US006075528A

United States Patent [19]

Curtis

| [11] | Patent Number: | 6,075,528 |
| [45] | Date of Patent: | Jun. 13, 2000 |

[54] **GRAPHICAL USER INTERFACE STREAM PROCESSOR**

[75] Inventor: **Bryce Allen Curtis**, Round Rock, Tex.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **08/954,647**

[22] Filed: **Oct. 20, 1997**

[51] Int. Cl.^{7} G06F 15/00
[52] U.S. Cl. 345/333; 345/356; 707/513; 707/203; 709/220; 709/240
[58] Field of Search 345/333, 356; 707/513, 203; 709/220, 246

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,802,530   9/1998   Van Hoff 707/513

Primary Examiner—Ba Huynh

Attorney, Agent, or Firm—Robert V. Wilder; Leslie A. Van Leeuwen

[57]   **ABSTRACT**

A method and implementing computer system is provided in which a function or applet for example may be downloaded from a network server and graphical user interface (GUI) processor interprets a data stream consisting of GUI controls and attributes, and places the controls and attributes in a presentation space. In an exemplary embodiment, a web browser program loads an HTML (Hypertext Markup Language) from a GUI (Graphical User Interface) stream processor applet from a web server. The web browser then loads the GUI stream processor (GSP) from the web server and begins executing the GSP. The GSP then requests the GUI stream from the web server by opening a stream to a file, CGI (Common Gateway Interface) script or servlet on the server. The GSP then receives the GUI stream and assembles the user interface in the JAVA applet presentation space managed by the browser. The GSP may periodically update itself by repeating the last two steps.
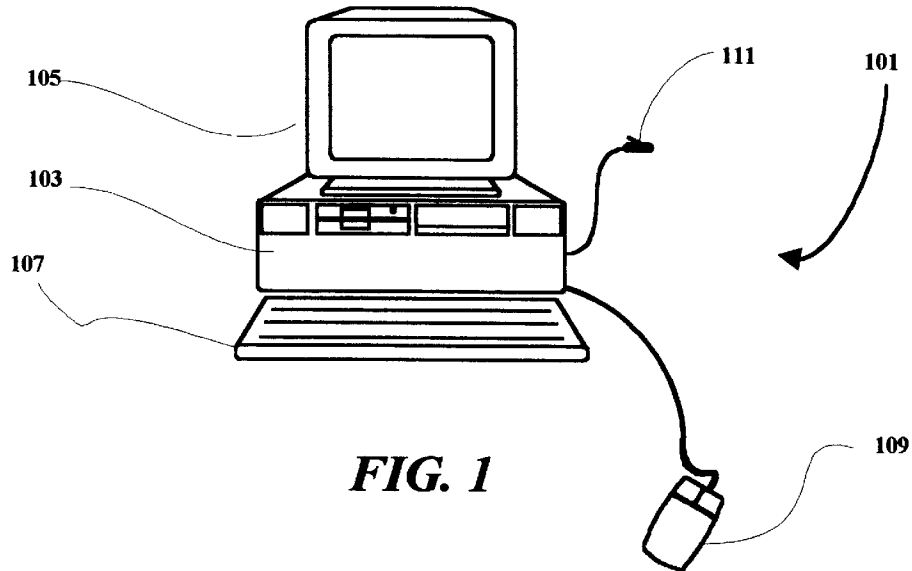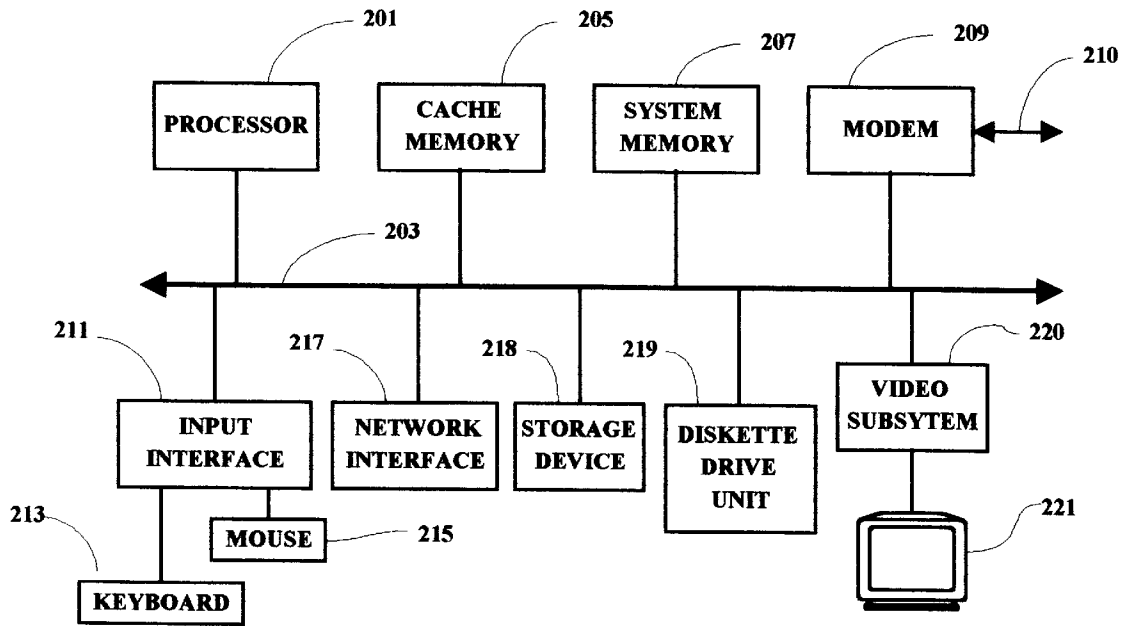
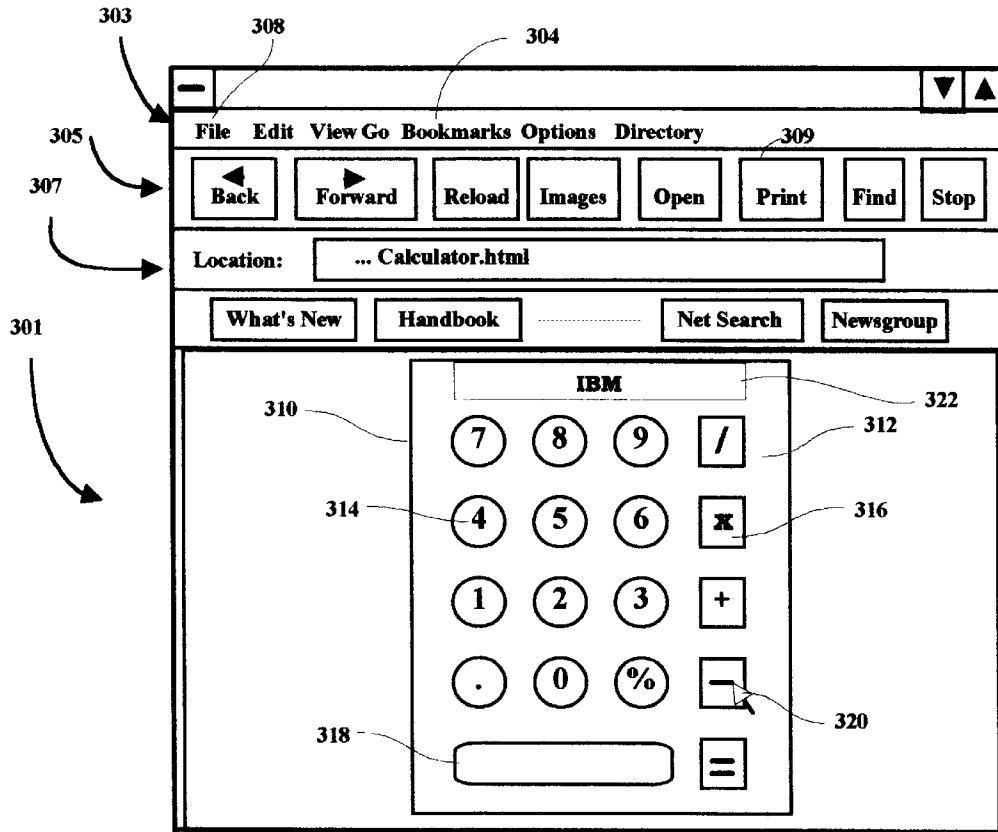**15 Claims, 2 Drawing Sheets**
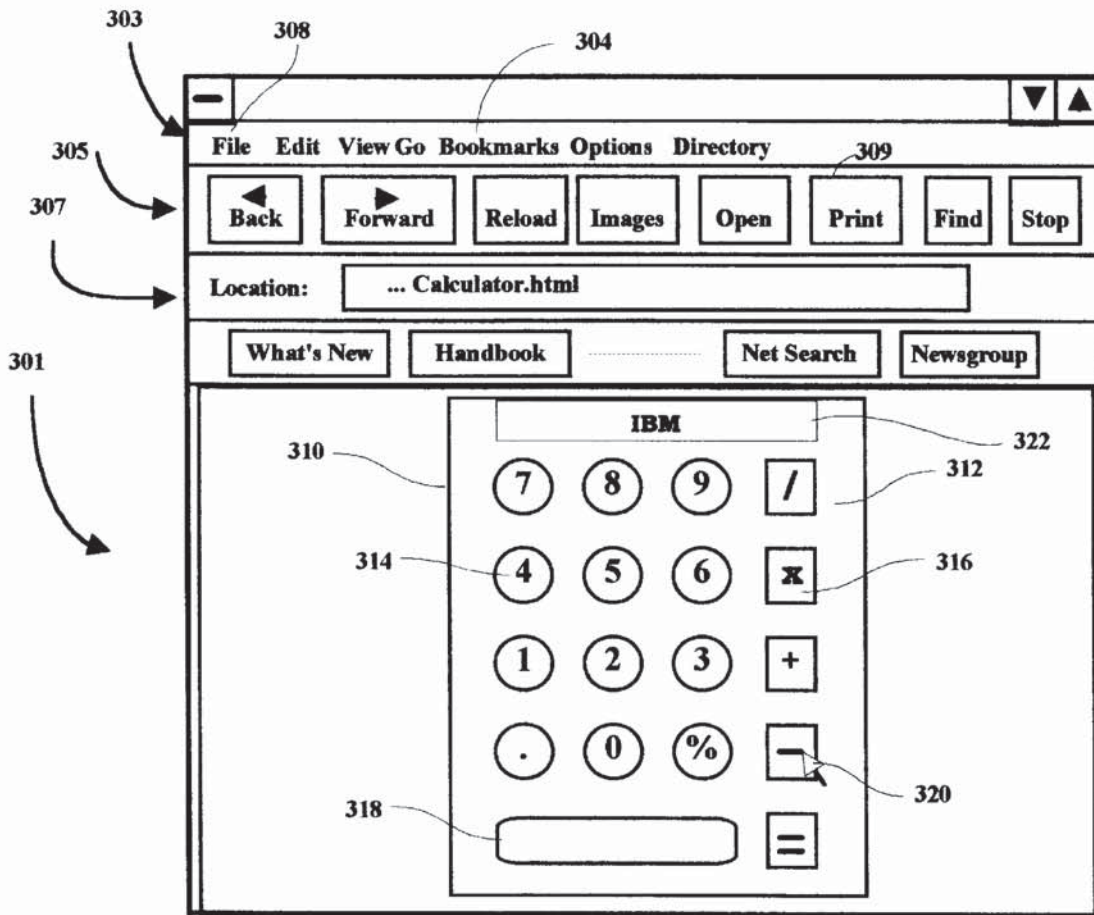
FIG. 1

