

(51) International Patent Classification:
A63F 9/24 (2006.01)(21) International Application Number:
PCT/US2011/046860(22) International Filing Date:
5 August 2011 (05.08.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/371,276 6 August 2010 (06.08.2010) US(71) Applicant (for all designated States except US): **WMS GAMING, INC.** [US/US]; 800 South Northpoint Blvd., Waukegan, Illinois 60085 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **ANTKOWIAK, Ryan D.** [US/US]; 5525 N Winthrop Ave, #415, Chicago, Illinois 60640 (US). **GAZDIC, Daniel J.** [US/US]; 3232 N Halsted, D605, Chicago, Illinois 60657 (US). **GURA, Damon E.** [US/US]; 928 West Buena Avenue, Number 3, Chicago, Illinois 60613 (US). **HIMAWAN, Budyanto** [US/US]; 1564 S. Kembley Ave., Palatine, Illi-nois 60067 (US). **HO, Audrey F.Y.** [SG/US]; 651 West Buckingham Pl., Unit 2E, Chicago, Illinois 60657 (US). **LEVERENZ, Jeffrey M.** [US/US]; 1737 W. Summerdale, #2E, Chicago, Illinois 60640 (US).(74) Agents: **DELIZIO, Andrew** et al.; 15201 Mason Road, Suite 1000-312, Cypress, Texas 77433 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK,

[Continued on next page]

(54) Title: BROWSER BASED HETEROGENOUS TECHNOLOGY ECOSYSTEM

(57) Abstract: Wagering game systems can leverage web browsers to present wagering game content with a variety of technologies. For instance, an electronic wagering game table can present diverse wagering game content (e.g., multiple wagering games, portal games, social networking applets, hospitality menus, etc.) with a variety of technologies that interface with a web browser. Also, a single wagering game can be constructed from multiple technologies. A wagering game system can implement a framework that dynamically builds up a web page to use multiple technologies for presenting content with a web browser on a wagering game system. The framework can dynamically construct the web page, which can involve modifying objects, embedding objects, removing objects, and replacing objects that correspond to different technologies.

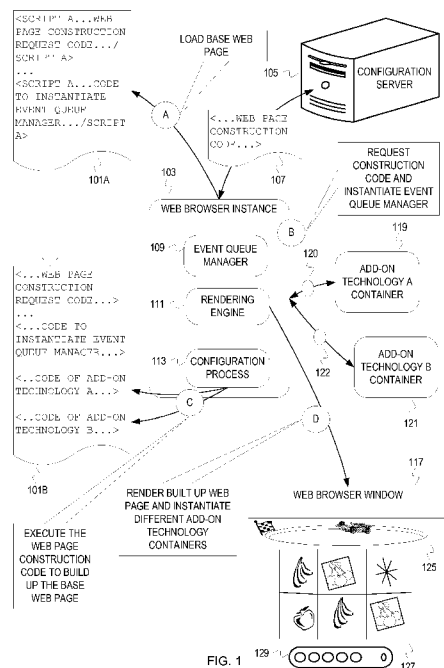


FIG. 1



EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU,
LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK,
SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ,
GW, ML, MR, NE, SN, TD, TG).

— *before the expiration of the time limit for amending the
claims and to be republished in the event of receipt of
amendments (Rule 48.2(h))*

Published:

— *with international search report (Art. 21(3))*

BROWSER BASED HETEROGENEOUS TECHNOLOGY ECOSYSTEM

RELATED APPLICATIONS

[0001] This application claims the priority benefit of U.S. Provisional Application Serial No. 61/371,276 filed Aug 6, 2010.

LIMITED COPYRIGHT WAIVER

[0002] A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever. Copyright 2011, WMS Gaming, Inc.

FIELD

[0003] Embodiments of the inventive subject matter relate generally to wagering game systems, and more particularly to a browser based heterogeneous technology system.

BACKGROUND

[0004] A variety of technologies are available for creating interactive web pages. Web pages often include scripts that imbue a web page with functionality beyond that available with hypertext markup language alone. In addition, features/functionalities are added to a web browser with web browser add-ons. Add-ons include themes, extensions, and plug-ins. These allow the appearance of a web browser window to be customized. Extensions typically integrate with logic of a web browser to add functionality to the web browser. Plug-ins also add functionality to a web browser, but remains separate from the web browser. A plug-in and a web browser communicate via an established interface, through which values and/or code are communicated. Further, web browsers aggregate additional technologies (e.g., extensible markup language, cascading style sheets, document object models) with scripts and add-ons for a highly interactive web page.

SUMMARY

[0005] In some embodiments, a method comprises a web browser loading a web page source that comprises configuration code. The web browser executes the configuration code on a machine that hosts the web browser. In response to executing the configuration code, an event queue manager that manages an event queue is instantiated responsive to said executing the

configuration code. Web page construction code that indicates web page elements is requested responsive to said executing the configuration code. The web page elements comprise a first web page element that corresponds to a first technology and a second web page element that corresponds to a second technology. The web page source is built up to include the web page elements in accordance with the web page construction code resulting in a built up web page source. The built up web page source is rendered. Rendering the built up web page source comprises instantiating a first add-on technology container to render the first web page element, and instantiating a second add-on technology container to render the second web page element.

[0006] In some embodiments, a method comprises establishing associations between an instance of the web browser and a plurality of add-on technology containers. The plurality of add-on technology containers correspond to a plurality of technologies and perform at least one of presenting content in accordance with a corresponding one of the plurality of technologies and execute code in accordance with a corresponding one of the plurality of technologies. A first visual component of the wagering game content is presented with a first of the plurality of add-on technology containers. A second visual component of the wagering game content is presented with a second of the plurality of add-on technology containers. Game play data of the wagering game content is processed with a third of the plurality of add-on technology containers.

[0007] In some embodiments, one or more machine-readable storage media have instructions stored therein. The instructions, when executed, cause a machine to modify a web page source to include a plurality of web page objects. The web page objects indicate different technologies for presenting content via a web browser. The instructions cause the machine to instantiate a plurality of web browser add-on containers for the web page objects in accordance with the different technologies. The instructions cause the machine to manage wagering game events received by an instance of the web browser. The instructions cause the machine to manage subscriptions to wagering game events by the plurality of web browser add-on containers. The instructions cause the machine to communicate indications of wagering game events to the plurality of web browser add-on containers in accordance with the subscriptions.

[0008] In some embodiments, one or more machine-readable storage media have instructions stored therein. The instructions, when executed, cause a machine to register a set of one or more add-on technology containers as subscribers to an event. The instructions cause the machine to detect the event received by a web browser instance at a wagering game machine. The web browser instance renders a web page from a web page source that comprises web page elements

that indicate the set of one or more add-on technology containers; determine the set of one or more add-on technology containers as subscribed to the event. The instructions cause the machine to communicate an indication of the event to the set of one or more add-on technology containers via one or more corresponding interfaces between the set of one or more add-on technology containers and the web browser instance.

[0009] In some embodiments, an apparatus comprises a physical network interface and means for loading a web page source and modifying the web page source to include a plurality of web page elements that correspond to different technologies. The apparatus also comprises means for instantiating a plurality of web browser add-on containers that implement the different technologies for rendering the web page elements. The plurality of web browser add-on containers perform at least one of present content indicated by the web page elements and execute code of the web page elements; and means for communicating wagering game events to the plurality of web browser add-on containers based on registered interest in the wagering game events by the plurality of web browser add-on containers.

BRIEF DESCRIPTION OF THE FIGURES

[0010] Embodiments of the invention are illustrated in the Figures of the accompanying drawings in which:

[0011] Figure 1 depicts an example conceptual diagram of a framework for dynamic construction of a web page source to leverage multiple technologies.

[0012] Figure 2 depicts an example conceptual diagram for selecting a web page construction code based on an index value.

[0013] Figure 3 depicts a flowchart of example operations for rendering a web page from a dynamically built up web page source.

[0014] Figure 4 depicts a flowchart of example operations for communicating events to registered containers.

[0015] Figure 5 depicts a flowchart of example operations for additional construction on a web page source.

[0016] Figure 6 is a block diagram illustrating a wagering game machine architecture, according to example embodiments of the invention.

[0017] Figure 7 is a block diagram illustrating a wagering game network 700, according to example embodiments of the invention.

DESCRIPTION OF THE EMBODIMENTS

[0018] The description that follows includes exemplary systems, methods, techniques, instruction sequences, and computer program products that embody techniques of the present inventive subject matter. However, it is understood that the described embodiments may be practiced without these specific details. For instance, although examples refer to web browsers operating in a client-server model to request resources from remote locations, embodiments are not limited. Embodiments can use resources local to the machine hosting the web browser. In other instances, well-known instruction instances, protocols, structures, and techniques have not been shown in detail in order not to obfuscate the description.

