(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 May 2008 (02.05.2008)

PCT

(10) International Publication Number
WO 2008/052207 A2

(51) International Patent Classification:
A63F 13/00 (2006.01)

(21) International Application Number:
PCT/US2007/082758

(22) International Filing Date: 27 October 2007 (27.10.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/863,273 27 October 2006 (27.10.2006) 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)*: **ANDERSON, Peter, R.** [US/US]; 616 Elmdale Road, Glenview, Illinois 60025 (US). **GURA, Damon, E.** [US/US]; 928 West Buena Avenue, Number 3, Chicago, Illinois 60613 (US).

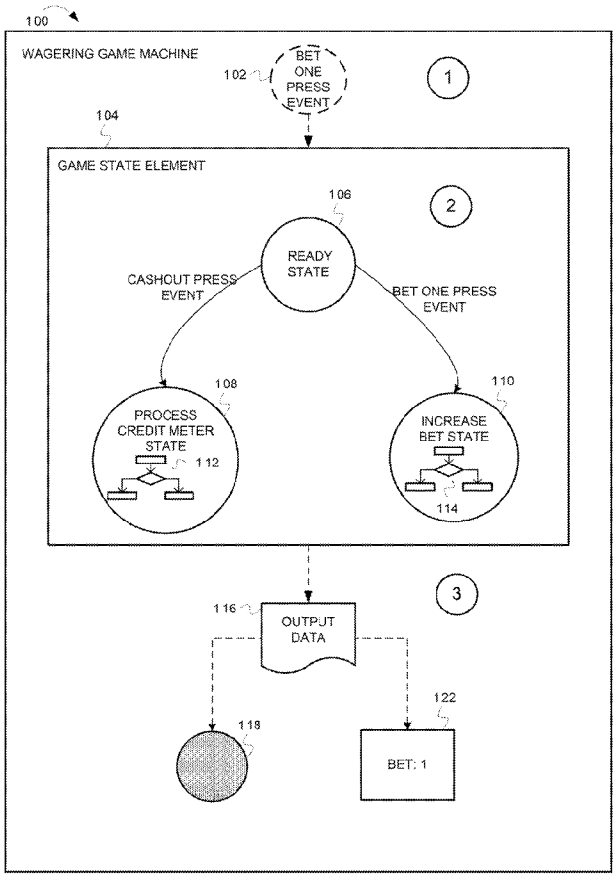
(74) Agents: **DELIZIO, Andrew** et al.; Suite 1000-312, 15201 Mason Road, Cypress, Texas 77433 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, 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, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: PROCESSING WAGERING GAME EVENTS



(57) Abstract: This description describes techniques for processing events in a wagering game machine. In some embodiments, a wagering game machine includes a game controller configured to instantiate a game state element based on game state element generation information and game state types, wherein the game state element is configured to present a wagering game, and wherein the game state element includes states, wherein each state includes behaviors. The wagering game machine can also include an event controller to notify the game state element about events, wherein the events cause the game state element to move between the states and to perform the behaviors.

WO 2008/052207 A2

Published:

— *without international search report and to be republished upon receipt of that report*

PROCESSING WAGERING GAME EVENTS

RELATED APPLICATIONS

[0001] This application claims the priority benefit of U.S. Provisional Application Serial No. 60\863,273 filed Oct 27, 2006.

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 2006, WMS Gaming, Inc.

FIELD

[0003] Embodiments of the inventive subject matter relate generally to wagering game systems and, more particularly, to wagering game systems that record and process events.

BACKGROUND

[0004] Wagering game machines, such as slot machines, video poker machines and the like, have been a cornerstone of the gaming industry for several years. Generally, the popularity of such machines depends on the likelihood (or perceived likelihood) of winning money at the machine and the intrinsic entertainment value of the machine relative to other available gaming options. Where the available gaming options include a number of competing wagering game machines and the expectation of winning at each machine is roughly the same (or believed to be the same), players are likely to be attracted to the most entertaining and exciting machines. Shrewd operators consequently strive to employ the most entertaining and exciting machines, features, and enhancements available because such machines attract frequent play and hence increase profitability to the operator. Therefore, there is a continuing need for wagering game machine manufacturers to continuously develop new games and gaming enhancements that will attract frequent play.

SUMMARY

[0005] A wagering game machine comprising a game controller configured to instantiate a game state element based on game state element generation information and game state types, wherein the game state element is configured to present a wagering game, and wherein the game state element includes states, wherein each state includes behaviors; and an event controller to notify the game state element about events, wherein the events cause the game state element to move between the states and to perform the behaviors.

[0006] In some embodiments, the game state element is associated with a game piece that is used in the wagering game, and wherein the events are player inputs associated with the game piece.

[0007] In some embodiments, the events indicate player inputs associated with the wagering game.

[0008] In some embodiments, some of the behaviors define operations for presenting the wagering game.

[0009] In some embodiments, the game state element generation information and game state types include object-oriented classes, and, in some embodiments, game state element generation information identifies the game state types.

[0010] In some embodiments, the game controller includes an interpreter.

[0011] In some embodiments, the game state element is associated with a game piece in the wagering game.

[0012] A method comprising receiving information indicating a set of game state elements to be used in presenting a wagering game, wherein each of the game state elements defines states and behaviors; creating the game state elements using the information; and presenting a wagering game using the game state elements, wherein presenting the wagering game includes, receiving an event; determining which of game state elements is to be notified of the event; and notifying the game state elements about the event.

[0013] In some embodiments, the information includes a scripting language file, and wherein the creating of the game state elements is performed by interpreting the script.

[0014] In some embodiments, the information includes program code defining object-oriented classes, and wherein the creating the game state elements includes instantiating objects based on the object-oriented classes.

[0015] In some embodiments, the determining is based on information contained in the event.

[0016] In some embodiments, some of the game state elements are associated with game pieces used in the wagering game.

[0017] In some embodiments, the method is further comprising determining, in the game state element, a current state, wherein the determining is based on the event; and performing the behaviors of the game state element.

[0018] In some embodiments, some of the behaviors define operations for presenting media as part of the wagering game.

[0019] A machine-readable medium including instructions that are executable by a machine, the instructions including instructions to detect events, wherein some of the events indicate player input associated with a wagering game, and wherein others of the events indicate machine-generated responses associated with the wagering game; instructions to move between states based on the events, wherein some of the states are associated with a game piece used in the wagering game, and wherein the states define operations for presenting a portion of the wagering game; and instructions to perform the operations for presenting the portion of the wagering game.

[0020] In some embodiments, the instructions are part of an object instantiated from object-oriented classes defining the states and operations.

[0021] In some embodiments, the instructions to perform the operations for presenting the portion of the wagering game include instructions to access media files.

[0022] In some embodiments, the instructions are represented in a scripting language.

[0023] In some embodiments, the instructions are represented in Lua source code.

[0024] In some embodiments, the instructions define objected-oriented classes, and wherein the instructions include source code for a scripting language.

