



(22) Date de dépôt/Filing Date: 2009/08/24

(41) Mise à la disp. pub./Open to Public Insp.: 2009/11/05

(51) Cl.Int./Int.Cl. *G06F 9/44* (2006.01),
G06F 3/14 (2006.01)

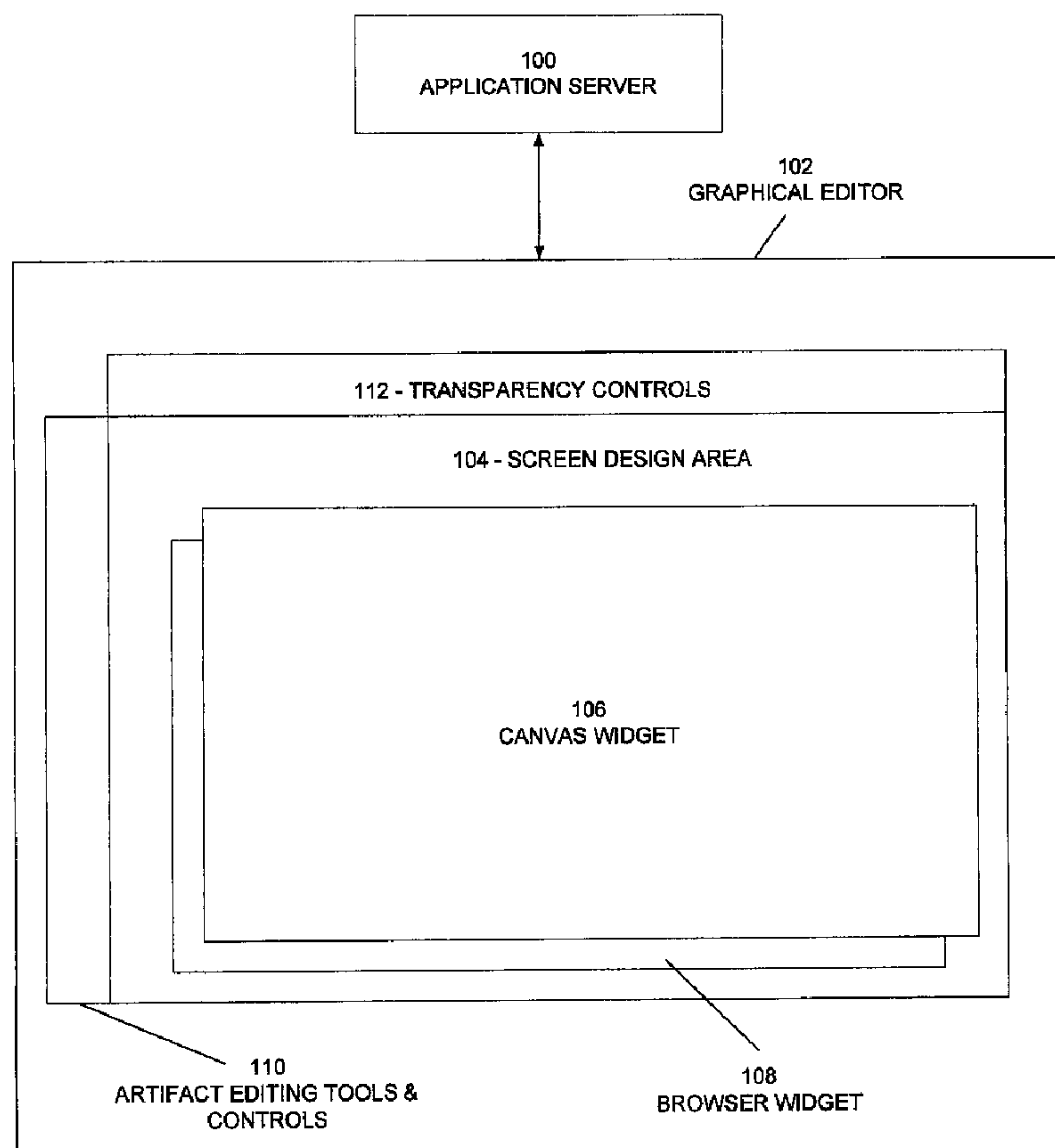
(71) Demandeur/Applicant:
IBM CANADA LIMITED - IBM CANADA LIMITEE, CA

(72) Inventeur/Inventor:
FARN, BRIAN GIN, CA

(74) Agent: WANG, PETER

(54) Titre : EDITEUR GRAPHIQUE

(54) Title: GRAPHIC EDITOR



(57) Abrégé/Abstract:

A graphical editor comprising a browser implemented as a screen widget. A canvas widget is overlaid onto the browser widget and has a level of transparency. Screen artifacts are displayed by the browser widget and the canvas widget displays an indicator of the position and size of the artifact precisely as rendered by the browser widget.

GRAPHIC EDITOR

ABSTRACT OF THE DISCLOSURE

A graphical editor comprising a browser implemented as a screen widget. A canvas widget is overlaid onto the browser widget and has a level of transparency. Screen artifacts are displayed by the browser widget and the canvas widget displays an indicator of the position and size of the artifact precisely as rendered by the browser widget.

CA920090025CA1

GRAPHIC EDITOR

BACKGROUND OF THE INVENTION

[0001] This invention is related generally to the fields of networking, computing, and graphical editors, and specifically to the use of tools to assist in the preparation of HTML source code for the distribution of worldwide web pages. A graphical page editor is disclosed for the purpose of generating true web page images and the corresponding HTML source code and displaying the page to a designer precisely as a selected browser will render the corresponding web pages. The disclosed, illustrative graphical editor is designed to execute on a computer such as a desktop, a workstation, a laptop or general-purpose mainframe, although alternative embodiments such as special-purpose electronics are possible.

