynamic media system tags are separated from content generation control tags **(2240)** and a unique intermediate image lookup key is generated **(2250)**. The cache is then checked for such intermediate media **(2261)**. If such intermediate media is found, it is used directly for dynamic processing, if required.

Otherwise, content is generated (**2262**) and cached (**2263**), and the result is evaluated for dynamic processing (**2270**). If dynamic processing is required, the media is operated upon by the dynamic content generator (**2271**), otherwise it is evaluated for valid content type (**2272**). If the content type is invalid, the media is automatically converted to a valid type (**2273**). The resulting image is then customized by the user profiling system (**2280**) for specified browser or client attributes. Finally, any cache-control directives specified are attached to the response (**2290**) and the media is delivered to the browser (**2295**).

[**0153**] **FIG. 23** is a flow chart showing an authoring process according to a preferred embodiment of the invention. The process starts (**2300**) when a user adds an original graphic or other media (**2310**) to the system. The author then creates a content generation procedure (**2320**) containing instruction on how the original media should be processed to generate the desired Web page content. The user then creates an HTML document (**2330**) that refers to that image by using a URL pointing to a content generation procedure on the image server. The system ends (**2340**). The authoring subsystem assists the Web site designer with choosing parameters and with designing the content generation procedure such that the desired Web site graphic is obtained.

[**0154**] It should be appreciated that differences exist between specifying an image with a URL and requesting an image using a content creation process that interprets proprietary parameters encoded within a URL. That is, URLs allow Web site designers to load specific graphic images into a Web page. In contrast and according to the invention, URLs containing proprietary content creation tags initiate a process whereby graphic images for a site are automatically produced.

[**0155**] Table D below is a list of example proprietary URL tags used for content generation within the system according to the preferred embodiment of the invention. Additional tags may be added to the system as necessary.

TABLE D

Tags
f=function Names the content creation procedure used to generate all or part of the desired graphic.
args=arguments Supplies page dependent parameters used to control the content creation procedure from within the Web page.
cr=crop rectangle Specifies that portion of the image generated by the content generation procedure to be returned to the browser.
st=slice table Specifies a rectangular grid to be placed over the image produced by the content generation procedure, each portion of which can be returned to the browser.
sp=slice position Specifies that portion of the slice table grid placed over the image generated by the content creation procedure to be returned to the browser.
is=image size parameter Specifies scale factors to be applied to any portion of an image generated by any combination of a content generation procedure, arguments, crop rectangles, slice tables, and slice positions.
p=user profile string Specifies a user profile identifier used to modify the final image prior to returning the image to the browser, thus allowing clients to modify the image returned to the browser to account for individual browsing conditions.
c=cache control Specifies a proxy-cache control string to accompany the returned image within an HTTP header.

[**0156**] Table E below is a list of example supported content creation commands according to a preferred embodiment of the invention. Additional commands may be added as necessary.

TABLE E

Content Creation Commands
Adjust HSB Allows the HSB of an image to be altered.
Adjust RGB Allows the contrast, brightness, and color balance of an image to be altered.
Colorize Alters the hue of the pixels in the image to that of the specified color.
Brush Composite Composites the specified brush image onto the current target image.

TABLE E-continued

Content Creation Commands
Convert Converts the rasters to the specified type/bit-depth.
Crop Crops the media to the specified size.
Dropshadow Adds a drop shadow to the image, based on the alpha-channel.
Equalize Performs an equalization on the relevant components of the media.
FixAlpha Adjusts the RGB components of an image relative to its alpha-channel.
Flip Flips the media vertically or horizontally.
Glow Produces a glow or halo around the image
Load Loads in a media from the specified file.
Normalize Similar to equalize, but for audio.
Reduce Reduces the image to a specified palette.
Rotate Rotates the media clockwise by the specified angle in degrees.
Save Saves the media to the specified file.
Scale Scales the media to the specified size.
SetColor Allows the background color, foreground color, and transparency state of the media to be set.
Text Drawing Composites the specified text onto the image.
Text Making This command, instead of compositing text onto the target, creates a new image that just encloses the text.
Zoom Zooms in on a specified portion of the media, and fits it to the specified size. Effectively this constitutes a crop followed by a scale.

[0157] Table F below lists comprises some, and is not limited to all major features of a preferred embodiment of the invention. Additional features may be added as necessary.