BRIEF DESCRIPTION OF THE FIGURES

[0025] Embodiments of the invention are illustrated in the Figures of the accompanying drawings in which:

[0026] **Figure 1** is a dataflow diagram illustrating dataflow and operations associated with events and states in a wagering game machine, according to example embodiments of the invention;

[0027] **Figure 2** is a block diagram illustrating a wagering game machine architecture, according to example embodiments of the invention;

[0028] **Figure 3** is a block diagram illustrating a wagering game engine, according to example embodiments of the invention;

[0029] **Figure 4** is a block diagram illustrating a game state element, according to example embodiments of the invention;

[0030] **Figure 5** is a block diagram illustrating a wagering game network 500, according to example embodiments of the invention;

[0031] **Figure 6** is a flow diagram illustrating operations for initializing a game engine, according to example embodiments of the invention;

[0032] **Figure 7** is a flow diagram illustrating operations for processing events in a game engine, according to example embodiments of the invention;

[0033] **Figure 8** is a flow diagram illustrating operations for processing events in a game state element, according to example embodiments of the invention;

[0034] **Figure 9** is a block diagram illustrating a game state element including states defined in a sample code segment, according to example embodiments of the invention and

[0035] **Figure 11** is a perspective view of a wagering game machine, according to example embodiments of the invention.

DESCRIPTION OF THE EMBODIMENTS

[0036] This description of the embodiments is divided into six sections. The first section provides an introduction to embodiments of the invention, while the second section describes example wagering game machine architectures. The third section describes example operations performed by some embodiments and the fourth section describes example wagering game machines in more detail. The fifth section includes a code sample. The sixth section presents some general comments.

Introduction

[0037] This section provides an introduction to some embodiments of the invention. Some embodiments include wagering game machines that generate and process events in the course of presenting wagering games. The events can represent player inputs (e.g., button presses), machine-generated responses (e.g., timers expiring, completion of animations, etc.), and other occurrences in a wagering game system. In some embodiments, wagering game machines include logic that defines a discrete set of states relating to the events. When the logic detects events, it can move between states and perform operations associated with the states. For

example, when a player presses a slot machine's "bet one" button, the machine can generate an event representing the button press. The machine can process the event using the states and operations.

[0038] In some embodiments, the logic for processing wagering game events is implemented using a script interpreter and script (i.e., a scripting language file). For example, the logic for determining a game result can be included in a script. To determine the game results, the script interpreter interprets and executes the script. One benefit of implementing event logic using a script is that the script interpreter can execute the script without first compiling and linking the script (i.e., without pre-execution processing). This allows technicians (or the wagering game machine itself) to replace event logic without shutting down the machine to compile and link the new event logic. Another benefit of using script is that script is typically more human-readable than other programming languages, so it can make game development more managable. Figure 1 provides an introduction to some embodiments of the event processing logic.

[0039] **Figure 1** is a dataflow diagram illustrating dataflow and operations associated with events and states in a wagering game machine, according to example embodiments of the invention. Figure 1 shows a wagering game machine 100 that includes a game state element 104 and output devices 118 and 122 (i.e., audio device 118 and display device 122). The game state element 104 includes logic that defines states (i.e., "ready state" 106, "increase bet state" 110, and "process credit meter state" 108) associated with a wagering game. The events (i.e., "cash-out press event" and "bet one press event") cause the game state element 104 to transition between states. When the game state element 104 makes a transition to a new state, it can perform operations associated with the new state (i.e., operations 112 or 114).

[0040] In Figure 1, the dataflow occurs in three stages. Before stage one, the game state element 104 is in the "ready" state 106. During stage one, the game state element 104 is notified of a bet one event 102, which indicates that a player has pressed a "bet one" button. During stage two, the bet one press event 102 causes the game state element 104 to move from the "ready" state 106 to the "increase bet" state 110. After entering the "increase bet" state 110, the game state element 104 performs operations 114, which record the bet. During stage three, as a result of the operations 114, the game state element 104 transmits output data 116 to one or more output devices 118 & 122. The operations 114 can cause a bet meter 122 to indicate that a player has bet one credit. The operations 112 can perform operations for zeroing-out a credit meter and presenting any associated graphics and sounds.

[0041] Although Figure 1 describes some embodiments, the following sections describe many other features and embodiments.

Wagering Game Machine Architectures

[0042] This section presents Figures 2-5, which describe example architectures according to embodiments of the invention. This section continues with a discussion of Figure 2.

[0043] **Figure 2** is a block diagram illustrating a wagering game machine architecture, according to example embodiments of the invention. As shown in Figure 2, the wagering game machine architecture 200 includes a wagering game machine 206, which includes a central processing unit (CPU) 226 connected to main memory 228. The CPU 226 can include any suitable processor, such as an Intel® Pentium processor, Intel® Core 2 Duo processor, AMD Opteron™ processor, UltraSPARC processor, etc. The main memory 228 includes a wagering game engine 236. In some embodiments, the wagering game engine 236 includes components (e.g., game state elements) that represent game pieces and game logic. The components can include discreet sets of states, and events can prompt transitions between the states. In some embodiments, the wagering game engine 236 can present wagering games, such as video poker, video black jack, video slots, video lottery, etc., in whole or part.

[0044] The CPU 226 is connected to an input/output (I/O) bus 222, which can include any suitable bus technologies, such as an AGTL+ frontside bus and a PCI backside bus. The I/O bus 222 is connected to a payout mechanism 208, primary display 210, secondary display 212, value input device 214, player input device 216, information reader 218, and storage unit 230. The player input device 216 can include the value input device 214 to the extent the player input device 216 is used to place wagers. The I/O bus 222 is also connected to an external system interface 224, which is connected to external systems 204 (e.g., wagering game networks).

[0045] In one embodiment, the wagering game machine 206 can include additional peripheral devices and/or more than one of each component shown in Figure 2. For example, in one embodiment, the wagering game machine 206 can include multiple external system interfaces 224 and/or multiple CPUs 226. In one embodiment, any of the components can be integrated or subdivided. Furthermore, in some embodiments, components shown inside the main memory 228 can be moved outside the main memory 228 (e.g., the components can be included in controllers, chips, or other devices in the wagering game machine 206).

[0046] Any component of the architecture 200 can include hardware, firmware, and/or machine-readable media including instructions for performing the operations described herein.

Machine-readable media includes any mechanism that provides (i.e., stores and/or transmits) information in a form readable by a machine (e.g., a wagering game machine, computer, etc.). For example, tangible machine-readable media includes read only memory (ROM), random access memory (RAM), magnetic disk storage media, optical storage media, flash memory machines, etc. Machine-readable media also includes any media suitable for transmitting software over a network.

[0047] This section continues with a more detailed description of embodiments of a wagering game engine.

[0048] **Figure 3** is a block diagram illustrating a wagering game engine, according to example embodiments of the invention. In Figure 3, the game engine 300 includes a game controller 312, which includes an event controller 320. The game controller 312 is connected to a plurality of game state elements 314, presentation manager 308, resources 306, event queue 304, and configuration information and logic 318. The resources 306 are connected to the presentation manager 308.

[0049] The configuration information and logic 318 includes game state element generation information 322 and a plurality of game state types 302. Each game state type 302 includes a plurality of state identifiers, events, and behaviors 316. The state identifiers can identify states associated with game state elements of a wagering game, while the events can identify occurrences that elicit transitions between the states. The behaviors can identify operations to perform after entering a state. In some embodiments, the behaviors can indicate that no operations are performed when a state is entered. States, events, and behaviors will be described in more detail below (see discussion of Figure 4).

[0050] In some embodiments, the game state types 302 and game state element generation information 322 are object-oriented classes. In some embodiments, the game state element generation information 322 is an object-oriented class that uses classes defined in the game state types 302. For example, the game state element generation information's game state type identifier can indicate a game state type 302 that includes a class which can be used in creating one of the game state elements 314. The classes can include source code from any suitable object-oriented programming language, including high-level languages (e.g., Java, C++, etc.), scripting languages (e.g., Lua, Python, etc.), etc.