FIG. 2

FIG. 3

GRAPHICAL USER INTERFACE STREAM PROCESSOR

FIELD OF THE INVENTION

The present invention relates generally to information processing systems and more particularly to a methodology and implementation for facilitating graphical user interface changes for user applications.

BACKGROUND OF THE INVENTION

In network computer systems, a plurality of computers are connected together and one or more of the computers generally perform the function of a network server. A network may be established through the hard-wired inter-connection of a plurality of computer terminals in a local network or on a wider scale such as the Internet or world wide web which may include telecommunication links. In any case, the network server acts as the central control unit for providing access to files, programs and program execu-tion to the individual computers connected within the net-work. In general, a networked computer terminal will .logon. to the network and obtain access to a network server. The network server will download a homepage presentation on the user's terminal which may include and audio and video effects. From the initial screen display, a user may designate any other .address. to get to another .location. or screen display. The user may also select from a plurality of functions which may be presented for user selection on the initial display.

Generally, almost all screen displays include designated selection areas on the screen where the user may point a selection device such as a mouse-controlled pointer, and .click. on the designated screen area to invoke a displayed function, or menu for other function selection options. When the homepage or selection screen is downloaded from the server, many small programs or .applets. are also down-loaded at the same time to the user's terminal memory. Those applets are associated with different ones of the designated screen selection areas and are executed when a user makes a selection or clicks on one of the selection areas on the display. The applets, in turn, may call for the presentation of a new user interface screen, or a new or different section of a portion of the screen, such as a pop-up menu or other graphical user interface (GUI) which may be within a framed or windowed sub-section of the main homepage.