[0019] Web browsers have become ubiquitous across machines/devices, and allow technological flexibility with extensions, plug-ins, extensible markup language, etc. Wagering game systems can leverage web browsers to present wagering game content with a variety of technologies. For instance, an electronic wagering game table can present diverse wagering game content (e.g., multiple wagering games, portal games, social networking applets, hospitality menus, etc.) with a variety of technologies that interface with a web browser. Also, a single wagering game can be constructed from multiple technologies. A wagering game system can implement a framework that dynamically builds up a web page to use multiple technologies for presenting content with a web browser on a wagering game system. The framework can dynamically construct the web page, which can involve modifying objects, embedding objects, removing objects, and replacing objects that correspond to different technologies.

[0020] Figure 1 depicts an example conceptual diagram of a framework for dynamic construction of a web page source to leverage multiple technologies. In Figure 1, a web browser instance 103 is depicted. The web browser instance 103 is an executing instance of a web browser program (e.g., Mozilla's Firefox® web browser, Microsoft's Internet Explorer® web browser, the Opera Software's Opera browser, Apple's Safari® browser). Although not depicted, a machine (e.g., wagering game table, wagering game terminal, portable wagering game machine, mobile phone, etc.) hosts the web browser instance. In Figure 1, the web browser instance interacts with a configuration server 105. Thus, the machine that hosts the web browser instance 103 communicates over any one of the Internet, a local area network (e.g., a network within wagering game establishment), and a wide area network (e.g., a network that spans multiple wagering game establishments). Figure 1 depicts several stages (A-D) of operations for the framework to dynamically construct a web page source that presents wagering

game content with multiple technologies. Of course, embodiments are not limited to four stages of operations.

[0021] At stage A, the web browser instance 103 loads a base web page source 101A. The base web page source may be a hypertext markup file, an extensible markup language file, etc. The base web page source 101A comprises code to request web page construction code and code to instantiate an event queue manager. It is not necessary for the base web page source to comprise the event queue manager instantiation code. Embodiments can implement the event queue manager instantiation code separate from the base web page source (e.g., as part of the web page construction code, incorporated into the web browser program, etc.).

[0022] At stage B, the web browser instance 103 requests the web page construction code and instantiates an event queue manager. Responsive to loading of the base web page at stage A, a rendering engine 111 of the web browser instance 103 recognizes a tag for a language. In this example, the rendering engine 111 recognizes a tag for “Script A” in the base web page source 101A. Embodiments are not limited to script languages, though. Any of a variety of programming language can be used. A short list of example rendering engines (also referred to as layout engines) includes Netscape Communications Corporation’s Gecko® layout engine, Apple’s WebKit layout engine, the Trident layout engine, and the Presto layout engine. The rendering engine 111 processes (e.g., interprets, translates, etc.) the marked up content of the base web page source 101A. After recognizing the programming language tag, the rendering engine 111 invokes a code execution engine, which is not depicted, to execute the embedded web construction request code. A few examples of code execution engines include the Rhino script execution engine, the SpiderMonkey script execution engine, Tamarin script execution engine, V8 script execution engine, the Chakra script execution engine, the Carakan script execution engine. The code execution engine executes the web construction request code to request web construction code 107 from the configuration server 105. The code execution engine also executes the event queue manager instantiation code to instantiate an event queue manager 109.

[0023] At stage C, the web browser instance 103 executes the web page construction code to build up the base web page. As a result of the web browser instance 103 (vis-à-vis the code execution engine) executing the web construction code received from the configuration server 105, a configuration process 113 is spawned in this example. Although it is not necessary for a separate process to be spawned, depicting a separate process is at least helpful for illustrative

purposes. Execution of the web page construction code causes the configuration process 113 modify the base web page source 101A. The configuration process 113 embeds tags and code of various technologies in accordance with the web page construction code. Figure 1 depicts the configuration process 113 adding code for an add-on technology A and code for an add-on technology B to the base web page source 101A, resulting in a web page source 101B. The various add-on technologies may be responsible for presenting different aspects of the web page including video, animation, social networking tools, browser appearance, etc. Examples of the various technologies added to the base web page source include the Flash® multimedia platform, Unity gaming engine, TM HTML5 Video, Microsoft Silverlight, JavaFX, Unity, etc. At this point, the web browser instance 103 may not present anything to a user yet.

[0024] At stage D, the rendering engine 111 renders the built up web page and instantiates a program(s) for different add-on technologies (“add-on technology containers”) based on the built up web page source 101B. The rendering engine 111 begins to process the built up web page source 101B. As the rendering engine 111 processes the web page source 101B, the rendering engine 111 invokes appropriate programs to execute the code embedded in the built up web page source 101B for the various technologies. In this example, the web page source 101B indicates add-on technology A and add-on technology B. Accordingly, add-on technology A container 119 and add-on technology B container 121 are instantiated. The web browser instance 103 and the add-on technology container A pass code and/or data via an interface 120. The web browser instance 103 and the add-on technology B container 121 pass code and/or data via an interface 122. With the add-on technology containers 119, 121, the web browser instance 103 renders wagering game content in a web browser window 117.

[0025] In this depicted example, the rendering engine 111 renders wagering game content that includes a bonus automobile racing game 125, a slot game 127, and a control panel 129. The rendering engine 111 renders the bonus automobile racing game 125 in accordance with animation technology. The rendering engine 111 renders the slot game 127 in accordance with a three-dimensional (3D) game authoring technology. The rendering engine 111 renders the control panel 129 in accordance with an aggregation of technologies (e.g., AJAX, dynamically loaded internal objects in the Unity3D® game development tool, the Silverlight® development platform, etc.). Assuming the add-on technology A container 119 implements animation technology, then the rendering engine 111 renders the animation for the bonus 125 in accordance with output of the add-on technology A container 119. Assuming the add-on technology B

container 121 implements a 3D gaming engine, the rendering engine 111 renders the slot game 127 in accordance with output from the add-on technology B container 121. Although a third technology container is not depicted, the rendering engine 111 renders the control panel 129 in accordance with output of a third technology container that corresponds to the aggregation of technologies and output of the rendering engine 111. Hence, in this example, the web browser instance 103 presents wagering game content with multiple different technologies in a single web browser window. Further, multiple technologies can be employed to present different aspects of a single wagering game. For instance, the web browser instance 103 can employ 3D gaming engine technology to present reels of the slot game 127, and employ animation technology to present background of the slot game 127. In another example, the web browser instance employs Silverlight technology to show a streaming video or content from a video recorder. The web browser instance uses Javascript® technology or HTML5 based technology to tie into social networking sites. And the web browser instance uses Flash technology to provide a menu for a user to select a type of game content.

[0026] The web page that is eventually presented by a web browser arises from a web page source that evolves from a base web page source modified by execution of the web page construction code retrieved from a server, although the web page construction code could be local and distinct from the base web page source. The web page construction code that is supplied to the web browser instance can be selected from a variety of web page construction code. A store can host different web page construction codes for different machines, locations, wagering games, environments, temporal values, etc. For example, a web page construction code can be selected based on time of year, particular casino, and type of wagering game machine. Figure 2 depicts one of many ways to select a web page construction code.

[0027] Figure 2 depicts an example conceptual diagram for selecting a web page construction code based on an index value. In this example, a web browser instance 201 (e.g., on a wagering game machine) communicates with a server process 209. The web browser instance 201 has access to a web page source 205. The server process 209 has access to a store of web page construction codes 213, 215, 217, via an index data structure 211.

[0028] At stage A, the web browser instance 201 loads the web page source 205, which is a base web page source at this point. Loading of the web page source 205 causes the web browser instance 201 to request web page construction code.

[0029] At stage B, the web browser instance 201 generates and transmits a request 207 for the web page construction code. In this example, the request 207 indicates an identifier “A.” The identifier can represent any one or more of wagering game, wagering game developer, wagering game machine, owner of the wagering game machine, a license identifier, etc. The identifier can be hard coded into the web page construction code, or determined from execution of the web page construction request code. Execution of the web page construction code can generate a query submitted to an operating system, a lookup in a data store identified in the web page construction request code, etc. Furthermore, the web page construction request code can determine additional variable values that affect web page construction code selection. Examples of other variable values include player identification data, time of day, time of year, eligibility of the wagering game machine in bonus games, web browser version, etc.

[0030] At stage C, the server process 209 looks up a web page construction code in accordance with the request 207. For this example, the identifier “A” corresponds to an “Index A” in the index data structure 211. The entry for “Index A” references the web page construction code 213. Although this example illustrates an index lookup, embodiments are not so limited. Embodiments can evaluate rules or conditions against the request to determine which web page construction code should be selected. In addition, embodiments can assemble or build a web page construction code. For instance, an embodiment can select different web page construction codes based on information in the request from the web browser instance. A first web page construction code can construct a control panel for wagering game when executed. A second web page construction code can construct a video for the wagering game when executed. A third web page construction code can construct a hospitality services menu when executed, and so on. The server process 209 can assemble these different web page construction codes into a single executable file.

[0031] Returning to the illustrated example, the server process 209 returns the selected web page construction code 213 to the web browser instance 201 at stage D. Embodiments can return the web page construction code 213, or a reference to the web page construction code 213.