[0051] The game controller 312 can use the game state element generation information 322 and game state types 302 to create the game state elements 314. In some embodiments, the game controller 312 includes an interpreter that creates the game state elements 314 by instantiating

objects, where the objects are defined by classes included in the game state types 302 and game state element generation information 322. In other embodiments, the game controller 312 includes executable program code that creates the game state elements 314 based on information in the game state types 302 and game state element generation information 322. After creating the game state elements 314, the game engine 300 can present a wagering game by processing the events 310.

[0052] The event controller 320 can store events 310 in the event queue 304. The events 310 can represent player inputs (e.g., button presses), machine-generated responses (e.g., timers expiring, completion of animations, etc.), and other occurrences in a wagering game system. In some embodiments, the events 310 can include an event identifier, event data, and/or a destination game state element. The event controller 320 can pass the events 310 to the game state elements 314. In some embodiments, the event controller 320 passes all the events to all the game state elements 314. In other embodiments, the event controller 320 forwards only certain events to certain game state elements 314. That is, the event controller 320 can act as an event filter. For example, a first game state element 314 may process button-related events, while a second game state element 314 may process events related to receipt of coins. The game controller 312 can forward button events to the first game state element 314, while passing coin-related events to the second game state element 314.

[0053] The resources 306 can include text, audio content, video content, animation content, and/or any other information useful in presenting a wagering game. The resources 306 are accessible to the game sate elements 314. For example, a game state element's behaviors can define operations for accessing and presenting audio content stored in the resources 306. Similarly, a game state element may access and present audio content stored the resources 306. The presentation manager 308 can assist the game state elements 314 in presenting media. In some embodiments, game designers can change a wagering game's look and feel by changing content in the resources 306.

[0054] As noted above, game state elements include logic that defines states, events, and behaviors. Figure 4 describes game state elements in more detail.

[0055] **Figure 4** is a block diagram illustrating a game state element, according to example embodiments of the invention. As shown in Figure 4, the game state element 402 includes state 1, state 2, and state 3. Also, the game state element 402 includes event A and event B.

[0056] At any given time, the game state element 402 is in one of the states 1, 2, or 3. In some embodiments, the game state element 402 can include different states and events, where it could

be in more than one state at any given time. Events cause the game state element 402 to move between states. For example, if the game state element 402 were in state 1, it would move to state 2 after detecting event B. Upon entering state 2, the game state element 402 can perform the behaviors 406. In some embodiments, the behaviors 406 represent operations for presenting a portion of a wagering game, such as operations for presenting video content on a display device, determining game results, etc. Although not shown in Figure 4, the game state element 402 includes logic (e.g., variables, tables, registers, program code, circuits, etc.) that tracks current states and performs behaviors.

[0057] Game state elements can be associated with game pieces or other aspects of game control. For example, a game engine can present a card game using a game state element for each card used in the card game and game state elements for controlling the cards, betting, and other aspects of the card game. The card-related game state elements could be notified of events that represent player inputs affecting the cards. The card-related game state elements can perform behaviors that respond to the events. For example, the card-related game state elements can respond to player input by presenting graphics indicating that a card is face-up or face-down.

[0058] To illustrate further, the card-related game state elements can be similar to those shown in Figure 4. Referring to Figure 4, state 1 can indicate that the playing card is available for selection from a deck, state 2 can indicate that the playing card was drawn and held face-up, and state 3 can indicate that the playing card was drawn and held face-down. Event A can be associated with player inputs, such as touchscreen presses indicating the card has been drawn and held face-down. Event B can be associated with player inputs, such as touchscreen presses indicating that the playing card has been drawn and held face-up. The game state element behaviors 404 & 406 can include operations that graphically show the playing card face-up and face-down.

[0059] While Figures 2-4 describe wagering game machine components, Figure 5 describes a wagering game network.

[0060] **Figure 5** is a block diagram illustrating a wagering game network 500, according to example embodiments of the invention. As shown in Figure 5, the wagering game network 500 includes a plurality of casinos 512 connected to a communications network 514.

[0061] Each of the plurality of casinos 512 includes a local area network 516, which may include a wireless access point 504, wagering game machines 502, and a wagering game server 506 that can serve wagering games over the local area network 516. The local area network 516 includes wireless communication links 510 and wired communication links 508. The wired and

wireless communication links can employ any suitable connection technology, such as Bluetooth, 802.11, Ethernet, public switched telephone networks, SONET, etc. In one embodiment, the wagering game server 506 can serve wagering games and/or distribute content to devices located in other casinos 512 or at other locations on the communications network 514.

[0062] Although not shown, the wagering game machines 502 can include game engines, as described above. The wagering game machines described herein can take any suitable form, such as floor standing models, handheld mobile units, bartop models, workstation-type console models, etc. Furthermore, the wagering game machines 502 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.

[0063] Any component of the gaming network 500 (e.g., the wagering game machines 502 and wagering game server 506) can include hardware and machine-readable media including instructions for performing the operations described herein. In one embodiment, the wagering game network 500 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.

[0064] In various embodiments, wagering game machines 502 and wagering game servers 506 work together such that a wagering game machine 502 may be operated as a thin, thick, or intermediate client. For example, one or more aspects of game play may be controlled by the wagering game machine 502 (client) or the wagering game server 506 (server). That is, in some embodiments, game engines can reside in the wagering game machines 502 and/or the game server 506. In a thin-client example, the wagering game server 506 can perform functions such as determining game outcome or managing assets, while the wagering game machine 502 can present the graphical representation of such outcome to a user (e.g., player). In a thick-client example, game outcome may be determined locally (e.g., by a game engine in a wagering game machine 502) and then communicated to the wagering game server 506 for recording or managing a player's account.

[0065] Similarly, functionality that is not directly related to game play may be controlled by the wagering game machine 502 (client) or the wagering game server 506 (server). For example, power conservation controls that manage a display screen's light intensity may be managed centrally (e.g., by the wagering game server 506) or locally (e.g., by the wagering game machine 502). Other functionality not directly related to game play may include presentation of advertising, software or firmware updates, system quality or security checks, etc.

Operations

[0066] This section describes operations performed by embodiments of the invention. In the discussion below, the flow diagrams will be described with reference to the block diagrams presented above. In certain embodiments, the operations are performed by executing instructions residing on machine-readable media (e.g., software), while in other embodiments, the operations are performed by hardware and/or other logic (e.g., firmware). In some embodiments, the operations are performed in series, while in other embodiments, one or more of the operations can be performed in parallel. This section begins with a discussion of operations for initializing a game engine.

[0067] **Figure 6** is a flow diagram illustrating operations for initializing a game engine, according to example embodiments of the invention. The flow 600 begins at block 602.

[0068] At block 602, the game controller 312 is notified of information indicating a set of game state elements to be used in presenting a wagering game. In some embodiments, the information is represented as interpretable source code that includes object-oriented class definitions defining a set of game state elements. The game state types 302 and game state element generation information 322 can include the object-oriented class definitions. In other embodiments, the information is represented as binary data that includes information from the game state types 302 and game state element generation information 322. The flow continues at block 604.

[0069] At block 604, the game controller 312 determines resources for each of the game state elements. In some embodiments, the information at block 602 indicates which of the resources 306 are associated with each game state element. The resources 306 can include media such as audio content, video content, animations, text, and/or any other information needed by a game state element. The flow continues at block 606.

[0070] At block 606, the game controller 312 creates the game state elements 314. In some embodiments, the game controller 312 creates the game elements 314 by interpreting source code (received at block 602) that includes class definitions defining a set of game state elements and instantiating the game state elements 314. In other embodiments, the game controller 312 creates the game state elements 314 using binary data (received at block 602) that defines the game state elements' states, events, and behaviors. The flow continues at block 608.