The GUI presentation is typically designed and custom-ized to present an attractive display and facilitate use of the computer interface by a user in making subsequent selec-tions or executing selected functional aspects associated with the GUI presentation. The GUI display is typically displayed as the end result of the execution of a program downloaded from the server in response to the user selection of a designated screen area. The design and display of the GUI screen is typically determined in advance and .hard-coded. by a programmer. Thus, on the Internet for example, there is no flexibility for different JAVA applet user inter-faces to be readily created when the content of the web server changes dynamically. In general, this hard-coded GUI program, which resides at the site of the server, has been difficult and time consuming to modify or make even the smallest change. If the GUI presentation screen needed to be changed in any aspect, the GUI program would have to be re-written, de-bugged, hard-coded and re-installed on the server. Further, if other aspects of the GUI presentation needed to be changed at a later time, the same process would

apply. As a result, the GUI screens tend to remain the same without ever changing to any great extent. This is unattrac-tive to a user who may wish to look at different screens to minimize an otherwise tedious search on the Internet for example, and also cumbersome to a network provider who may wish to change GUI screen appearance's on a regular basis to enhance the appeal of the particular function with which the GUI is associated. Moreover, with an improved methodology for changing GUI presentations, the GUI displays to the user can be more easily and quickly modified with relative ease and cost to a network provider. Further, sponsorship for GUI displayed functions may be incorpo-rated and readily changed at short notice to accommodate various sponsoring entities or functional program providers or contributors.

Thus, there is a need for an improved GUI presentation methodology which is effective to facilitate and accommo-date frequent changes to the displayed interface whereby a GUI interface program may be modified quickly and easily by a network server in providing a greater selection of GUI presentations to network users.

SUMMARY OF THE INVENTION

A method and implementing computer system is provided in which a GUI stream processor interprets a data stream consisting of GUI controls and attributes, and places the controls and attributes in a presentation space. In an exem-plary embodiment, a web browser program loads an HTML (Hypertext Markup Language) from a GUI (Graphical User Interface) stream processor applet from a web server. The web browser then loads the GUI stream processor (GSP) from the web server and begins executing the GSP. The GSP then requests the GUI stream from the web server by opening a stream to a file, CGI (Common Gateway Interface) script or servlet on the server. The GSP then receives the GUI stream and assembles the user interface in the JAVA applet presentation space managed by the browser. The GSP may periodically update itself by repeating the last two steps.

BRIEF DESCRIPTION OF THE DRAWINGS

A better understanding of the present invention can be obtained when the following detailed description of a pre-ferred embodiment is considered in conjunction with the following drawings, in which:

FIG. **1** is a diagram of a computer system in which the present invention may be implemented;

FIG. **2** is a simplified schematic diagram showing selected components and subsystems of the computer system illus-trated in FIG. **1**; and

FIG. **3** is an illustration of an exemplary display screen of a typical web page for reference in explaining the present invention.

DETAILED DESCRIPTION

With reference to FIG. **1**, the various methods discussed herein may be implemented within a computer network including a computer terminal **101**, which may comprise either a workstation or a PC for example. In general, an implementing computer system may include computers con-figured with a plurality of processors in a multi-bus system in a network of similar systems. However, since the work-station or computer terminal **101** implementing the present invention in an exemplary embodiment, is generally known in the art and composed of electronic components and

circuits which are also generally known to those skilled in the art, circuit details beyond those shown in FIG. **1**, are not specified to any greater extent than that considered necessary as illustrated, for the understanding and appreciation of the underlying concepts of the present invention and in order not to obfuscate or distract from the teachings of the present invention.

In FIG. **1**, the computer system includes a processor unit **103** which is typically arranged for housing a processor circuit along with other component devices and subsystems of the computer terminal **101**. The computer terminal **101** also includes a monitor unit **105**, a keyboard **107** and a mouse or pointing device **109**, which are all interconnected with the computer terminal illustrated. Also shown is a connector **111** which is arranged for connecting a modem within the computer terminal to a communication line such as a telephone line in the present example. The present invention may also be implemented in a cellular system.

Several of the major components of the terminal **101** are illustrated in FIG. **2**. A processor circuit **201** is connected to a system bus **203** which may be any host system bus. It is noted that the processing methodology disclosed herein will apply to many different bus and/or network configurations. A cache memory device **205**, and a system memory unit are also connected to the bus **203**. A modem **209** is arranged for connection **210** to a communication line, such as a telephone line, through a connector **111** (FIG. **1**). The modem **209**, in the present example, selectively enables the computer terminal **101** to establish a communication link and initiate communication with network server, such as the Internet.