BRIEF SUMMARY OF THE INVENTION

[0002] According to one embodiment of the invention, there is disclosed a graphical editor comprising a browser implemented as a screen widget. A canvas widget is overlaid over the browser widget and has a level of transparency. Screen artifacts are displayed by the browser widget and the canvas widget displays an indicator of the position and size of each artifact as rendered by the browser widget.

[0003] According to a second embodiment, there is disclosed a method of generating graphical pages for display by a selected browser, comprising creating a browser widget from the browser, overlaying a canvas widget with a level of transparency onto the browser widget, displaying a browser artifact with the browser widget and displaying with the canvas widget an indicator of the position and size of the artifact as rendered by the browser widget.

[0004] According to a third embodiment, there is disclosed a computer program product for designing graphic pages to be displayed by a selected browser. The computer program product comprises a computer usable medium having computer usable program code embodied

CA920090025CA1

therewith. The computer usable program code comprises computer usable program code configured to create a browser widget, computer usable program code configured to overlay the browser widget with a canvas widget having a level of transparency; computer usable program code configured to display an artifact on the browser widget, and computer usable program code configured to display on the canvas widget an indicator of size and position of the artifact as rendered by the browser widget.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

[0005] In the drawings,

[0006] Fig. 1 shows a conceptual view of the front of a monitor having graphical editing tools and controls and in which a top-level canvas widget overlays a bottom-level browser widget. The front of the monitor also contains transparency controls for the canvas widget;

[0007] Fig. 2 shows a functional-level flowchart of the operations that occur within a graphical editor to populate artifacts and size and location indicators onto the browser widget and the canvas widget, respectively;

[0008] Fig. 3 shows a more detailed flowchart of the populating process of Fig. 2;

[0009] Fig. 4 shows a sequence of message flows between an application server and a graphical editor, including the browser and canvas widgets; and

[0010] Fig. 5 is a simplified view of what a designer might see on a monitor screen during a design process.

DETAILED DESCRIPTION OF THE INVENTION

[0011] As will be appreciated by one skilled in the art, the present invention may be embodied as a method, system, or computer program product. Accordingly, the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a “circuit,” “module” or “system.” Furthermore, the present invention may take the form of a computer program product on a computer-usable storage medium having computer-usable program code embodied in the medium.

[0012] Any suitable computer usable or computer readable medium may be utilized. The computer-usable or computer-readable medium may be, for example but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, device, or propagation medium. More specific examples (a non-exhaustive list) of the computer-readable medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a transmission media such as those supporting the Internet or an intranet, or a magnetic storage device. Note that the computer-usable or computer-readable medium could even be paper or another suitable medium upon which the program is printed, as the program can be electronically captured, via, for instance, optical scanning of the paper or other medium, then compiled, interpreted, or otherwise processed in a suitable manner, if necessary, and then stored in a computer memory. In the context of this document, a computer-usable or computer-readable medium may be any medium that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device. The computer-usable medium may include a propagated data signal with the computer-usable program code embodied therewith, either in baseband or as part of a carrier wave. The computer

CA920090025CA1

usable program code may be transmitted using any appropriate medium, including but not limited to the Internet, wireline, optical fiber cable, RF, etc.

[0013] Computer program code for carrying out operations of the present invention may be written in an object oriented programming language such as Java, Smalltalk, C++ or the like. However, the computer program code for carrying out operations of the present invention may also be written in conventional procedural programming languages, such as the "C" programming language or similar programming languages. The program code may execute entirely on the designer's computer, partly on the designer's computer, as a stand-alone software package, partly on the designer's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the designer's computer through a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider).

[0014] The present invention is described below with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0015] These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory

produce an article of manufacture including instruction means which implement the function/act specified in the flowchart and/or block diagram block or blocks.

[0016] The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide steps for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0017] With reference now to Fig. 1, there is shown the front of an illustrative monitor containing artifact editing tools and controls 110 and screen components of an illustrative graphical editor 102. The disclosed editor comprises a screen design area 104 that contains two widgets as seen on the screen. A top-level canvas widget 106 overlays a bottom-level browser widget 108. The canvas widget 106 is typically transparent, revealing to the designer the contents displayed by the browser widget 108 below. The browser widget 108 can represent any browser such as Microsoft Internet... etc. The selected browser is implemented as a widget. A designer using the editor interacts with the canvas widget, while typically viewing the browser widget through the canvas widget. The canvas widget serves as isolation from the browser so that a designer cannot inadvertently operate a hyperlink, for example, and lose control of the design process. The canvas widget also contains indicators that reveal to a designer the position and size of artifacts contained in the browser widget. In the disclosed embodiment, such indicators are chosen to be lines or brackets that define portions of the perimeters of artifacts on the browser widget. So a designer will typically see an artifact contained on the browser widget, outlined in some manner by indicators contained on the canvas browser. For reasons that will become clear, transparency controls 112 are provided to the designer so that the transparency of the canvas widget can be adjusted by the designer. An array of artifact editing tools and controls are provided on the left-side of the screen design area. The use and type of such tools and controls are well-known in web and graphical design products and do require further discussion. As shown in Fig. 1, such an editor might be connected to an application server 100,

CA920090025CA1

such as Websphere Application Server marketed by IBM Corporation, which might be used for storage purposes or perhaps as a typical web server such as in this disclosed embodiment.