[0071] At block 608, if needed, the game controller 312 presents media associated with one or more of the game state elements 314. For example, some of the game state elements 314 can be

associated with game pieces, such as playing cards, selectable game elements, etc. Thus, the game controller 312 can present graphics, sounds, and/or other media to reveal the game pieces. Some of the game state elements 314 may not be associated with media. The flow continues at block 610.

[0072] At block 610, the game engine 300 presents a wagering game using the game state elements 314. For example, after revealing the game pieces, the game state elements 314 process events and perform operations that present a wagering game. Operations performed by some embodiments of a game state element will be described in more detail below (see discussion of Figure 8).

[0073] This section continues by discussing operations for processing events.

[0074] **Figure 7** is a flow diagram illustrating operations for processing events in a game engine, according to example embodiments of the invention. The flow 700 begins at block 702.

[0075] At block 702, the event controller 320 detects an event 310. As noted above, events can indicate user input, machine-generated results, and other occurrences in a wagering game machine and/or wagering game network. The flow continues at block 704.

[0076] At block 704, the event controller 320 determines which of the game state elements 314 are to be notified of the event. In some embodiments, the event controller 320 forwards the event to all the game elements 314. In other embodiments, the event controller 314 determines what events about which it will notify the different game state elements 314. The flow continues at block 706.

[0077] At block 706, the event controller 320 notifies the game state element(s) 314 about the event. From block 706, the flow ends.

[0078] While Figure 7 describes operations for notifying game state elements about events received in a game engine, this section continues by describing how events affect game states and elicit various behaviors.

[0079] **Figure 8** is a flow diagram illustrating operations for processing events in a game state element, according to example embodiments of the invention. The flow 800 begins at block 802.

[0080] At block 802, a game state element is notified of an event associated with the wagering game. For example, referring to Figure 4, the game state element 402 is notified of event A from an event controller. The flow continues at block 804.

[0081] At block 804, the game state element determines a state, based on the event. For example, referring to Figure 4, if the game state element 402 were in state 1 when it is notified of event A (at block 802), it would move to state 3. The flow continues at block 806.

[0082] At block 806, the game state element determines whether the event triggers behaviors for the current state. For example, the game state element 402 determines whether state 3 includes behaviors. If there are behaviors for the current state, the flow continues at block 808. Otherwise, the flow continues at block 810.

[0083] At block 808, the game state element performs the behaviors. For example, upon entering state 3, the game state element 402 performs the behaviors 404. In some embodiments, the behaviors include operations for presenting part of wagering game. For example, the behaviors 404 can include operations for presenting graphics that represent actions of a playing card, such as turning the card face up, discarding the card, etc. Additionally, the behaviors can include other operations related to wagering games, such as operations for recording wagering game results, recording gaming session statistics, etc. The flow continues at block 810.

[0084] At block 810, the game state element determines whether there will be more events. For example, the game state element 402 determines whether it has reached a terminal state. If there will be no more events, the flow ends. Otherwise, the flow continues at block 802.

[0085] This section will conclude with a discussion about how game engines can process replacement game state types and game state element generation information. As noted above, the game state types 302 and game state element generation information 322 can include source code (e.g., a script) defining object-oriented classes, where the classes can be used to create the game state elements 312. As also noted above, the game controller 312 can include an interpreter (e.g., a scripting language interpreter) that instantiates the game state elements 314 based on the classes in the source code. In embodiments where the game controller 312 includes an interpreter, technicians can replace game state types 302 while the game engine is running. When the replacement code is needed, the game controller's interpreter can interpret the replacement code at runtime. Therefore, these embodiments can avoid shutting-down the game engine to recompile and relink the source code.

[0086] In some embodiments, after presenting a wagering game, a game engine can be reconfigured to present a different wagering game. For example, the game controller 312 can load new state element generation information 322 that defines game state elements 314 for a different wagering game. In some embodiments, technicians (or system processes) can change a wagering game's look and feel by changing associations to resources in the game state element generation information 322. For example, technicians can change the game state element generation information's associations to resources to include different animation files. As a

result, because animation files have been changed, the wagering games' game pieces will look different.

Sample Game State Types

[0087] This section shows some example game state types. The following code segment serves as an example of how some embodiments can represent game state types in program code.

```
module (..., package.seeall)

require "Column"
require "PillarOrbAnimation"
require "PillarKeyAnimation"
require "PillarAttract"

-- Bonus game states ...
local ANIM_PREBONUS                    = "ANIM-PREBONUS"
local PILLAR_BONUS                     = "PILLAR-BONUS"
local REVEAL_ALL                       = "REVEAL_ALL"
local SHOW_POOPER                      = "SHOW_POOPER"

local function PillarGameConstructor(self)
        StateMachine.CStateMachine.init(self, "obPillarGame", ANIM_PREBONUS);

        self.strPillarStage = "PillarStage"

        self.strPillarBackground = "PillarBG"
        self.strCrownText = "CrownText"

        self.strCreditsMeterText = "CreditsMeterText"
        self.strTotalBetMeterText = "TotalBetMeterText"
        self.strBonusWonMeterText = "BonusWonMeterText"

        self.strCreditsMeter = "CreditMeter"
        self.strTotalBetMeter = "TotalBetMeter"
        self.strBonusWonMeter = "BonusMeter"

        self.lstColumns = {}

        self.obStateFunctions[ANIM_PREBONUS] = nil
        self.obStateFunctions[PILLAR_BONUS] = nil
        self.obStateFunctions[REVEAL_ALL] = nil
end

CPillarGame = class.class(StateMachine.CStateMachine, PillarGameConstructor);

function CPillarGame:Start()
        obStartAnimation:Perform(self.ShowPillarBonus, self)
        self:SetState(ANIM_PREBONUS)
end

function CPillarGame:CreatePillarBackground()
        CreateImage(self.strPillarStage,
                                self.strPillarBackground, "Bonus1_BG", 0, 0,
0);
        ObjectCommand(self.strPillarStage, self.strPillarBackground, "Show");
```

```
end

function CPillarGame:CreateColumns()
	for i = 1, 28 do
		self.lstColumns[i] = Column.CColumn(i, self.strPillarStage);
		self.lstColumns[i]:Show(true)
	end
end

function CPillarGame:CreateCrownText()
	CreateImage(self.strPillarStage,
					self.strCrownText, "Orbs_Awarded", 356,
101, 10);
	ObjectCommand(self.strPillarStage, self.strCrownText, "Show");
end

function CPillarGame:CreateMeters()
	CreateImage(self.strPillarStage,
					self.strCreditsMeterText,
"METER_B1_Credits", 73, 535, 110);
	CreateImage(self.strPillarStage,
				self.strTotalBetMeterText, "METER_B1_TotalBet",
335, 535, 110);
	CreateImage(self.strPillarStage,
				self.strBonusWonMeterText, "METER_B1_BonusWon",
579, 535, 110);

	CreateMeter(self.strPillarStage, self.strCreditsMeter);
	CreateMeter(self.strPillarStage, self.strTotalBetMeter);
	CreateMeter(self.strPillarStage, self.strBonusWonMeter);

	local nTotalBet = GetTotalBet();
	local nCredits = GetCredits();

	ObjectCommand(self.strPillarStage,
					self.strCreditsMeter, "SetValue
" .. nCredits);
	ObjectCommand(self.strPillarStage,
					self.strTotalBetMeter, "SetValue
" .. nTotalBet);
	ObjectCommand(self.strPillarStage, self.strBonusWonMeter, "SetValue
0");

	ObjectCommand(self.strPillarStage, self.strCreditsMeterText, "Show");
	ObjectCommand(self.strPillarStage, self.strTotalBetMeterText, "Show");
	ObjectCommand(self.strPillarStage, self.strBonusWonMeterText, "Show");

	ObjectCommand(self.strPillarStage, self.strCreditsMeter, "Show");
	ObjectCommand(self.strPillarStage, self.strTotalBetMeter, "Show");
	ObjectCommand(self.strPillarStage, self.strBonusWonMeter, "Show");
end

function CPillarGame:Initialize()
	CreateStage(self.strPillarStage, 500);
	self:CreatePillarBackground()
	self:CreateColumns()
	self:CreateCrownText()
	self:CreateMeters()
	Sparks.CreateSparks(self.strPillarStage)
	PillarOrbAnimation.CPillarOrbAnimation(self.strPillarStage)
	PillarKeyAnimation.CPillarKeyAnimation(self.strPillarStage)
```

```
        PillarAttract.CPillarAttract(self.strPillarStage)
end

function CPillarGame:ShowPillarBonus()
        ShowStage(self.strPillarStage);
        self:SetState(PILLAR_BONUS);
end

function CPillarGame:BangCreditMeter()
        ObjectCommand(self.strPillarStage,
                        self.strBonusWonMeter, "Bang " ..
BonusMath.nCreditsSelected);
end

function CPillarGame:DisableColumns(obExcept)
        for i = 1,28 do
                if self.lstColumns[i].strName ~= obExcept.strName then
                        self.lstColumns[i]:Disable();
                end
        end
end

function CPillarGame:AttractColumn(nIndex)
        if self.lstColumns[nIndex] then
                self.lstColumns[nIndex]:Attract();
        end
end

function CPillarGame:RevealAllColumns()
        for i = 1,28 do
                self.lstColumns[i]:Reveal();
        end
end

function CPillarGame:StartPooperAnimation()
        self:RevealAllColumns()
        self:SetState(REVEAL_ALL)
        Delay(3000);
        self:SetState(SHOW_POOPER)
        obEndAnimation:Perform(self.HideAll, self)
end

function CPillarGame:HideAll()
        HideStage(self.strPillarStage);
end
```