The system bus **203** is also connected through an input interface circuit **211** to a keyboard **213** and a mouse or pointing device **215**. The bus **203** may also be coupled through a hard-wired network interface subsystem **217**. A diskette drive unit **219** is also shown as being coupled to the bus **203**. A video subsystem **220**, which may include a graphics subsystem, is connected to a display device **221**. A storage device **218**, which may comprise a hard drive unit, is also coupled to the bus **203**. The diskette drive unit provides a means by which individual diskette programs may be loaded on to the hard drive, or accessed directly, for selective execution by the computer terminal **101**. As is well known, program diskettes containing application programs represented by magnetic indicia on the diskette, may be read from the diskette drive, and the computer system is selectively operable to read such magnetic indicia and create program signals. Such program signals are selectively effective to cause the computer system to present displays on the screen of a display device and respond to user inputs in accordance with the functional flow of the application program on the diskette.

In running an Internet access program or browser program on the computer terminal **101**, the access program is typically stored in the storage device **218** and either selectively or automatically, partially or totally, loaded into the system memory **207** when the system is initially powered-on, or at a later time if so desired by a user. The browser is selectively operable to access and execute a site selection program, as herein described, either directly from a diskette in the diskette drive unit **219** or directly from a copy of the site selection program stored on the hard drive unit **218**. As a program is running, either a portion of the program or the entire program may be loaded into the system memory **207** and/or the system cache memory **205**. Depending on specific program design, the system may store any information accessed from a database in the storage unit **218**, the cache memory **205**, the system memory **207** or directly from a

diskette loaded into the diskette drive **219**. Assuming a user has started-up the system, and is actively running a browser program for example, from memory, a series of screens will be displayed to the user on the display device **221**. Each screen typically has one or more selections for the user to make in navigating through the program. In general, a user will make selections from a home page display screen using the keyboard **213** or the mouse or pointer device **215**. In an Internet operating program, the selections made by the user will determine .where. the user .goes., i.e. to what .site. or .webpage., and also, in some cases, the communications link or the path taken to get to the site selected.

FIG. **3** illustrates a typical program screen display **301** in an Internet operating session. The browser screen generally includes a mode row **303** which displays several different modes which the user may select such as the .File. selection **308** and the .Bookmarks. selection **304**. Another row **305** may be displayed to help a user quickly move through documents, sites, or pages in a network application. An address or .location. section **307** enables a user to key-in, and also displays the name of, an internet address of a site to be, or being, visited. In the present example, the location designation is a .calculator. function which may be keyed into the location field **307** or clicked on by means of an icon from a previous page. Other quick access buttons may be implemented on the screen for quick access to other selected network services and/or network functions such as .What's New. or .Net Search. In general, any of the illustrated items may be selected through a .point and click. methodology associated with the mouse device **215**, and a cursor or pointer **320** visible on the display screen. For example, a download of data from a remote site may be immediately terminated during the transmission by pointing to the .Stop. button and clicking on a designated mouse button. Similarly, the .Back. and .Forward. buttons may be used to return to the last screen display or go forward to the next screen display, respectively.

In the FIG. **3** example, the .location. or Uniform Resource Locator (URL) designation is illustrated as . . . Calculator.html. The exemplary webpage shows a calculator function and image which has been downloaded from a server in response to a user selection of the appropriate URL or .location. The illustrated calculator function includes a calculator outer frame or edge **310**. Also illustrated is a faceplate **312**. Within the face of the illustrated calculator frame **310** are typical buttons including numerical buttons **314** and function buttons **316** as well as a .results. area **318** for displaying the results of entered calculations. Also illustrated is a display area **322** which may be used to display the identity of a sponsoring entity. Typically, a user will use a selector arrow **320** to select numbers and functions to perform a desired calculation. It is noted that even the simple calculator example includes many attributes and variables which may be changed in the display of the calculator image. For example, the numerical .buttons. **314** can assume a variety of aspects such as different colors, different patterns or apparent textures, different shapes (round, square, oblong etc.) and even different numerical designations. Similarly, the function buttons **316** and the backplate **312** may take on many different appearances. Also text can assume different values for font type and/or font size. Furthermore the combination of the different attributes, including different colors, shapes, sizes and fonts may be varied to present a large number of possible combination presentations of the displayed GUI screen.

The calculator display shown in FIG. **3** is typically generated by a program or application or applet that is

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase
Smarter legal research.