[0018] Fig. 2 shows illustrative process steps executed by the disclosed embodiment of the invention. The editor is opened by a designer as illustrated at 200. At 202 the editor requests a session identifier (session ID) from the application server 100 or in alternative embodiments the application server might be replaced by the computer on which the editor is executing. At 204, under control of the designer, the editor either opens an existing page X for editing, or it logically creates a new page X. The browser widget 108 and canvas widget 106 are created at step 206. If the page X does not already exist, as determined at step 206, the editor 102 moves to step 212. If the page X already exists, the editor 102 populates the browser widget at step 210 with the existing artifacts that are defined for the page before moving to step 212.

[0019] Step 212 establishes a loop to wait for and to respond to operations of the designer in creating or editing this page X. When a modification is made, such as the designer dragging an icon to the screen, or entering text in a text field, step 214 analyzes the designer action and updates the HTML source code for the page accordingly. The editor then populates the browser widget and the canvas widget at step 216 as needed. This loop continues with steps 212, 214 and 216 until the designer closes the page at 218.

[0020] Steps 210 and 216 for populating a page with artifacts are implemented as a subroutine in the disclosed embodiment. An illustrative subroutine is shown in Fig. 3. Because the canvas widget isolates the designer from the browser widget, some communication messages between the editor and the canvas and browser widgets are made through the application server 100 in the disclosed embodiment... Fig. 4 illustrates the message flows created by the steps of Fig. 3 and will assist the reader if both Figs. are viewed together. To populate the browser widget 108 for a given artifact, the editor at step 302 sends a Universal Resource Locator (URL) and the session ID to the browser widget. This corresponds to message 1 in Fig. 4. In response, the browser widget requests and receives the HTML code for the artifact from application server 100 in a standard manner. This corresponds to messages 2 and 3 of Fig. 4. When the browser

CA920090025CA1

completes the rendering of the artifact in the browser widget at 306, it notifies application server 100 that rendering is complete. This corresponds to message 4 of Fig. 4. Application server 100 forwards this message to the editor as shown by message 5 in Fig. 4. In response to message 5, the editor 102 at step 308 sends message 6 (Fig. 4) to application server 100 requesting the screen location coordinates and size of the rendered artifact. This information is known only by the browser widget, so application server 100 forwards message 6 to the browser widget as message 7 (Fig.4). When it receives message 7 at step 310, the browser widget 108 gathers the location coordinates, size parameters of the artifact and, if relevant, the parent-child relationships of the artifact and returns this information to the editor. These parameters are derived from the Document Object Model (DOM) in the disclosed embodiments. This return corresponds to message 8 to the application server 100, which forwards the message to the editor as message 9. In response to message 9, editor 102 uses the position, size and parent-child relationship information to overlay the bracket indicators for the artifact onto the canvas widget (step 312). One illustrative example of bracket indicators is shown in Fig. 5. In this way, the designer sees precisely how the web page will look when rendered with a browser corresponding to the browser widget. When finished with rendering of the artifact bracket indicators, the process of Fig. 3 returns to step 210 or step 216 of Fig. 2.

[0021] It is possible that the visual attributes of the artifact indicators placed onto the canvas widget are similar or identical to visual attributes present on the browser widget. If so, this might make it difficult for a designer to actually see the artifact indicators. The transparency controls 112 shown in Fig. 1 are useful in such a case. By increasing the opaqueness of the canvas widget, the artifact indicators can be made more visible. There are other ways of accomplishing a similar result. For example, an embodiment might adjust the color scheme used for indicators, rather than opaqueness of the canvas widget. Even brightness and contrast controls can be effective in certain cases to increase the visibility of the canvas widget indicators as against the artifacts on the browser widget.

[0022] Fig. 5 is a simplified view of what a designer might see on a monitor. An illustrative artifact 502, a cloud in this example, has been dragged and dropped onto the screen and rendered

CA920090025CA1

by browser widget 108. The illustrative position and size bracket indicators 504 are added to the canvas widget 106 by the editor 102 according to the location and size parameters returned to the editor 102 by the browser widget. It is understood that there are many variations on how the indicators might be implemented. For example, an indicator might be a rectangle rather than brackets, or an indicator could outline the perimeter of an artifact if enough location and size information are returned by a browser widget.

[0023] The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that, in some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

[0024] The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting of the invention. As used herein, the singular forms “a”, “an” and “the” are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms “comprises” and/or “comprising,” when used in this specification, specify the presence of stated features, integers, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components, and/or groups thereof.

[0025] The corresponding structures, materials, acts, and equivalents of all means or step plus function elements in the claims below are intended to include any structure, material, or act for performing the function in combination with other claimed elements as specifically claimed. The description of the present invention has been presented for purposes of illustration and description, but is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope and spirit of the invention. The embodiment was chosen and described in order to best explain the principles of the invention and the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.

[0026] Having thus described the invention of the present application in detail and by reference to preferred embodiments thereof, it will be apparent that modifications and variations are possible without departing from the scope of the invention defined in the appended claims.