[0088] The game state types in the code sample define game state elements that control a bonus game. The game state elements that control the bonus game can be used with other game state elements, such as game state elements associated with game pieces and game controls. **Figure 9** is a block diagram illustrating a game state element including states defined in a sample code segment, according to example embodiments of the invention. In Figure 9, the game state element 902 includes states defined in the sample code segment shown above. In particular, the game state element 902 includes an ANIM_PREBONUS state 904, PILAR_BONUS state 906, REVEAL_ALL state 908, and a SHOW_POOPER state 910.

More about Game Engines and Scripts

[0089] This section describes additional details about game engines and scripts. As noted above, the game engine can include a script interpreter and script (i.e., a scripting language file). **Figure 10** is a block diagram illustrating a wagering game machine including a script interpreter and script, according to some embodiments of the invention. The wagering game 1006 includes the same components as the wagering game machine 206 of Figure 2. However, in Figure 10, the wagering game machine's main memory 1028 includes an operating system 1036, middleware 1034, script interpreter 1032, and script 1038. Furthermore, in Figure 10, the wagering game machine's storage unit 1030 includes a script library 1038 and media files 1040.

[0090] In the main memory 1028, the operating system 1036 can be any operating system suitable for a wagering game machine, such as adaptations of Linux and Windows. The middleware 1034 provides a layer of abstraction between the operating system 1036 and the script interpreter 1032, script 1038, and other application programs (not shown). That is, the script interpreter 1032 and script 1038 request services from the middleware 1034 that they would typically request from an operating system. In turn, the middleware 1034 provides those services. Because the script 1038, script interpreter 1032, and other application programs are designed to request services from the middleware, they can operate with any operating system compatible with the middleware 1034. For example, if the middleware is compatible with Linux, Windows, and Solaris, the script 1038 and script interpreter 1032 can present wagering games when the operating system 1036 is Linux, Windows, or Solaris.

[0091] The script interpreter 1032 can include any suitable scripting language interpreter, such as a Lua interpreter, Python interpreter, etc. In some embodiments, the script interpreter 1032 is an embodiment of the game controller 312 of Figure 3. The script 1038 can include one or more files including scripting language code (e.g., text), such as Lua code, Python code, etc. The script 1038 can define and instantiate game state elements (see 314 in Figure 3) and a presentation manager (see 308). Thus, after the script interpreter 1032 interprets and executes a portion of the script 1038, the script 1038 represents the components used in presenting a wagering game (see Figure 3).

[0092] As noted above, the storage unit 1030 includes a script library 1038. The script library 1038 can include portions of the script 1038 that are not needed in main memory 1028. When contents of the script library 1038 are needed in main memory 1028, the script interpreter 1032 (with the assistance of the middleware 1034 and operating system 1036) can load them into main

memory 1028 as part of the script 1038. Furthermore, the script library 1038 can include configuration information and logic (see 318 in Figure 3).

[0093] During operation, the wagering game machine 1006 processes events and presents wagering games. For example, the operating system 1036 can detect player input, such as input from the player input device 1016. The operating system 1036 can provide a record of the input to the middleware 1034. In turn, the middleware 1034 provides the input to the script 1038, which is being interpreted and executed by the script interpreter 1032. The script 1038 processes the input as an "event" (as described above). The script 1038 processes the event by providing the event to a game state element (a portion of the script 1038) suited for processing the event. For example, if the input indicates that a player pressed a "spin reels" button, the script 1038 provides the event to the game state element capable of determining a result for a slots game. Next, the game state element can determine a result, which can constitute yet another event, which the script 1038 will process. The script 1038 can define data structures that store multiple events for later processing (see discussion of event queue 304). Eventually, the script 1038 will utilize the media files 1040 to graphically audibly present the result to the player.

[0094] As discussed above, one or more portions of the script 1038 can be replaced without processing other portions of the script 1038. For example, if technicians want to replace a portion of the script 1038 (e.g., a game state element) that determines results for a slots game, they can replace it without affecting the wagering game machine's ability to present games. After the script portion is replaced, the wagering game machine 1006 can present wagering games without recompiling and relinking the script 1038.

Wagering Game Machines

[0095] This section describes additional details of wagering game machines in which embodiments of the invention can be practiced.

[0096] **Figure 11** is a perspective view of a wagering game machine, according to example embodiments of the invention. Referring to Figure 11, a wagering game machine 1100 is used in gaming establishments, such as casinos. According to embodiments, the wagering game machine 1100 can be any type of wagering game machine and can have varying structures and methods of operation. For example, the wagering game machine 1100 can be an electromechanical wagering game machine configured to play mechanical slots, or it can be an electronic wagering game machine configured to play video casino games, such as blackjack, slots, keno, poker, blackjack, roulette, etc.

[0097] The wagering game machine 1100 comprises a housing 1112 and includes input devices, including value input devices 1118 and a player input device 1124. For output, the wagering game machine 1100 includes a primary display 1114 for displaying information about a basic wagering game. The primary display 1114 can also display information about a bonus wagering game and a progressive wagering game. The wagering game machine 1100 also includes a secondary display 1116 for displaying wagering game events, wagering game outcomes, and/or signage information. While some components of the wagering game machine 1100 are described herein, numerous other elements can exist and can be used in any number or combination to create varying forms of the wagering game machine 1100.