[0032] At stage E, the web browser instance 201 executes the web page construction code 213. For instance, a TM Javascript interpreter interprets the web page construction code 213.

[0033] Pursuant to execution of the web page construction code 213, the web browser instance 201 modifies the web page source 205 at stage F. The web browser instance 201 embeds code, tags, and objects into the web page source 205 in accordance with the web page construction

code 213. In this example, the web browser instance 201 inserts a video player object of a technology A, and a gaming engine object of a technology B. Although this example suggests literal embedding, web page construction can involve referential embedding of code or content (e.g., a reference to a file to be presented or executed by an add-on technology container).

[0034] A web browser instance can either render a web page as the web page source is augmented or built up, or the web browser instance can wait until the web page source completes construction. When rendering the web page, the web browser instance instantiates or launches technology containers for the different technologies of web page elements in the web page source. The technology containers, sometimes referred to as application containers, hold or execute commands or programs and/or present content of a particular technology, separate from the web browser instance (e.g., an Adobe Flash container presents Flash content). In some embodiments, a container of a first technology can be used to present content of a second technology. For instance, a Unity container can be used to present Flash based content. For brevity, the term “container” is used herein to refer to both a set of instructions (e.g., program) for a container and an executing instance of instructions for a container.

[0035] Figure 3 depicts a flowchart of example operations for rendering a web page from a dynamically built up web page source. At block 301, a loop of operations begins for each web page element or object. Examples of web page elements include applets, images, videos, fields, panels, etc.

[0036] At block 303, it is determined if the web page element indicates a web browser add-on technology container. For instance, tag attributes are examined to determine if a particular technology is indicated for the element. If the web page element does not indicate a web browser add-on technology, then control flows to block 305. If the web page element indicates a web browser add-on technology, then control flows to block 307.

[0037] At block 305, the element is rendered. The element rendered did not require an add-on technology container. For instance, a web browser renders text or an image.

[0038] At block 307, it is determined if the web browser add-on technology container is currently instantiated. For example, a web browser determines whether a container for a video has already been instantiated. If the add-on technology container is not currently instantiated, then control flows to block 309. If the add-on technology container is currently instantiated, then control flows to block 313.

[0039] At block 309, the web browser add-on technology container is instantiated. For instance, a web browser instance can invoke an Adobe Flash player TM. The web browser can instantiate an interpreter or compiler to interpret or compile code.

[0040] At block 311, a technology container state structure is updated. The web browser instance updates the technology container state structure to indicate that the add-on technology container is currently instantiated. When an add-on technology container is terminated, the technology container state structure is updated accordingly. Embodiments are not required to maintain a structure to indicate state of add-on technology containers. Embodiments can leverage operating system facilities to determine whether an add-on technology container is currently instantiated. Although not represented in Figure 3, embodiments are not limited to relying on a single instance of an add-on technology container, and can also instantiate an add-on technology container multiple times.

[0041] At block 313, the web page element is rendered with the web browser add-on technology container instance. For instance, the web browser instance passes content for a video to an instantiated video player container. The video player container then plays the video.

[0042] At block 315 loop control ends. If the web page source includes another web page element, then control flows back to block 301. Otherwise, the flow ends.

[0043] After a web page is rendered and the add-on technology containers are instantiated, the web page and add-on technology containers can respond to events and perform operations accordingly. As illustrated in Figure 1, an event queue manager is instantiated to manage an event queue for the web browser instance. Local events (e.g., events generated from add-on technology containers or from the host device) and remote events (e.g., events from bonus games, community games, casino events, etc.) are stored in the event queue. Add-on technology containers, a configuration process, and the web browser instance itself can register interest in types of events and sources of events. For instance, an add-on technology container can register with the event queue manager an interest in all events or particular types of events from a specific bonus games. As an example, a configuration process can register an interest in all configuration type of events regardless of the source. As another example, multiple add-on technology containers can register an interest in wagering game events. The event queue manager can multicast a wagering game event to the multiple add-on technology containers. When each of the add-on technology containers consume the event, they each perform corresponding operations. To illustrate, a spin game event is communicated to a video player

container, an audio player container, and a TM Java virtual machine container. The video player container plays a reel spin video in response to the event. The audio player container plays sound effects based on player history and the reel spin. The Java virtual machine executes code that changes a control panel to depict a pressed spin button.

[0044] The event queue manager can also perform operations to maintain the event queue beyond reporting or communicating events to entities. The event queue manager can accumulate or aggregate events and store a single data item that represents an accumulation or aggregation of events. Entities can use operating system inter-process communications facilities to learn of events of interest. Entities can also register with the event queue manager for direct communication of events. In other words, whether local or remote, the event queue manager can communicate an event directly to a process. In case of conflicts, the event queue manager can also enforce prioritization among registered processes for direct communication. For instance, the event queue manager may only communicate event A to the construction process even though both the construction process and an add-on technology container are registered for direct communication, assuming the web page construction process has a higher priority than the add-on technology container.

[0045] Figure 4 depicts a flowchart of example operations for communicating events to registered containers. At block 401, an event is detected. For instance, a web browser instance receives an event from a wagering game server. A web browser can also detect an event generated by any one of the web browser instance, an add-on technology container, a configuration process, and the host device/machine.

[0046] At block 403, it is determined if a container is registered for exclusive direct communication. If a container is registered for direct communication of the event, then control flows to block 415. In the case of direct communication, then the event should be communicated to the registered container without action being taken by the event queue manager. Hence, the event queue manager should be able to discard the event. A container, however, may not be registered for exclusive direct communication. In other words, other containers can be subscribed to an event even though another container has registered for direct communication. One example of this scenario would be if a first technology container needs to quickly exchange messages with the process or application that generates the event. Meanwhile, a second and third technology container update their state based on the event, but without replying to the

process or application that generates the event. If a container is not registered for exclusive direct communication of the event, then control flows to block 404.

[0047] At block 404, the event is enqueued. An event can be a message, data unit, object, etc. The event can contain and/or reference one or more pieces of data. Examples of the pieces of data include an event type (e.g., wagering game event, bonus game event, hospitality event, etc.), time of the event, source of the event (e.g., a particular bonus game, a network address of a server, etc.), data type (e.g., integer, float, string, defined class, etc.), and attributes of the event (e.g., amount wagered, amount won, cost of a beverage, a receipt, a virtual asset, etc.).

[0048] At block 410, an event is dequeued. The dequeued event may or may not be the event enqueued at block 404. The event queue manager can manage the event queue in accordance with a variety of techniques. The event queue manager can comprise a first set of one or more threads or processes to perform enqueue operations and a second set of one or more threads or processes to perform dequeued operations. The event queue manager can also switch back and forth between enqueue and dequeued operations.

[0049] At block 411, it is determined if there are any subscribers to the event. If there are no subscribers to the event, then control flows to block 415. If there are subscribers to the event, then control flows to block 413.

[0050] At block 413, the event is communicated to the subscriber container(s) via the corresponding interface(s). For example, the event queue manager calls a method or function defined by the interface that passes the data (e.g., a packet) to the subscriber container. As another example, the event queue manager could pass a reference to the event data via the interface.

[0051] At block 415, the event is discarded. The event queue manager can mark the event for removal to a heap, deletion, etc. The event queue manager or an associated process can log detection and/or discard of events.

[0052] Although Figure 4 depicts enqueue and dequeue, embodiments can depart from the depicted ordering. An event queue manager can enforce prioritization among either or both of registrants and events. For instance, events can regularly be stored in a FIFO order. But the event queue manager can evaluate each event to determine whether the event should be placed higher within the event queue, or even whether the event should be communicated immediately without queuing.

[0053] Figure 4 also was limited to depicting event consumption by technology containers. Events can also be consumed by a configuration process. Although earlier examples illustrated the initial web page source construction, the configuration process can further alter a web page source after the initial build up. Various events can trigger a configuration process to add, remove, replace, and modify web page elements. For example, a player may wish to swap out wagering games. The new wagering game may use a different technology container than the one being swapped out, while using the same technology containers for the hospitality services menu and for an ongoing bonus game.

[0054] Figure 5 depicts a flowchart of example operations for additional construction on a web page source. The example operations of Figure 5 are predicated on the assumption that events trigger additional constructions. For instance, a change in a bonus game or a win can trigger construction operations. Control is depicted as flowing from block 401 of Figure 4 to block 501 of Figure 5. At block 501, it is determined if a configuration process is registered or subscribed to the event. If the configuration process is registered for the event, then control flows to block 503. If not, then control flows to block 403 of Figure 4.

[0055] At block 503, construction code that corresponds to the event is retrieved. The construction code can be retrieved from a local or remote store. The constructions code can be retrieved based on information indicated by the event. For examples, the construction can be retrieved based on any one of an event type, an event code, a source of the event, and a reference to the construction code in the event. The configuration process may wait for a bonus game event that indicates the bonus game has been concluded. The construction for that bonus game event may require construction of a new bonus game web page element or tearing down of the web page element(s) that correspond to the bonus game,

[0056] At block 505, loop of operations begins for each element indicated in the retrieved construction code. In this example flowchart, the loop of operations includes operations to determine state of a relevant technology container and operations to implement the construction.