CLAIMS

What is claimed is:

1. A graphical editor comprising
a browser implemented as a screen widget,
a canvas widget overlaying the browser widget with a level of transparency, wherein the browser widget displays a screen artifact, and the canvas widget displays an indicator of the position and size of the artifact as rendered by the browser widget.
2. The graphical editor of claim 1 wherein the editor further comprises a control for rendering the indicator visible against a background of the browser widget.
3. The graphical editor of claim 2 wherein the control further comprises a transparency control for varying the level of transparency of the canvas widget.
4. The graphical editor of claim 2 wherein the control is a brightness and contrast control.
5. The graphical editor of claim 2 wherein the control further comprises a color control for varying the color of the indicator on the canvas screen.
6. The graphical editor of claim 1 wherein the size and position of the artifact on the browser widget is obtained from the browser widget.
7. The graphical editor of claim 6 where the browser widget obtains the size and position of the artifact by examining a document object model (DOM) of the artifact.
8. A method of generating graphical pages for display by a selected browser, comprising
creating a browser widget from the browser,
overlaying a canvas widget with a level of transparency onto the browser widget,
displaying a browser artifact with the browser widget and

CA920090025CA1

displaying with the canvas widget an indicator of the position and size of the artifact as rendered by the browser widget.

9. The method of claim 7 further comprising rendering the indicator visible against a background of the browser widget using a control.

10. The method of claim 9 further comprising rendering the indicator visible against a background of the browser widget using a transparency control for varying the level of transparency of the canvas widget.

11. The method of claim 9 further comprising rendering the indicator visible against a background of the browser widget using a brightness and contrast control.

12. The method of claim 9 further comprising rendering the indicator visible against a background of the browser using a control for varying the color of the indicator on the canvas screen.

13. The method of claim 8 further comprising obtaining the size and position of the artifact on the browser widget from the browser widget.

14. The method of claim 13 further comprising obtaining the size and position of the artifact by examining a document object model (DOM) of the artifact.

15. A computer program product for designing graphic pages to be displayed by a browser, the computer program product comprising:
a computer usable medium having computer usable program code embodied therewith, the computer usable program code comprising:
computer usable program code configured to create a browser widget;
computer usable program code configured to overlay the browser widget with a canvas widget having a level of transparency;

computer usable program code configured to display an artifact on the browser widget, and computer usable program code configured to display on the canvas widget an indicator of size and position of the artifact as rendered by the browser widget.

16. The computer program product of claim 15 further comprising computer usable program code configured to render the indicator visible against a background of the browser widget.

17. The computer program product of claim 16 wherein the computer usable program code configured to render the indicator visible against a background of the browser widget further comprises code for varying the level of transparency of the canvas widget.

18. The computer program product of claim 16 wherein the computer usable program code configured to render the indicator visible against a background of the browser widget further comprises code for varying brightness and contrast.

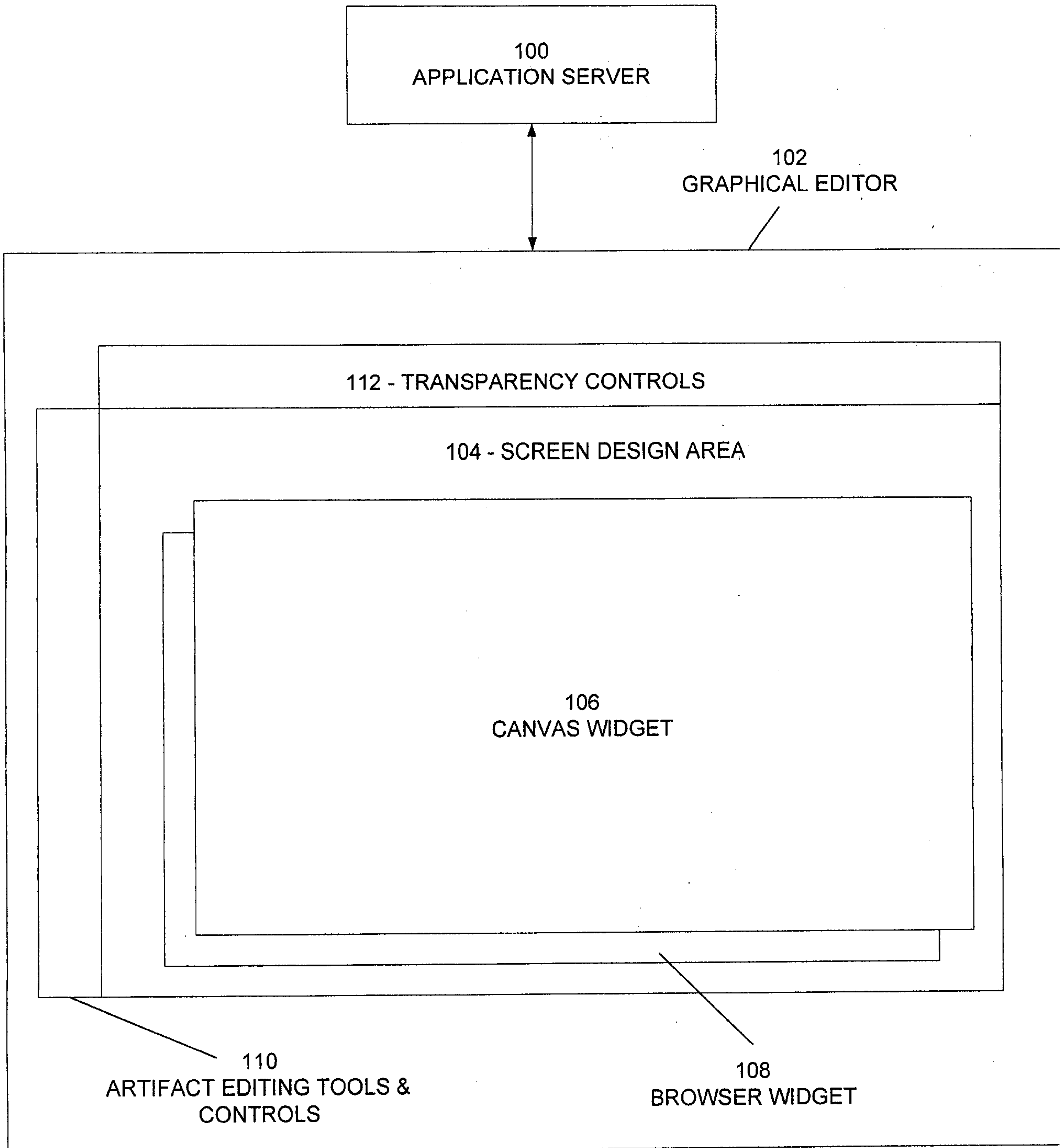
19. The computer program product of claim 16 wherein the computer usable program code configured to render the indicator visible against a background of the browser widget further comprises code for varying the color of the indicator on the canvas screen.