[0098] The value input devices 1118 can take any suitable form and can be located on the front of the housing 1112. The value input devices 1118 can receive currency and/or credits inserted by a player. The value input devices 1118 can include coin acceptors for receiving coin currency and bill acceptors for receiving paper currency. Furthermore, the value input devices 1118 can include ticket readers or barcode scanners for reading information stored on vouchers, cards, or other tangible portable storage devices. The vouchers or cards can authorize access to central accounts, which can transfer money to the wagering game machine 1100.

[0099] The player input device 1124 comprises a plurality of push buttons on a button panel 1126 for operating the wagering game machine 1100. In addition, or alternatively, the player input device 1124 can comprise a touch screen 1128 mounted over the primary display 1114 and/or secondary display 1116.

[00100] The various components of the wagering game machine 1100 can be connected directly to, or contained within, the housing 1112. Alternatively, some of the wagering game machine's components can be located outside of the housing 1112, while being communicatively coupled with the wagering game machine 1100 using any suitable wired or wireless communication technology.

[00101] The operation of the basic wagering game can be displayed to the player on the primary display 1114. The primary display 1114 can also display a bonus game associated with the basic wagering game. The primary display 1114 can include a cathode ray tube (CRT), a high resolution liquid crystal display (LCD), a plasma display, light emitting diodes (LEDs), or any other type of display suitable for use in the wagering game machine 1100. Alternatively, the primary display 1114 can include a number of mechanical reels to display the outcome. In Figure 11, the wagering game machine 1100 is an "upright" version in which the primary display 1114 is oriented vertically relative to the player. Alternatively, the wagering game machine can be a

"slant-top" version in which the primary display 1114 is slanted at about a thirty-degree angle toward the player of the wagering game machine 1100. In yet another embodiment, the wagering game machine 1100 can exhibit any suitable form factor, such as a free standing model, bartop model, mobile handheld model, or workstation console model.

[00102] A player begins playing a basic wagering game by making a wager via the value input device 1118. The player can initiate play by using the player input device's buttons or touch screen 1128. The basic game can include arranging a plurality of symbols along a payline 1132, which indicates one or more outcomes of the basic game. Such outcomes can be randomly selected in response to player input. At least one of the outcomes, which can include any variation or combination of symbols, can trigger a bonus game.

[00103] In some embodiments, the wagering game machine 1100 can also include an information reader 1152, which can include a card reader, ticket reader, bar code scanner, RFID transceiver, or computer readable storage medium interface. In some embodiments, the information reader 1152 can be used to award complimentary services, restore game assets, track player habits, etc.

General

[00104] In the following detailed description, reference is made to specific examples by way of drawings and illustrations. These examples are described in sufficient detail to enable those skilled in the art to practice the inventive subject matter, and 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 or limitations 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. The following detailed description does not, therefore, limit embodiments of the invention, which are defined only by the appended claims.

[00105] 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 wagering game machine comprising:

a game controller configured to instantiate a game state element based on game state element generation information and game state types, wherein the game state element is configured to present a wagering game, and wherein the game state element includes states, wherein each state includes behaviors; and

an event controller to notify the game state element about events, wherein the events cause the game state element to move between the states and to perform the behaviors.

2. The wagering game machine of claim 1, wherein the game state element is associated with a game piece that is used in the wagering game, and wherein the events are player inputs associated with the game piece.

3. The wagering game machine of claim 1, wherein the events indicate player inputs associated with the wagering game.

4. The wagering game machine of claim 1, wherein some of the behaviors define operations for presenting the wagering game.

5. The wagering game machine of claim 1, wherein the game state element generation information and game state types include object-oriented classes, and wherein game state element generation information identifies the game state types.

6. The wagering game machine of claim 1, wherein the game controller includes an interpreter.

7. The wagering game machine of claim 1, wherein the game state element is associated with a game piece in the wagering game.

8. A method comprising:

receiving information indicating a set of game state elements to be used in presenting a wagering game, wherein each of the game state elements defines states and behaviors;

creating the game state elements using the information; and

presenting a wagering game using the game state elements, wherein presenting the wagering game includes,

receiving an event;

determining which of game state elements is to be notified of the event; and

notifying the game state elements about the event.

9. The method of claim 8, wherein the information includes a scripting language file, and wherein the creating of the game state elements is performed by interpreting the script.

10. The method of claim 8, wherein the information includes program code defining object-oriented classes, and wherein the creating the game state elements includes instantiating objects based on the object-oriented classes.

11. The method of claim 8, wherein the determining is based on information contained in the event.

12. The method of claim 8, wherein some of the game state elements are associated with game pieces used in the wagering game.

13. The method of claim 8 further comprising:

determining, in the game state element, a current state, wherein the determining is based on the event; and

performing the behaviors of the game state element.

14. The method of claim 8, wherein some of the behaviors define operations for presenting media as part of the wagering game.

15. A machine-readable medium including instructions that are executable by a machine, the instructions including:

instructions to detect events, wherein some of the events indicate player input associated with a wagering game, and wherein others of the events indicate machine-generated responses associated with the wagering game;

instructions to move between states based on the events, wherein some of the states are associated with a game piece used in the wagering game, and wherein the states define operations for presenting a portion of the wagering game; and

instructions to perform the operations for presenting the portion of the wagering game.

16. The machine-readable medium of claim 15, wherein the instructions are part of an object instantiated from object-oriented classes defining the states and operations.

17. The machine-readable medium of claim 15, wherein the instructions to perform the operations for presenting the portion of the wagering game include instructions to access media files.

18. The machine-readable medium of claim 15, wherein the instructions are represented in a scripting language.

19. The machine-readable medium of claim 15, wherein the instructions are represented in Lua source code.

20. The machine-readable medium of claim 15, wherein the instructions define objected-oriented classes, and wherein the instructions include source code for a scripting language.