[0057] At block 507, the type of construction operation for the web page element is determined. For example, it is determined whether the construction code calls for removing the web page element, modifying the web page element, replacing the web page element, or inserting the web page element.

[0058] At block 509, it is determined if the operation will affect an active technology container. For instance, an add-on technology container may be operating upon the web page

element. For the configuration process to modify the web page element, the add-on technology container may be suspended or terminated. If the construction operation will affect an active container, then control flows to block 511. If the construction operation will not affect an active container, then control flows to one of blocks 513, 517, 521, and 525.

[0059] At block 511, the active container is interrupted. For instance, a configuration process submits a request to an operating system to terminate the active container. As another example, the configuration process submits a request to the active container for the active container to enter a pause or stall type of state. The active container can then inform the configuration process when it has entered a state of not, at least, writing data corresponding to the web page element, and wait until a notification from the configuration process allows the paused/stalled active container to resume or begin accessing data corresponding to the web page element. The configuration process can also request the active container to terminate itself, instead of requesting through the operating system. An active container may not be affected by a construction operation if the active container does not operate upon any data corresponding to the web page element; if the active container is currently operating upon data that is independent of the web page element, and the construction operation will complete beforehand; etc.

[0060] If the construction operation is a remove type of construction operation, then control flows to block 513. At block 513, the web page element being removed from the web page source is located. For example, the configuration process searches for a particular tag or string of the web page element.

[0061] At block 515, the web page element is removed from the web page source. For instance, the configuration process can make a copy of the web page source. The configuration process can then remove the web page element from the copy of the web page source, which will replace the web page source. In another example, the configuration process clears a memory space that hosts the web page element after clearing any references or dependencies upon the data in that memory space.

[0062] Control flows from block 515 to block 529, which determines whether the loop ends. If there are additional web page elements, then control flows from the block 529 back to block 505. If there are no additional web page elements, then the flow ends at block 529.

[0063] If the construction operation is a modify type of construction operation, then control flows to block 517. At block 517, the web page element is located in the web page source.

[0064] Then, at block 519, an attribute of the web page element is modified. For instance, the configuration process modifies a value assigned to an attribute of an object. For example, screen coordinates may be modified to resize or move an object. A drag/drop property can be modified to allow or disallow a player from moving the web page element. A value of a visibility attribute of an object or the web page element can be modified to temporarily make the object or web page element visible or not visible. Control flows from block 519 to block 529.

[0065] If the construction operation is a replace type of construction operation, then control flows to block 521. At block 521, the web page element to be replaced is located in the web page source. For instance, the configuration process indicates a web page element to be replaced and a new web page element that will replace it.

[0066] At block 523, the web page element to be replaced is overwritten with the new web page element. If the web page elements occupy different sizes of memory, then the configuration process will adjust the web page source accordingly. Control flows to block 529.

[0067] If the construction operation is a insert type of construction operation, then control flows to block 525. At block 525, an insertion is located within the web page source.

[0068] At block 527, the web page element is inserted into the web page source at the insertion point. For example, the web page source is segmented in memory. The new web page element is then appended to a first segment in memory. And then a second segment is appended to the new web page element. As another example, the new web page element is appended to the web page source and branches are inserted to direct flow of the web page source accordingly. Control flows to block 529 from block 527.

[0069] In some embodiments, the operations of Figures 3-5 can be performed by logic not described in the block diagrams. In certain embodiments, the operations can be performed by executing instructions residing on machine-readable media (e.g., software), while in other embodiments, the operations can be performed by hardware and/or other logic (e.g., firmware). In some embodiments, the operations can be performed in series, while in other embodiments, one or more of the operations can be performed in parallel. Moreover, some embodiments can perform less than all the operations shown in any flow diagram.

Wagering Game Machine Architectures

[0070] Figure 6 is a block diagram illustrating a wagering game machine architecture, according to example embodiments of the invention. As shown in Figure 6, the wagering game

machine architecture 600 includes a wagering game machine 606, which includes a central processing unit (CPU) 626 connected to main memory 628. The CPU 626 can include any suitable processor, such as an Intel® Pentium processor, Intel® Core 2 Duo processor, AMD Opteron™ processor, or UltraSPARC processor. The main memory 628 includes a wagering game unit 632 and a web page construction unit 636. In one embodiment, the wagering game unit 632 can present wagering games, such as video poker, video black jack, video slots, video lottery, etc., in whole or part. The web page construction unit 636 performs operations that involves building up or constructing an initial web page source to include web page elements of various technologies. For instance, the web page construction unit can perform the examples operations depicted in Figure 5.

[0071] The CPU 626 is also connected to an input/output (I/O) bus 622, which can include any suitable bus technologies, such as an AGTL+ frontside bus and a PCI backside bus. The I/O bus 622 is connected to a payout mechanism 608, primary display 610, secondary display 612, value input device 614, player input device 616, information reader 618, and storage unit 630. The player input device 616 can include the value input device 614 to the extent the player input device 616 is used to place wagers. The I/O bus 622 is also connected to an external system interface 624, which is connected to external systems 604 (e.g., wagering game networks).

[0072] In one embodiment, the wagering game machine 606 can include additional peripheral devices and/or more than one of each component shown in Figure 6. For example, in one embodiment, the wagering game machine 606 can include multiple external system interfaces 624 and/or multiple CPUs 626. In one embodiment, any of the components can be integrated or subdivided.

[0073] Any component of the architecture 600 can include hardware, firmware, and/or machine-readable media including instructions for performing the operations described herein. Machine-readable media includes any mechanism that stores (“machine-readable storage media”) and/or transmits (“machine-readable signal media”) information in a form readable by a machine (e.g., a wagering game machine, computer, etc.). Examples of machine-readable storage media include read only memory (ROM), random access memory (RAM), magnetic disk storage media, optical storage media, flash memory machines, etc. Examples of machine-readable signal media includes media suitable for transmitting software over a network.

[0074] While Figure 6 describes an example wagering game machine architecture, this section continues with a discussion wagering game networks.

Wagering Game Networks

[0075] Figure 7 is a block diagram illustrating a wagering game network 700, according to example embodiments of the invention. As shown in Figure 7, the wagering game network 700 includes a plurality of casinos 712 connected to a communications network 714.

[0076] Each casino 712 includes a local area network 716, which includes an access point 704, a web page configuration server 706, and wagering game machines 702. The access point 704 provides wireless communication links 710 and wired communication links 708. The wired and wireless communication links can employ any suitable connection technology, such as Bluetooth, 802.11, Ethernet, public switched telephone networks, SONET, etc. The web page configuration server 706 serves configuration code to wagering games 702 that request construction code. The web page configuration server 706 can also look up various configuration codes based on information in the requests and/or environment information (e.g., casino defined variables, time of day, holiday, current convention, etc.). In some embodiments, the web page configuration server 706 can also serve wagering games and distribute content to devices located in other casinos 712 or at other locations on the communications network 714.

[0077] The wagering game machines 702 described herein can take any suitable form, such as floor standing models, handheld mobile units, bartop models, workstation-type console models, etc. Further, the wagering game machines 702 can be primarily dedicated for use in conducting wagering games, or can include non-dedicated devices, such as mobile phones, personal digital assistants, personal computers, etc. In one embodiment, the wagering game network 700 can include other network devices, such as accounting servers, wide area progressive servers, player tracking servers, and/or other devices suitable for use in connection with embodiments of the invention.

[0078] In some embodiments, wagering game machines 702 and wagering game servers 706 work together such that a wagering game machine 702 can be operated as a thin, thick, or intermediate client. For example, one or more elements of game play may be controlled by the wagering game machine 702 (client) or the web page configuration server 706 (server). Game play elements can include executable game code, lookup tables, configuration files, game outcome, audio or visual representations of the game, game assets or the like. In a thin-client example, the web page configuration server 706 can perform functions such as determining game outcome or managing assets, while the wagering game machine 702 can present a graphical representation of such outcome or asset modification to the user (e.g., player). In a

thick-client example, the wagering game machines 702 can determine game outcomes and communicate the outcomes to the web page configuration server 706 for recording or managing a player's account.

[0079] In some embodiments, either the wagering game machines 702 (client) or the web page configuration server 706 can provide additional functionality that is not directly related to serving configuration code. For example, account transactions and account rules may be managed centrally (e.g., by the web page configuration server 706) or locally (e.g., by the wagering game machine 702). Other functionality not directly related to game play may include power management, presentation of advertising, software or firmware updates, system quality or security checks, etc.

[0080] Any of the wagering game network components (e.g., the wagering game machines 702) can include hardware and machine-readable media including instructions for performing the operations described herein.