20. The computer program product of claim 15 further comprising code for obtaining the size and position of the artifact on the browser widget from the browser widget.

21. The computer program product of claim 20 where the code for obtaining the size and position of the artifact on the browser widget further comprises code for obtaining the size and position of the artifact from a document object model (DOM) of the artifact.

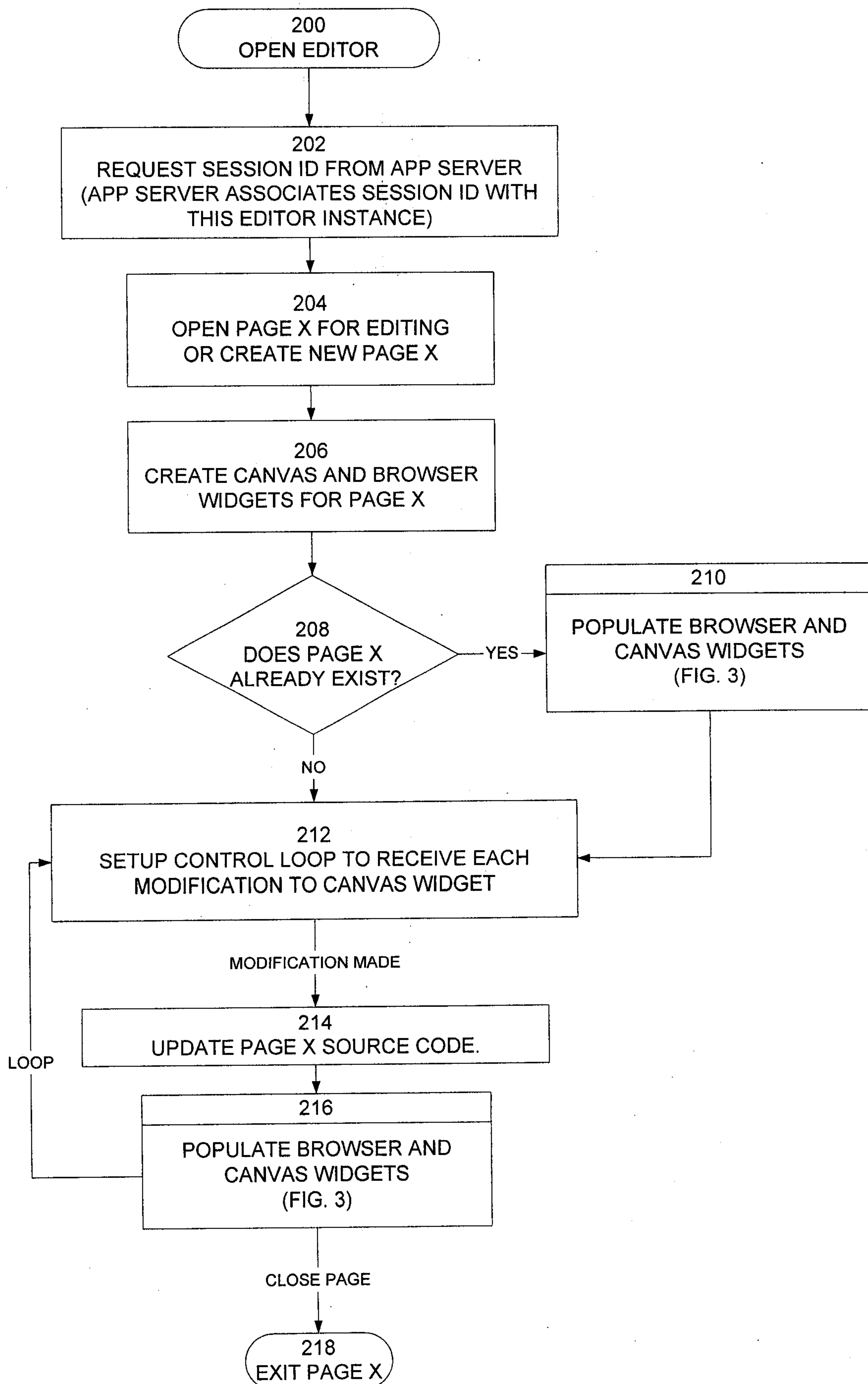
CA9-2009-0025CA1

Fig. 1