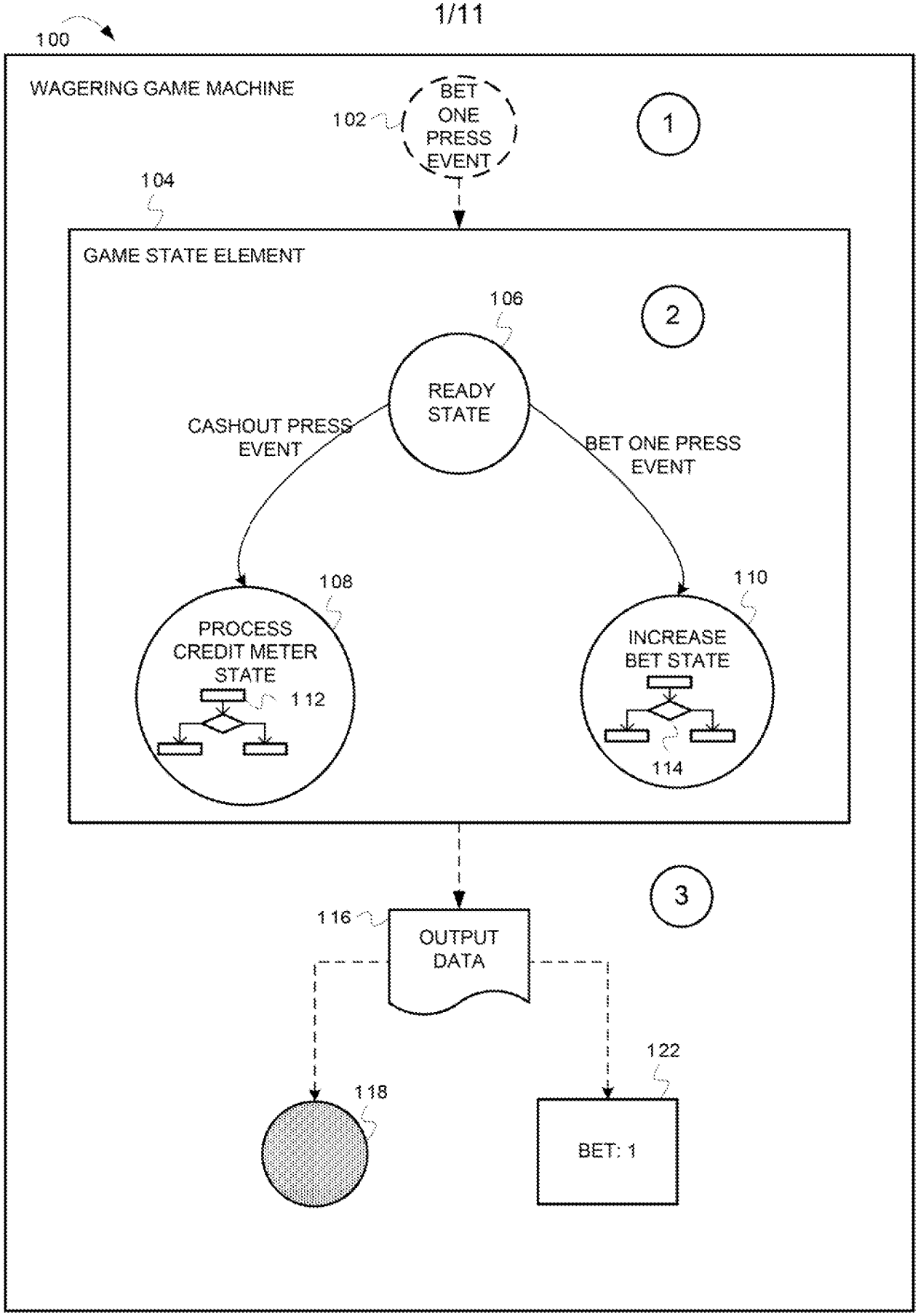

100
WAGERING GAME MACHINE
BET ONE PRESS EVENT
102
1
104
GAME STATE ELEMENT
106
2
READY STATE
CASHOUT PRESS EVENT
BET ONE PRESS EVENT
108
PROCESS CREDIT METER STATE
112
110
INCREASE BET STATE
114
3
116
OUTPUT DATA
118
122
BET: 1

FIG. 1

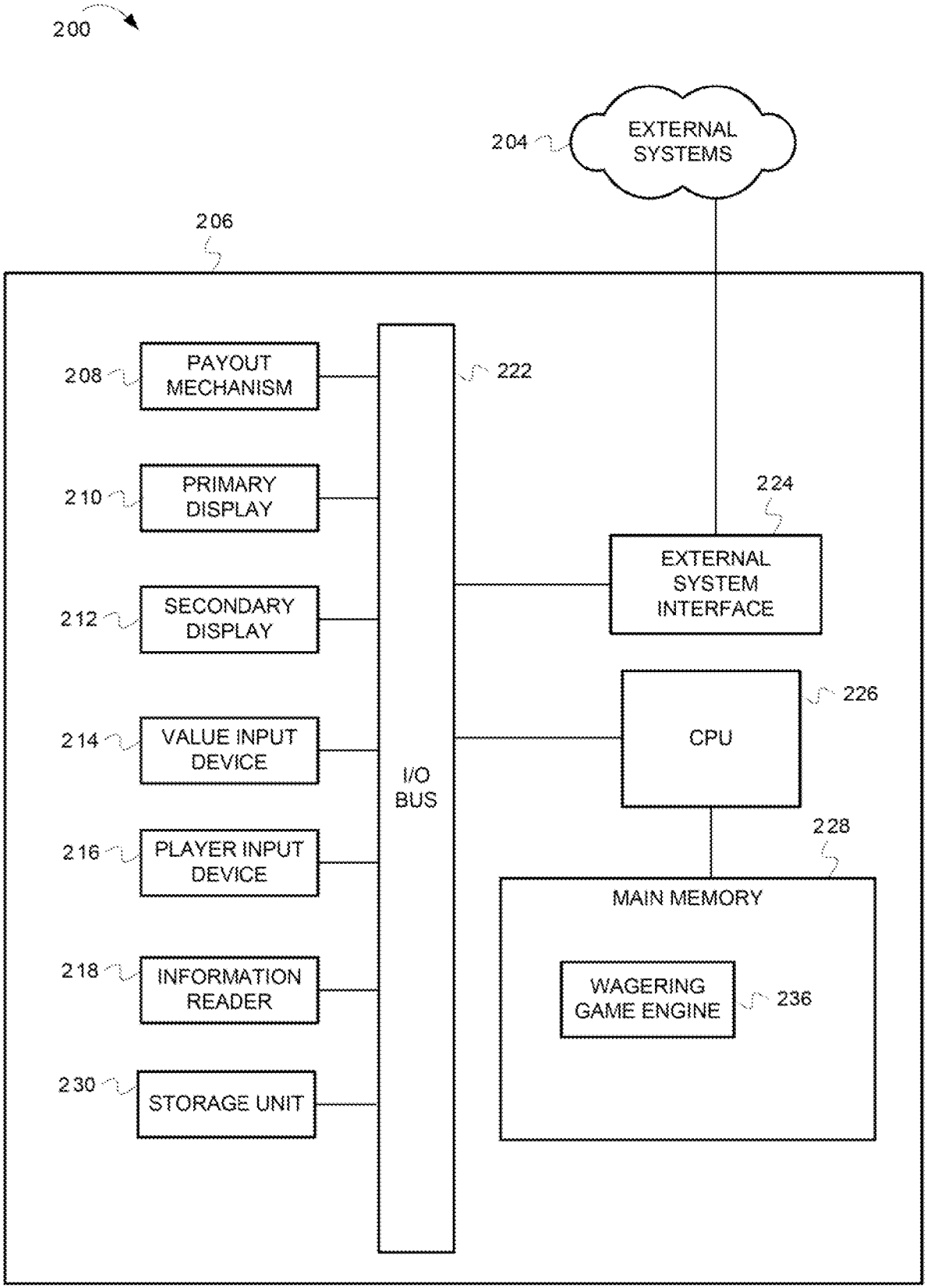

200
204
EXTERNAL SYSTEMS
206
208
PAYOUT MECHANISM
210
PRIMARY DISPLAY
212
SECONDARY DISPLAY
214
VALUE INPUT DEVICE
216
PLAYER INPUT DEVICE
218
INFORMATION READER
230
STORAGE UNIT
222
I/O BUS
224
EXTERNAL SYSTEM INTERFACE
226
CPU
228
MAIN MEMORY
WAGERING GAME ENGINE
236