General

[0081] This detailed description refers to specific examples in the drawings and illustrations. These examples are described in sufficient detail to enable those skilled in the art to practice the inventive subject matter. These examples also serve to illustrate how the inventive subject matter can be applied to various purposes or embodiments. Other embodiments are included within the inventive subject matter, as logical, mechanical, electrical, and other changes can be made to the example embodiments described herein. Features of various embodiments described herein, however essential to the example embodiments in which they are incorporated, do not limit the inventive subject matter as a whole, and any reference to the invention, its elements, operation, and application are not limiting as a whole, but serve only to define these example embodiments. This detailed description does not, therefore, limit embodiments of the invention, which are defined only by the appended claims. Each of the embodiments described herein are contemplated as falling within the inventive subject matter, which is set forth in the following claims.

CLAIMS

1. A method of creating a browser based heterogeneous technology ecosystem, the method comprising:
 - a web browser loading a web page source that comprises configuration code;
 - the web browser executing the configuration code on a machine that hosts the web browser;
 - in response to said executing the configuration code,
 - instantiating an event queue manager that manages an event queue responsive to said executing the configuration code;
 - requesting web page construction code that indicates web page elements responsive to said executing the configuration code, wherein the web page elements comprise a first web page element that corresponds to a first technology and a second web page element that corresponds to a second technology;
 - building up the web page source to include the web page elements in accordance with the web page construction code resulting in a built up web page source;
 - rendering the built up web page source, wherein said rendering the built up web page source comprises,
 - instantiating a first add-on technology container to render the first web page element; and
 - instantiating a second add-on technology container to render the second web page element.
2. The method of claim 1 further comprising the first add-on technology container either executing code of the first web page element in accordance with the first technology or presenting content of the first web page element in accordance with the first technology.
3. The method of claim 2, wherein the first add-on technology container comprises one of an interpreter, a virtual machine, an audio player, an animation engine, a game engine, and a video player.
4. The method of claim 1, wherein said web page source comprises at least one of hypertext markup language and extensible markup language.

5. The method of claim 1, wherein said requesting the web page construction code that indicates web page elements comprises generating a construction code request that indicates at least one of a construction code identifier, a wagering game identifier, a wagering game developer identifier, an operating system of wagering game machine, and information about the browser.
6. The method of claim 1 further comprising:
receiving a request that indicates an operation for the first web page element of the web page source;
performing the operation on the first web page element, wherein the operation comprises one of removing, modifying, and replacing;
rendering the web page source after said performing the operation on the first web page element completes.
7. The method of claim 6 further comprising:
terminating the first add-on technology container prior to said performing the operation on the first web page element,
wherein said rendering the web page source after said performing the operation on the first web page element completes comprises re-instantiating the first add-on technology container.
8. The method of claim 6 further comprising:
suspending the first add-on technology container prior to performing the operation on the first web page element,
wherein said rendering the web page source after said performing the operation on the first web page element completes comprises resuming the first add-on technology container.
9. The method of claim 1 further comprising:
receiving a request that indicates an operation to insert a new web page element into the web page source, wherein the new web page element corresponds to the first technology;
terminating or suspending the first add-on technology container;
inserting the new web page element into the web page source; and

rendering the web page source with the inserted new web page element, wherein said rendering the web page source with the inserted new web page element comprises restoring the first add-on technology container to render the new web page element.

10. A method of presenting wagering game content on a wagering game machine with different technologies associated with an instance of a web browser, the method comprising:
 - establishing associations between the instance of the web browser and a plurality of add-on technology containers, wherein the plurality of add-on technology containers correspond to a plurality of technologies and perform at least one of presenting content in accordance with a corresponding one of the plurality of technologies and execute code in accordance with a corresponding one of the plurality of technologies;
 - presenting a first visual component of the wagering game content with a first of the plurality of add-on technology containers;
 - presenting a second visual component of the wagering game content with a second of the plurality of add-on technology containers; and
 - processing game play data of the wagering game content with a third of the plurality of add-on technology containers.
11. The method of claim 10, wherein the first visual component of the wagering game content comprises a video and the first of the plurality of add-on technology containers comprises a video player.
12. The method of claim 10, wherein the second visual component of the wagering game content comprises a wagering game control panel and the second of the plurality of add-on technology containers comprises an animation engine.
13. The method of claim 10, wherein the third of the plurality of add-on technology containers comprises a gaming engine.
14. One or more machine-readable storage media having instructions stored therein, wherein the instructions, when executed, cause a machine to:

modify a web page source to include a plurality of web page objects, wherein the web page objects indicate different technologies for presenting content via a web browser;

instantiate a plurality of web browser add-on containers for the web page objects in accordance with the different technologies; and

manage wagering game events received by an instance of the web browser;

manage subscriptions to wagering game events by the plurality of web browser add-on containers; and

communicate indications of wagering game events to the plurality of web browser add-on containers in accordance with the subscriptions.

15. The machine-readable storage media of claim 14 further comprising instructions to cause the machine, when executed, to instantiate an event queue to host the wagering game events and an event queue manager that enqueues and dequeues the wagering game events to and from the event queue.

16. The machine-readable storage media of claim 14 further comprising instructions, which when executed, cause the machine to collect information about the machine and to generate a request that indicates the information, wherein the request is for the instructions that cause the machine to modify the web page source.

17. The machine-readable storage media of claim 14, wherein the different technologies comprise gaming technology, animation technology, and video technology.

18. The machine-readable storage media of claim 14, wherein the plurality of web page objects comprise a video, game data, and an animation.

19. The machine-readable storage media of claim 18, wherein the plurality of web object also comprise interpretable code.

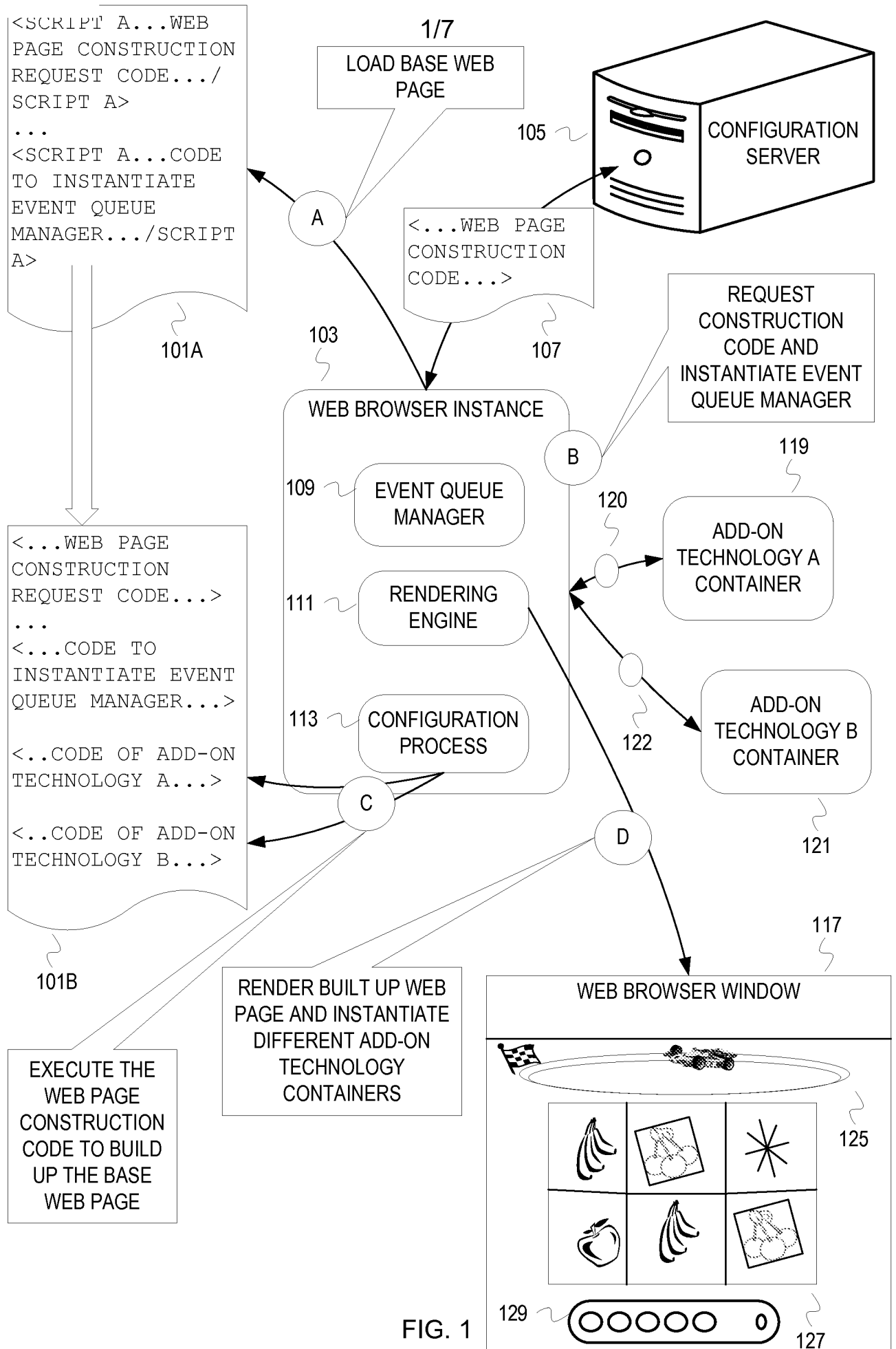
20. One or more machine-readable storage media having instructions stored therein, wherein the instructions, when executed, cause a machine to:

register a set of one or more add-on technology containers as subscribers to an event;

detect the event received by a web browser instance at a wagering game machine,

wherein the web browser instance renders a web page from a web page source

- that comprises web page elements that indicate the set of one or more add-on technology containers;
- determine the set of one or more add-on technology containers as subscribed to the event;
- and
- communicate an indication of the event to the set of one or more add-on technology containers via one or more corresponding interfaces between the set of one or more add-on technology containers and the web browser instance.
21. The machine-readable storage media of claim 20, wherein the set of add-on technology containers comprise at least one of a gaming engine, a video player, and an animation engine.
22. The machine-readable storage media of claim 20 further comprising second instructions to cause the web browser instance to collect information about the machine and to generate a request, which indicates the information, for the instructions that cause the machine to register the set of one or more add-on technology containers as subscribers to the event, wherein the request is transmitted over a network.
23. An apparatus comprising:
- a physical network interface;
- means for loading a web page source and modifying the web page source to include a plurality of web page elements that correspond to different technologies;
- means for instantiating a plurality of web browser add-on containers that implement the different technologies for rendering the web page elements, wherein the plurality of web browser add-on containers perform at least one of present content indicated by the web page elements and execute code of the web page elements;
- and
- means for communicating wagering game events to the plurality of web browser add-on containers based on registered interest in the wagering game events by the plurality of web browser add-on containers.
24. The apparatus of claim 23 further comprising means for suspending a first of the plurality of web browser add-on containers, for replacing a first of the plurality of web page elements in the web page source, and for restoring the first of the plurality of web browser add-on containers afterwards.