TABLE F

System Features
Reads and writes various file formats;
Supports many image processing operations;
Dynamically creates Web images from original assets;
Dynamically creates thumbnail images;
Dynamically and efficiently creates images that can be panned, zoomed, or sliced from original assets without Browser plugins;
Automatically propagates changes in original assets throughout the Web site;
Uses an intelligent caching mechanism for both final and intermediate graphics, comprising: Clean up cache on demand; Eliminates orphaned Web files; and Optimizes Web server cache by providing most recent images;
Renders True-Type fonts on server instead of browser;
Uses intelligent scaling of line drawings;
Allows Web designers to manipulate images using a combination of content generation procedures and proprietary URL tags;
Preserves original image assets;
Optimizes Web server traffic by adjusting the bandwidth of graphics;
Optimizes images for client connection speed;
Allows clients to specify the quality of images on a Web site:
Allow site-specific customized image optimizations for a variety of purposes; and
Allows Web designers to dynamically create images by manipulating proprietary URL tags in applications.

[0158] 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.

1. A method for generating and delivering Web page content comprising media, said method comprising:

parsing a URL containing proprietary tags to determine:

a content generation procedure to execute and corresponding input to said procedure;

dynamic modifications to perform on said media;

user profile characteristics; and

proxy-cache control;

generating a unique final lookup key for said media;

checking a media cache, using said final lookup key;

if said media exists in said media cache, then

passing control to said proxy-cache control; and

delivering said media;

if said media does not exist in said media cache, then

separating dynamic media system tags from content generation control tags; and

generating a unique intermediate image lookup key;

checking said media cache using said intermediate image lookup key;

if said intermediate media exists in said media cache, then

using said intermediate media for further processing;

if said intermediate media does not exist in said media cache, then

generating and caching said media, using said intermediate image lookup key;

determining if dynamic processing is required and, if affirmative, then

operating upon said media by a dynamic content generator;

determining if content type is valid and, if negative, then

converting said media automatically to a valid type;

customizing said media for specified browser or client attributes using a user profiling system;

attaching any specified cache-control directives to a response; and

delivering said media.

2. The method of claim 1, wherein dynamic processing comprises directives for zoom, pan, slice, and the like.

3. A method for an end user to generate Web page content comprising media, said method comprising:

adding original media to an image system;

creating a content generation procedure containing instruction for processing said original media for said Web page content; and

creating an HTML document referring to said processed media by using a URL pointing to said content generation procedure on said image system.

4. The method of claim 3, further comprising:

providing an authoring subsystem, said subsystem assisting with choosing parameters and with designing said content generation procedure.

5. An apparatus for generating Web-safe media, using an HTML page having a URL, said URL having encoded proprietary tags, said apparatus comprising:

a server for receiving media requests and delivering said Web-safe media;

a media repository for storing original media;

a content generation procedure containing instructions for transforming said original media into said Web-safe media;

a URL tag parser for determining:

said content generation procedure to execute and any corresponding input parameters to be used by said procedure for generating a primary media to be cached;

dynamic content processing to be performed, if necessary by dynamic media procedures;

user profile information, if any to be used for modification of a resulting image; and

cache control HTTP headers, if any to accompany said resulting image;

a unique primary lookup key generated by said parser and associated with said primary cached media, wherein said primary cached media, if existing, is delivered as said Web-safe media through a media cache; and

a unique secondary lookup key corresponding to intermediate media requiring said dynamic media processing by said dynamic media procedures, thereby generating corresponding derivative intermediate media, and said unique secondary lookup key corresponding to said derivative intermediate media stored in a secondary media cache.

6. The apparatus of claim 5, wherein any of said original media, said content generation procedure, and said proprietary URL tags are modified, thereby rendering an automatically updated Web-safe media.

7. The apparatus of claim 5, wherein said apparatus is configured in parallel to a Web server infrastructure, said existing Web server infrastructure providing said media requests.

8. The apparatus of claim 7, wherein said apparatus is configurable to work either on-site or off-site to said Web server infrastructure.

9. The apparatus of claim 5, wherein said primary cached media is composed of one or more original media images acquired from said media repository.

10. The apparatus of claim 5, wherein said derivative intermediate media is suitably modified for the needs of zooming, panning, slicing, or the like.

11. An apparatus for delivering an HTML document and derivative Web-safe media, comprising:

means for placing within said HTML document universal resource locators, herein referred to as URLs, said URLs having proprietary tags, and said HTML document accessible to a Web server by a browser;

means for through a browser said URLs generating a request for media from an image system;

means for said image system:

processing said URLs by interpreting proprietary tags;

executing an image generation procedure on said media, said procedure indicated within said tags; and

returning derivative Web-safe media and said HTML document.

12. The apparatus of claim 11, wherein said generated media is cached on said image system, and, whereby upon subsequent requests for said cached media, said image system retrieves and delivers said cached media.

13. The apparatus of claim 11, wherein original media is stored in a media repository.

* * * * *