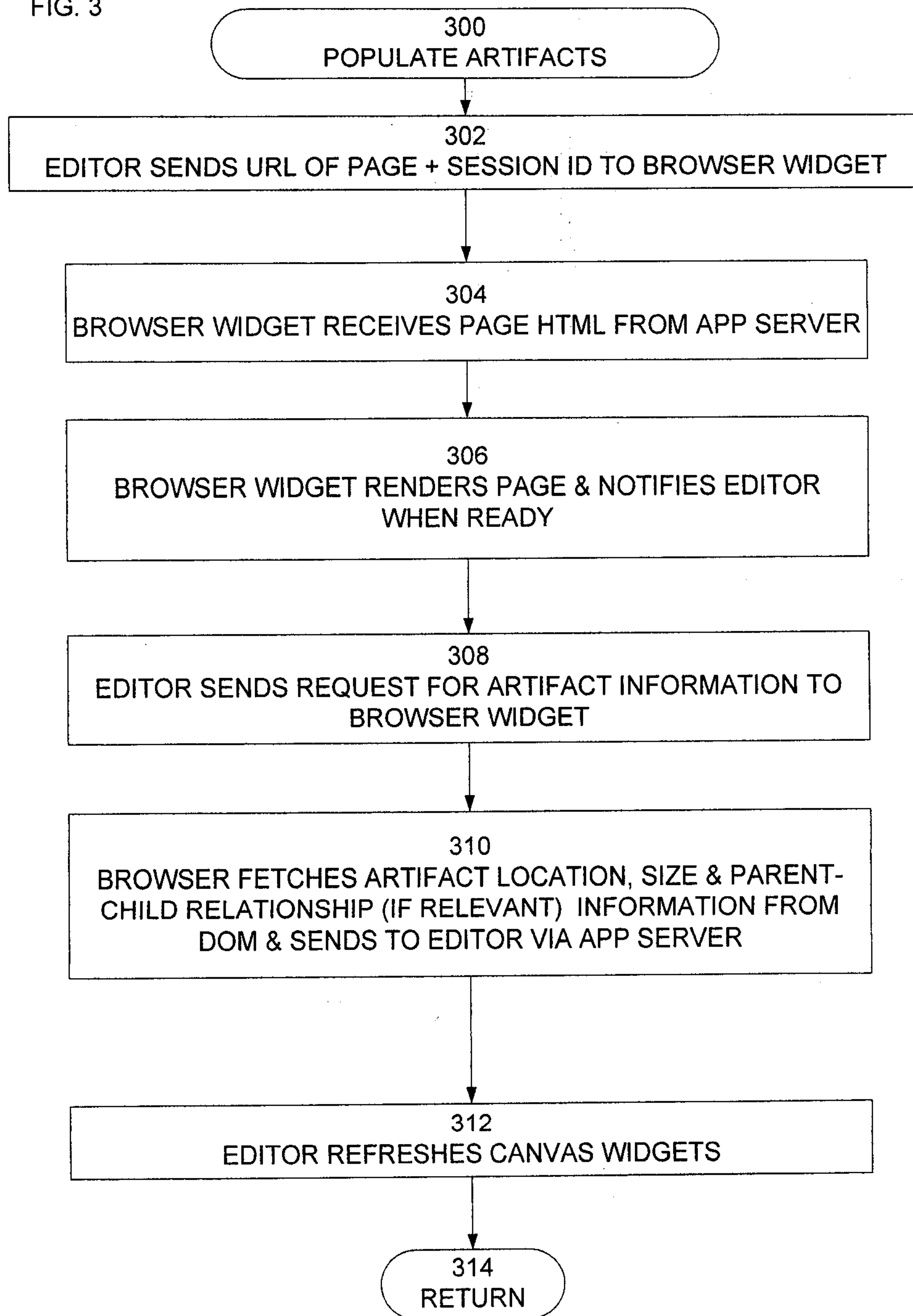
CA9-2009-0025CA1

FIG. 2



CA9-2009-0025CA1

FIG. 3



CA9-2009-0025CA1

FIG. 4

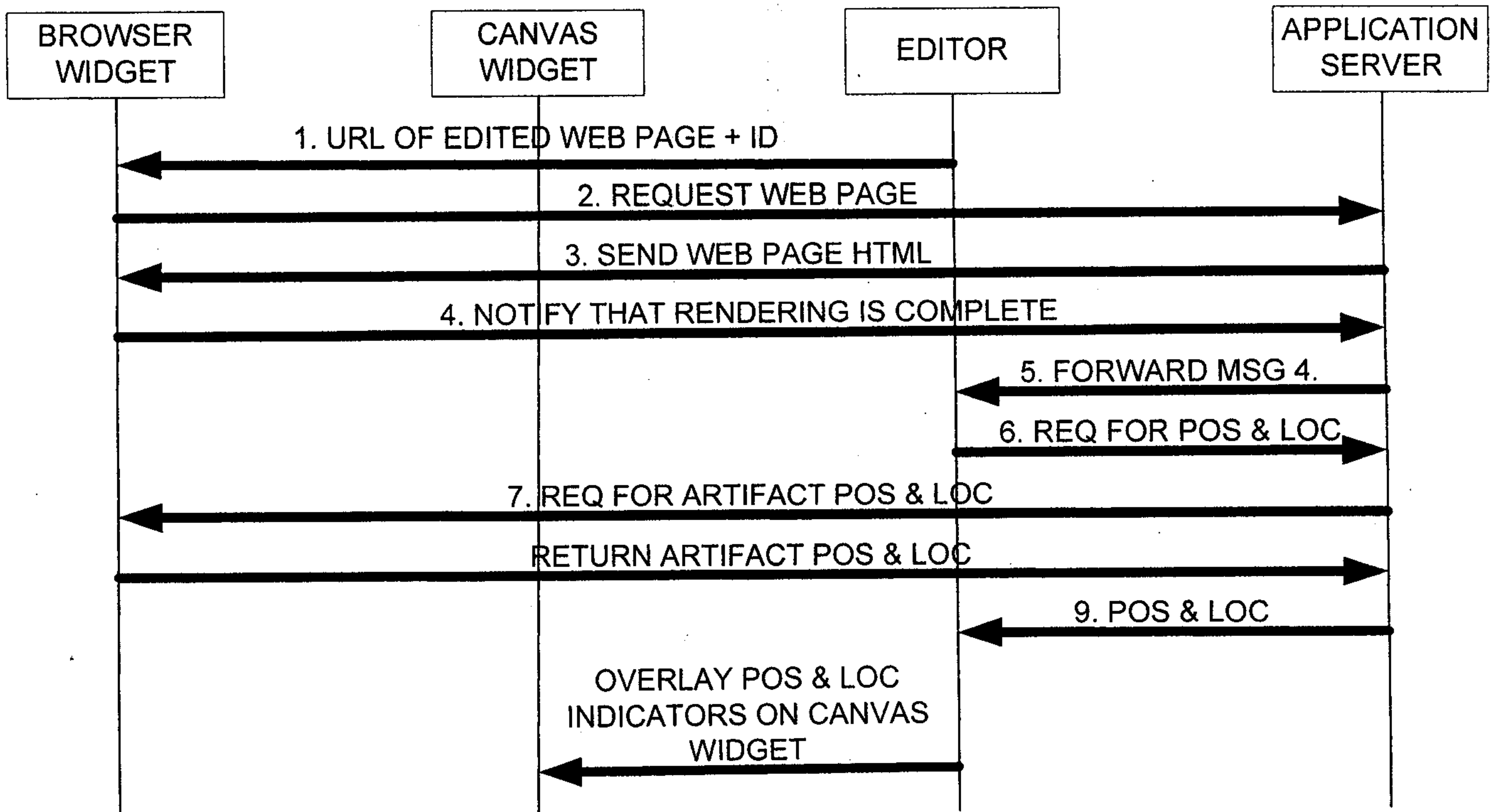
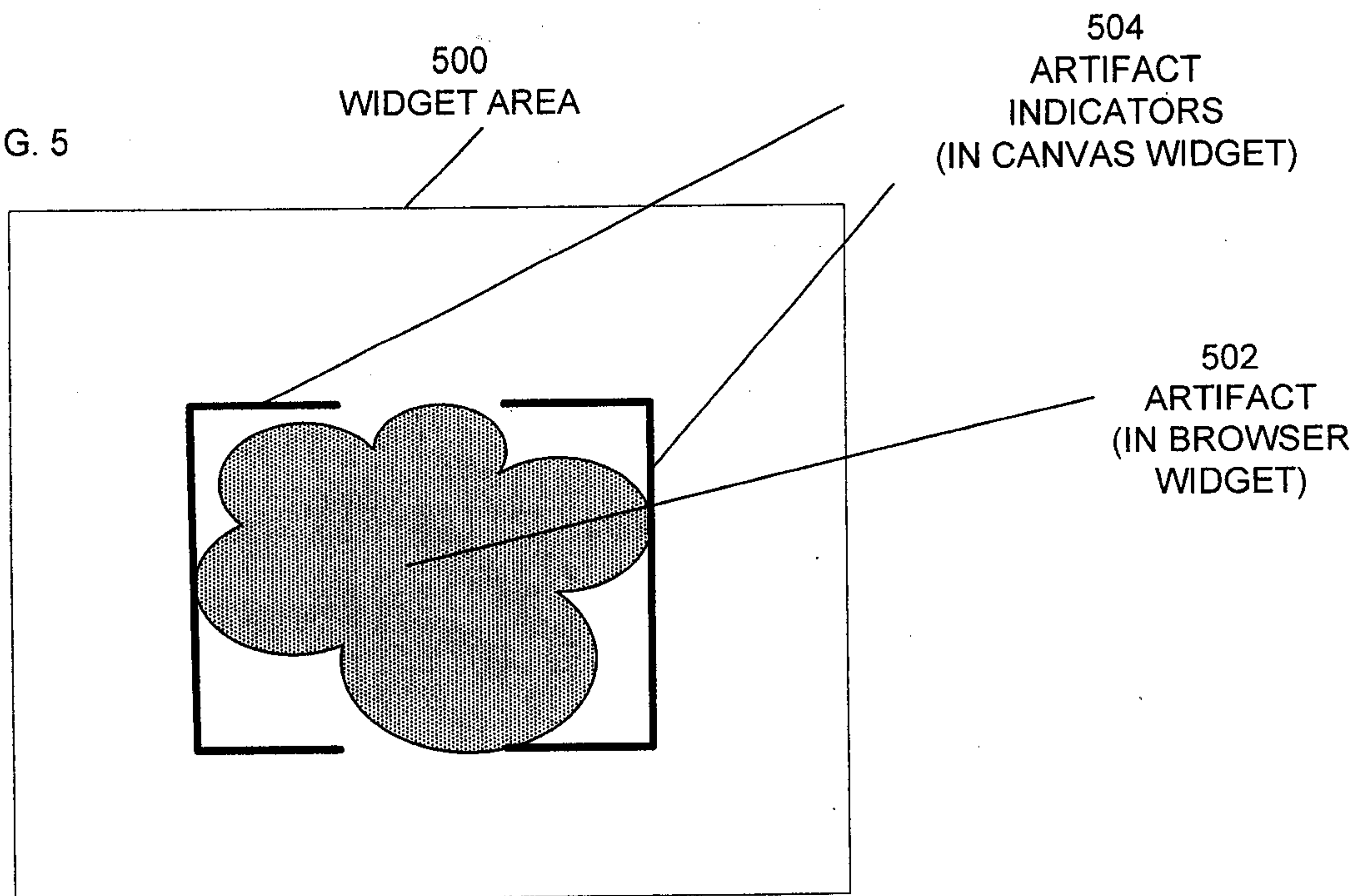
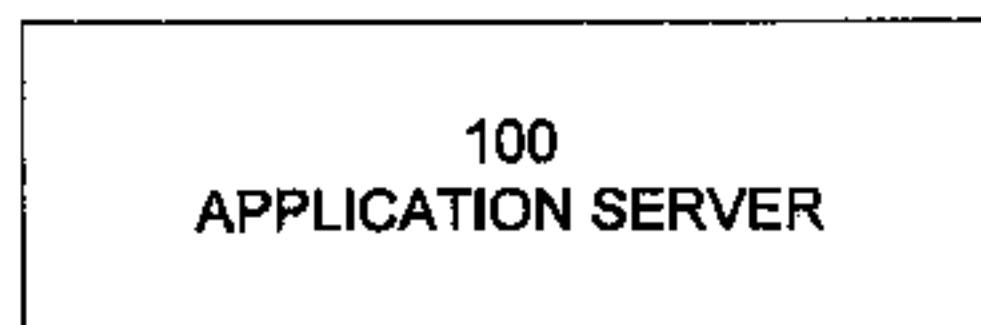