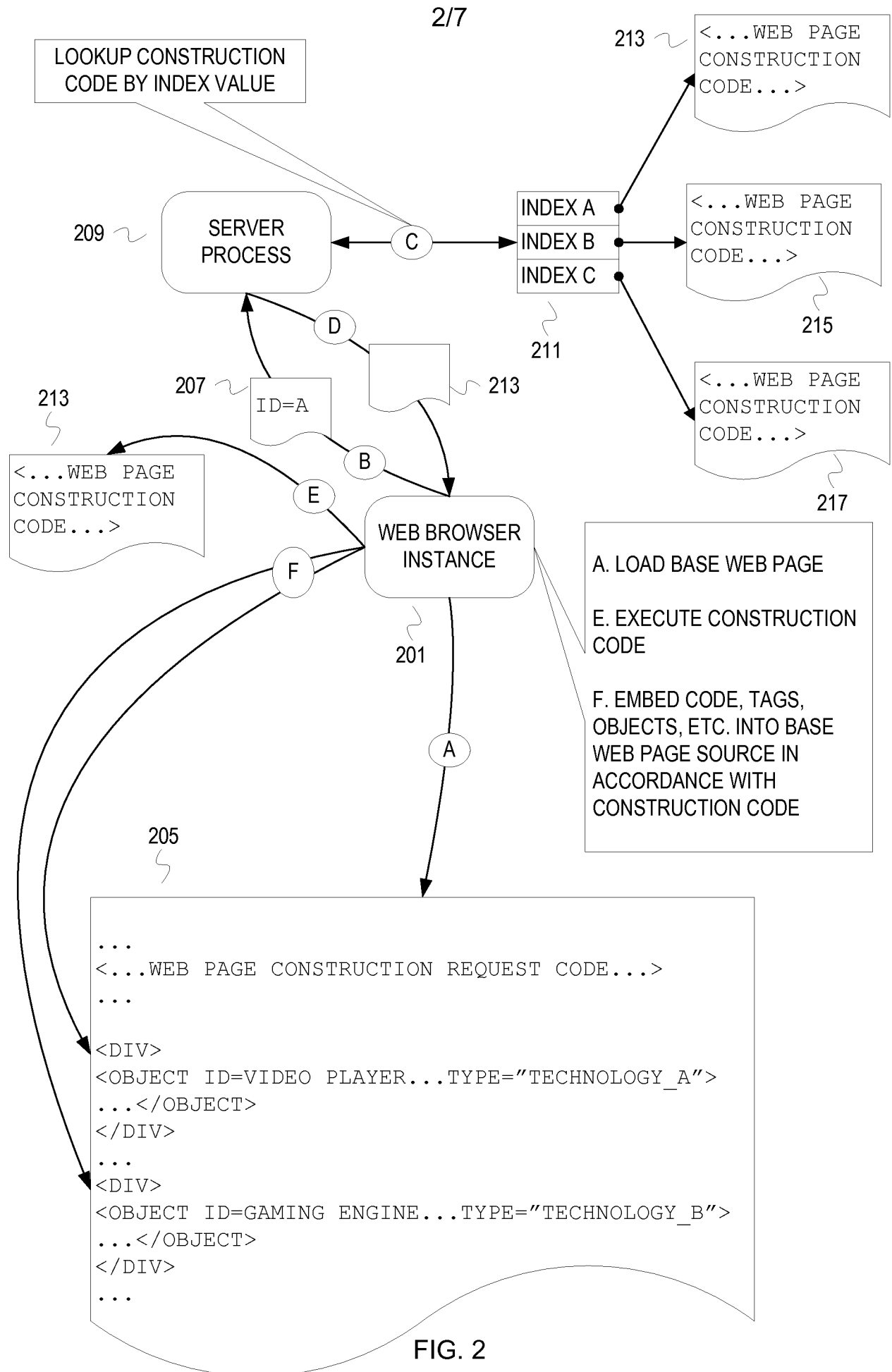


FIG. 2

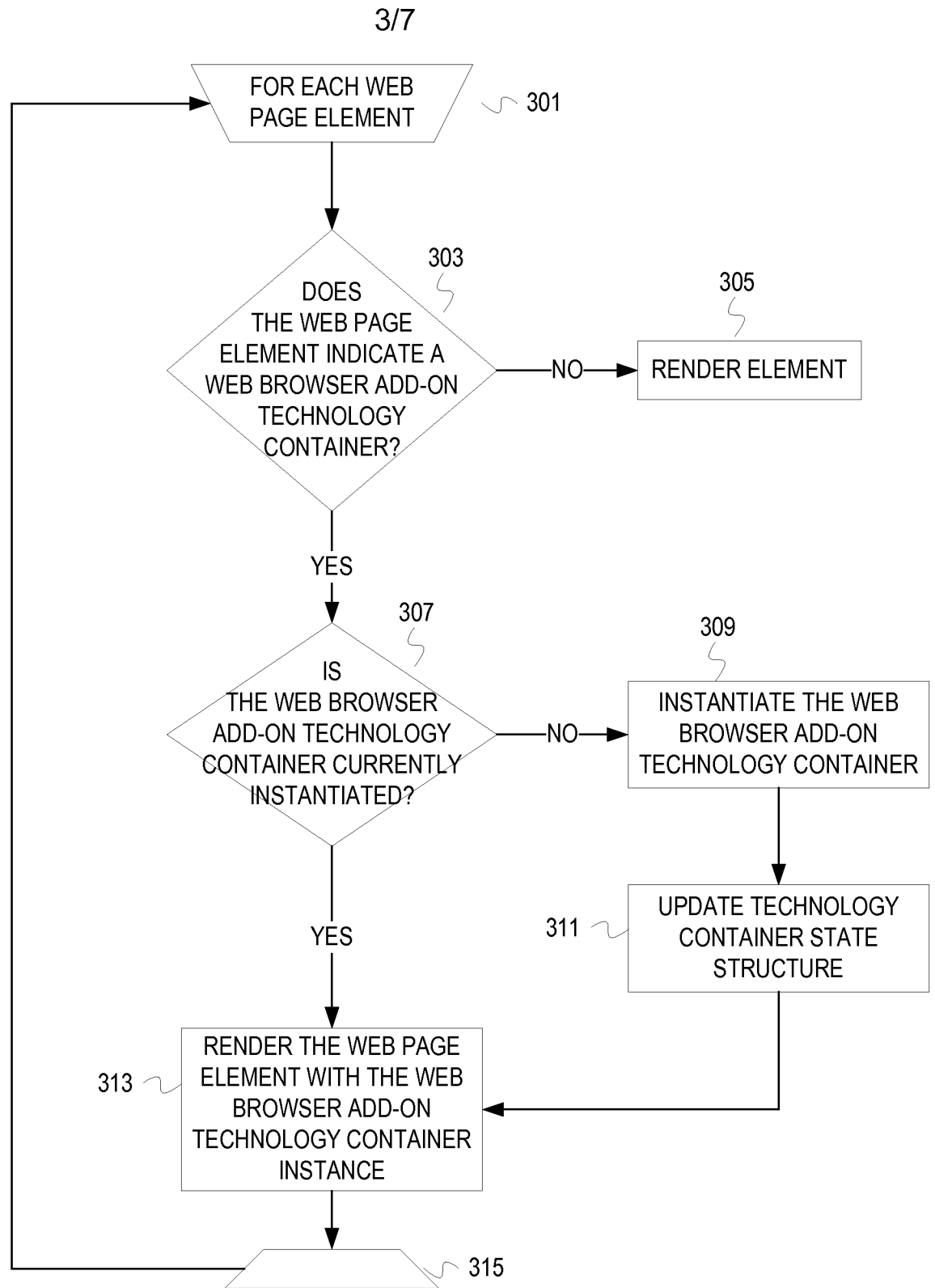


FIG. 3

4/7

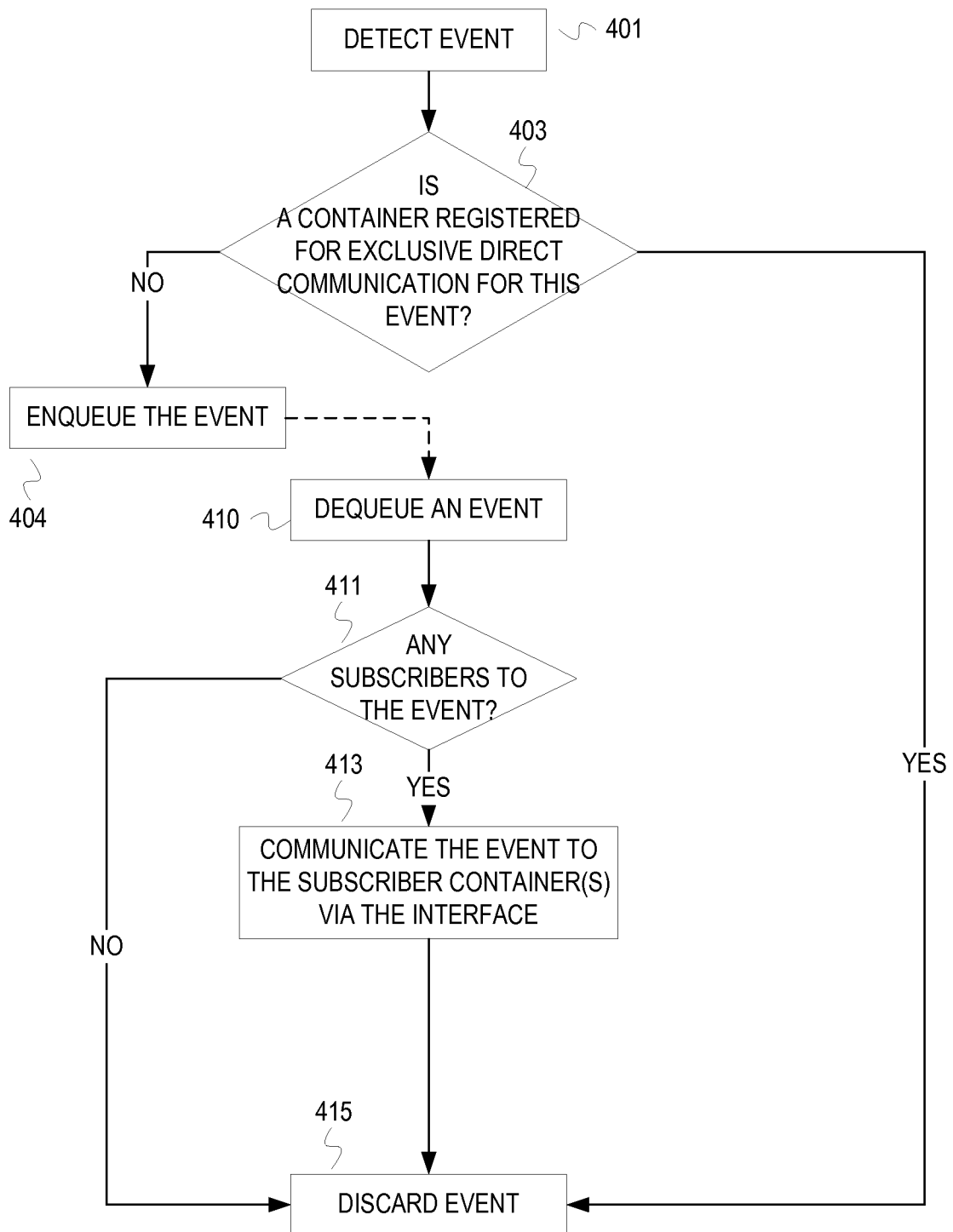


FIG. 4

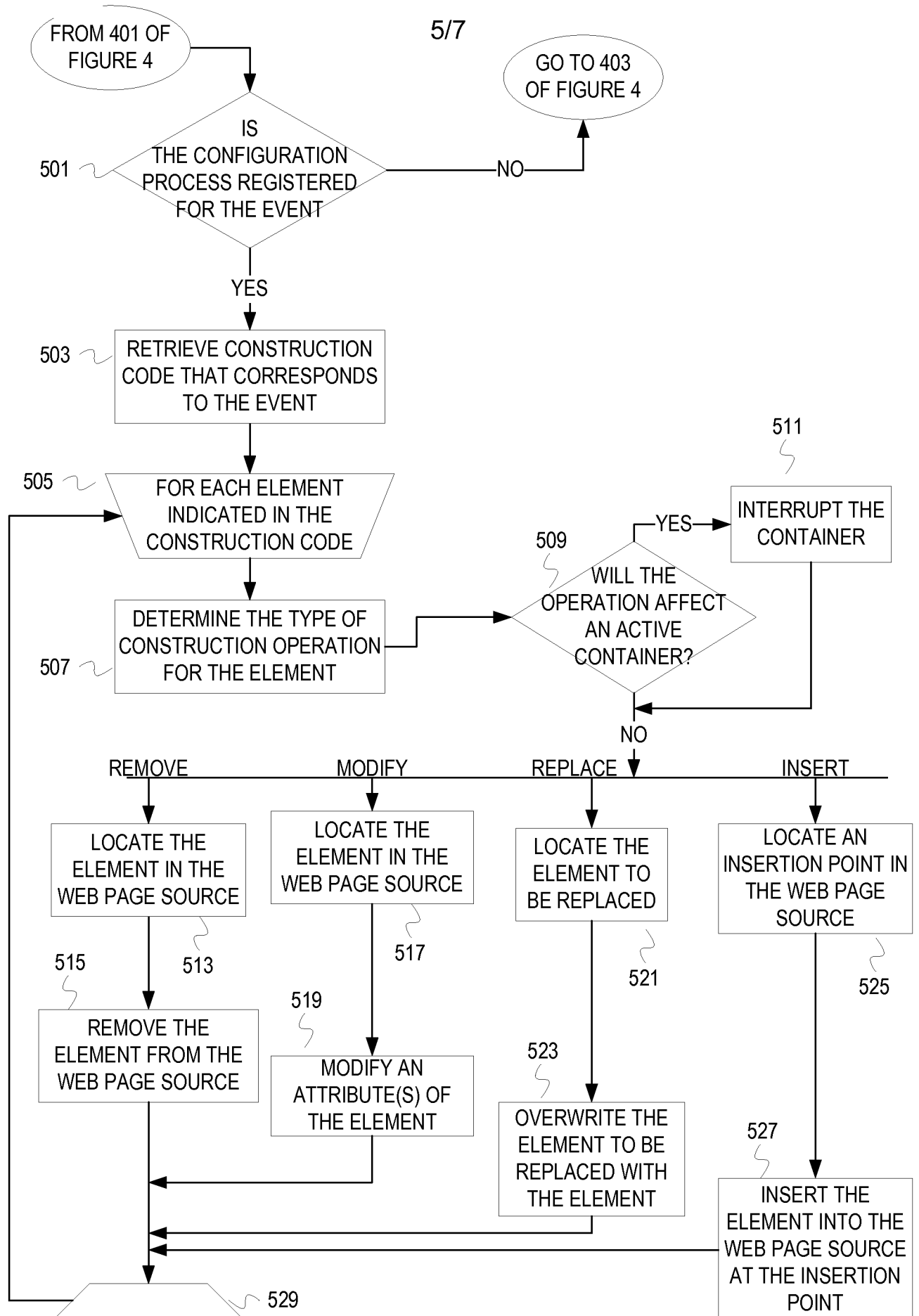


FIG. 5

6/7

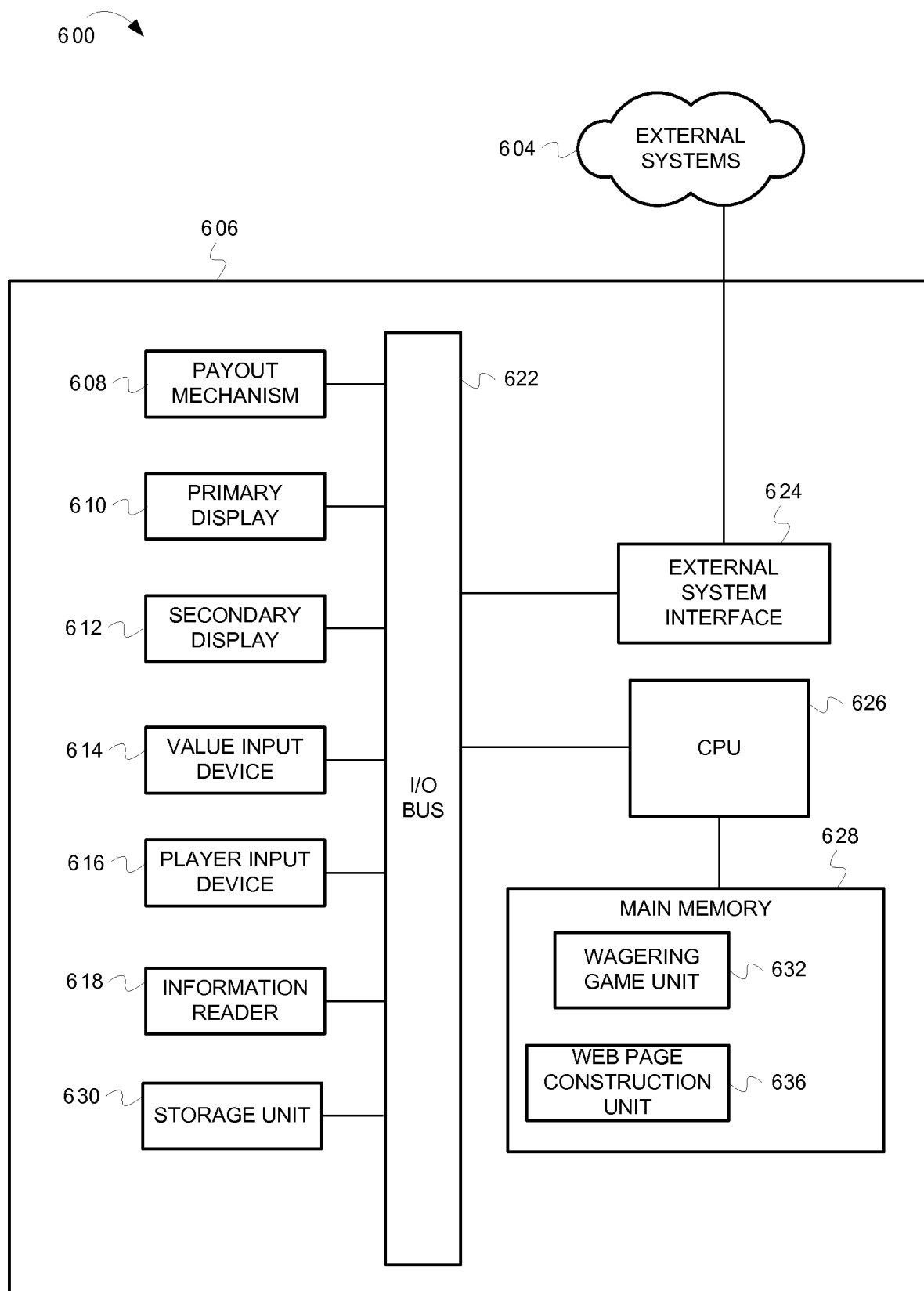


FIG. 6

7/7

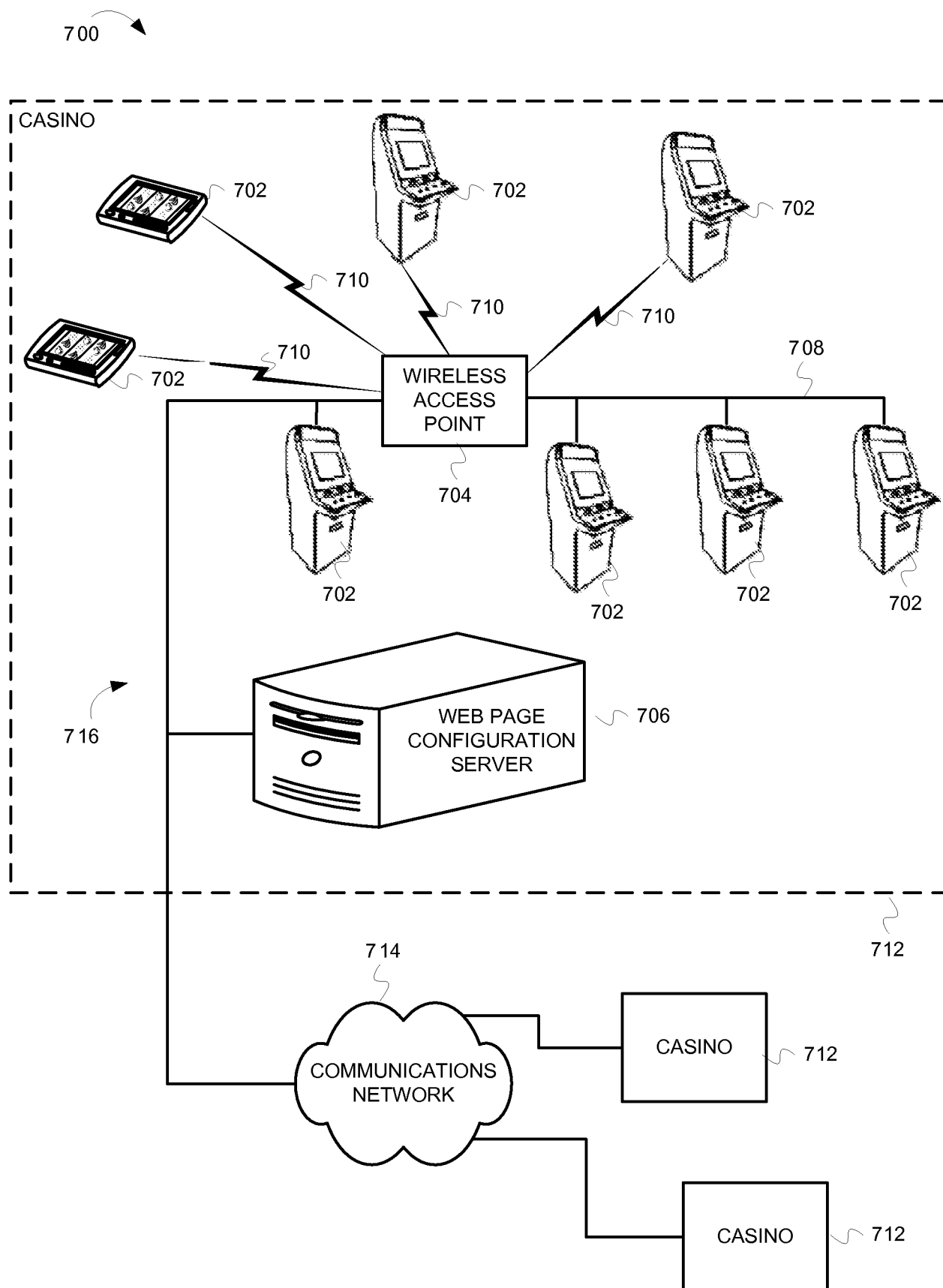


FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 11/46860

A. CLASSIFICATION OF SUBJECT MATTER

IPC(8) - A63F 9/24 (2011.01)

USPC - 463/1

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

USPC: 463/1

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC: 463/1; 463/16; 463/25; 463/42 (keyword limited; terms below)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Electronic Database Searched: PubWEST(PGPB, USPT, EPAB, JPAB), Google Scholar

Search Terms Used: queue, event, manage, manager, event, order, process, processes, element, add, on, add-on, theme, extension, plug, in, script, container, object, applet, image, video, field, site, page, website, webpage, hypertext, document, web, multiple, many