FIG. 5





100
APPLICATION SERVER

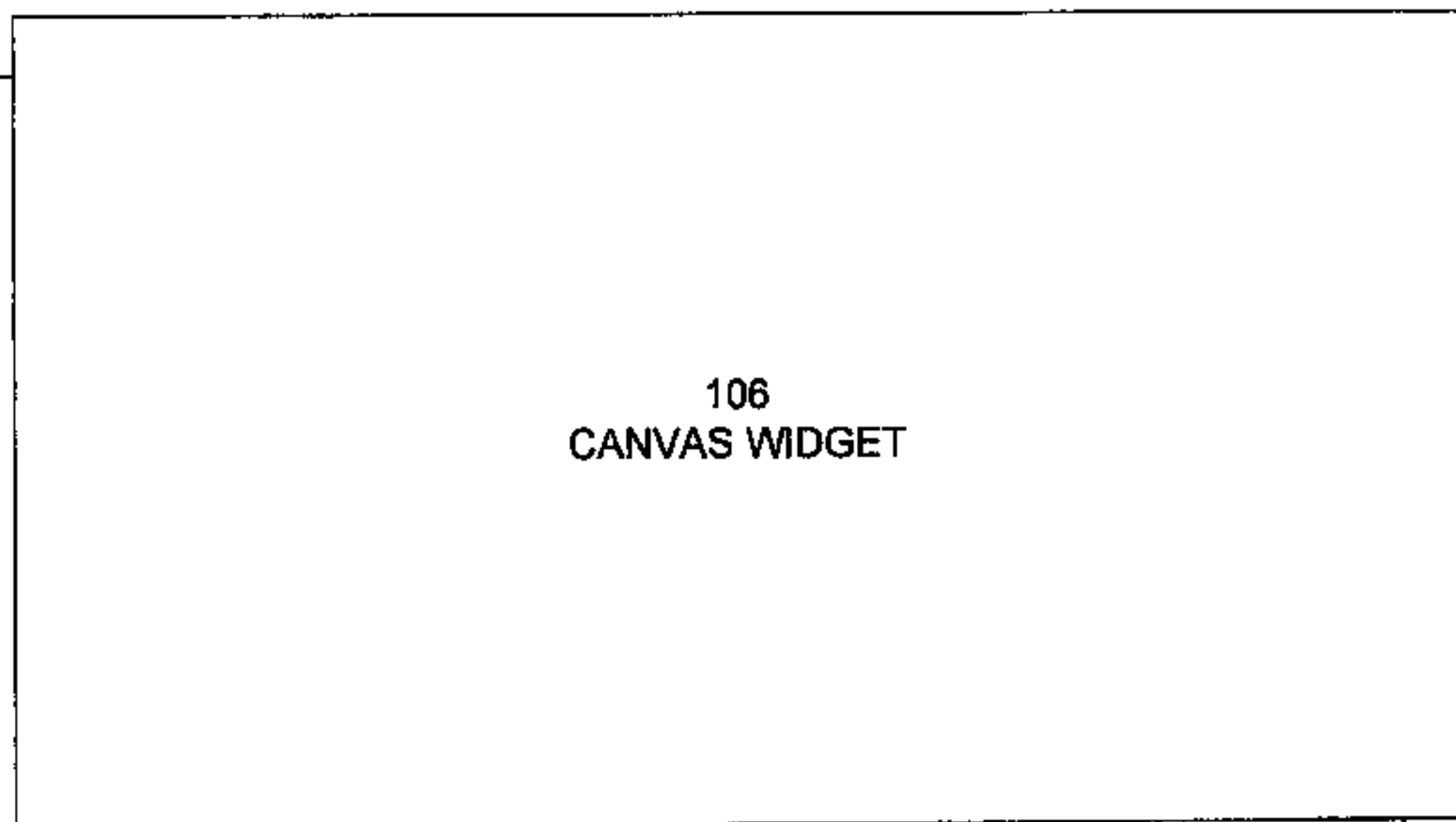


102
GRAPHICAL EDITOR



112 - TRANSPARENCY CONTROLS

104 - SCREEN DESIGN AREA



106
CANVAS WIDGET

110
ARTIFACT EDITING TOOLS &
CONTROLS

108
BROWSER WIDGET