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X — Y	US 2003/0073494 A1 (Kalpakian et al.) 17 April 2003 (17.04.2003), see entire document; especially para [0031], [0063]-[0072], [0175]-[0182], [0197]-[0198], [0209]-[0212], [0224], [0249]-[0254], [0274], [0299], [0301]-[0302], [0350], Fig. 2-3, 18-22	1-10, 12-16, 20-24 ----- 11, 17-19
Y	US 2007/0077978 A1 (Walker et al.) 05 April 2007 (05.04.2007), see para [0093], [0212], [0218]-[0219]	11, 17-19
A	US 2009/0124386 A1 (LaMontagne) 14 May 2009 (14.05.2009), see entire document	1-24
A	US 2003/0188306 A1 (Harris et al.) 02 October 2003 (02.10.2003), see entire document	1-24

☐ Further documents are listed in the continuation of Box C.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

19 December 2011 (19.12.2011)

Date of mailing of the international search report

05 JAN 2012

Name and mailing address of the ISA/US

Mail Stop PCT, Attn: ISA/US, Commissioner for Patents
P.O. Box 1450, Alexandria, Virginia 22313-1450

Facsimile No. 571-273-3201

Authorized officer:

Lee W. Young

PCT Helpdesk: 571-272-4300
PCT OSP: 571-272-7774