FIG. 2

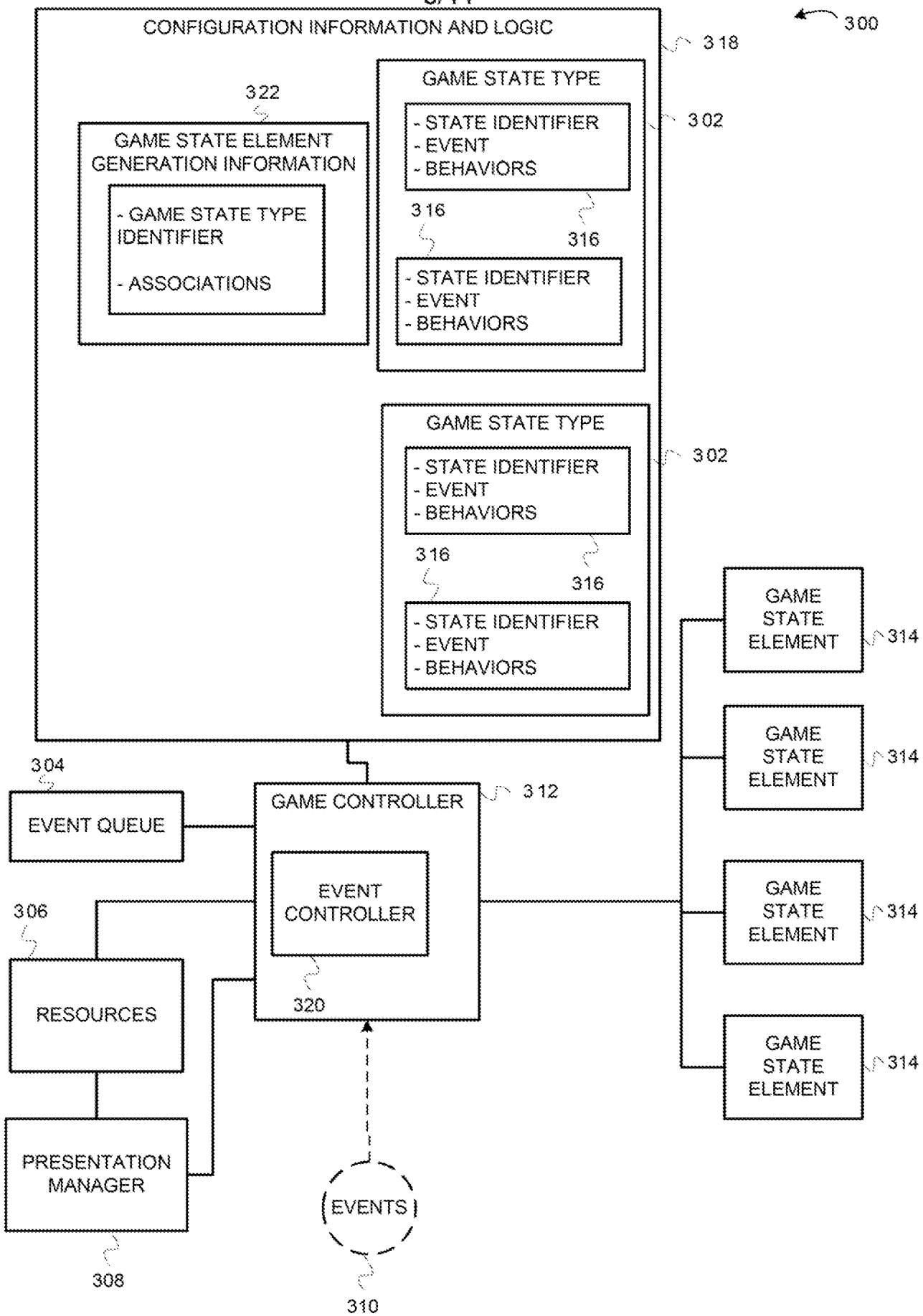

CONFIGURATION INFORMATION AND LOGIC
318
300
GAME STATE TYPE
- STATE IDENTIFIER
- EVENT
- BEHAVIORS
302
322
GAME STATE ELEMENT GENERATION INFORMATION
- GAME STATE TYPE IDENTIFIER
- ASSOCIATIONS
316
316
- STATE IDENTIFIER
- EVENT
- BEHAVIORS
GAME STATE TYPE
- STATE IDENTIFIER
- EVENT
- BEHAVIORS
302
316
316
- STATE IDENTIFIER
- EVENT
- BEHAVIORS
GAME STATE ELEMENT
314
GAME STATE ELEMENT
314
GAME STATE ELEMENT
314
GAME STATE ELEMENT
314
304
EVENT QUEUE
GAME CONTROLLER
312
EVENT CONTROLLER
320
306
RESOURCES
PRESENTATION MANAGER
308
EVENTS
310

FIG. 3

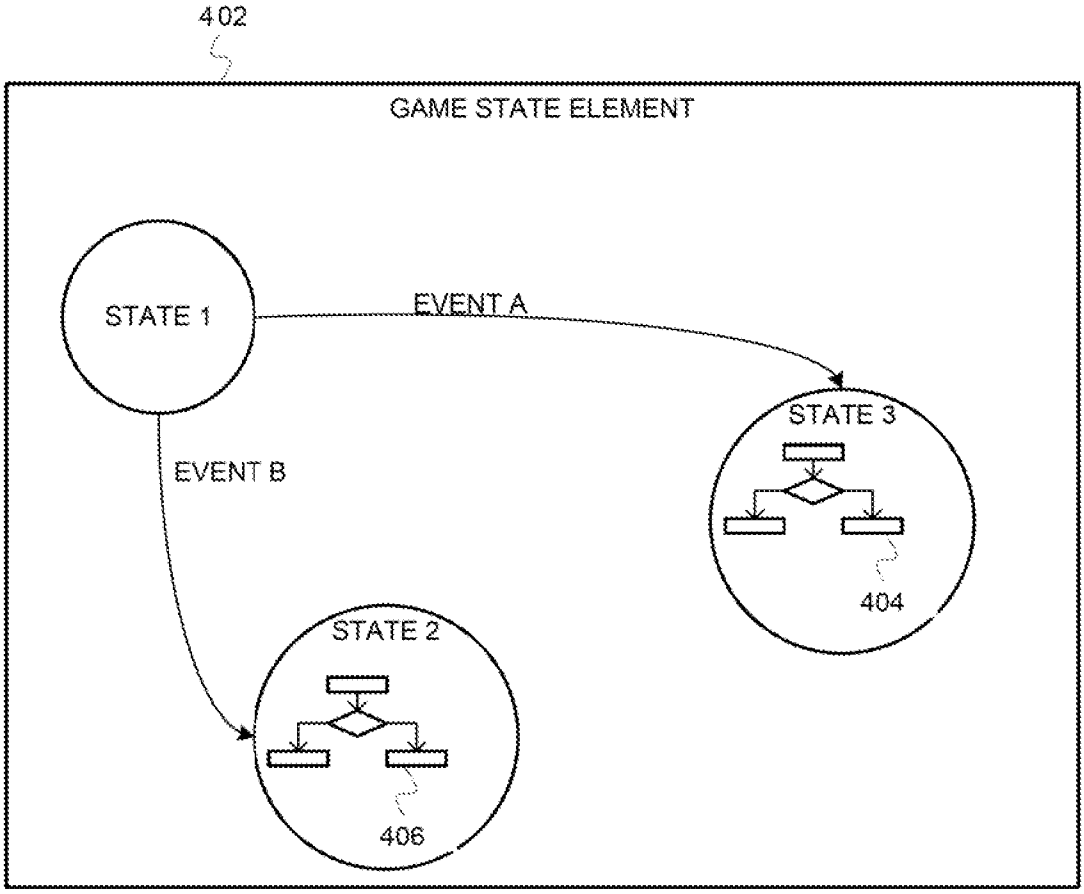

402
GAME STATE ELEMENT
STATE 1
EVENT A
EVENT B
STATE 3
404
STATE 2
406

FIG. 4

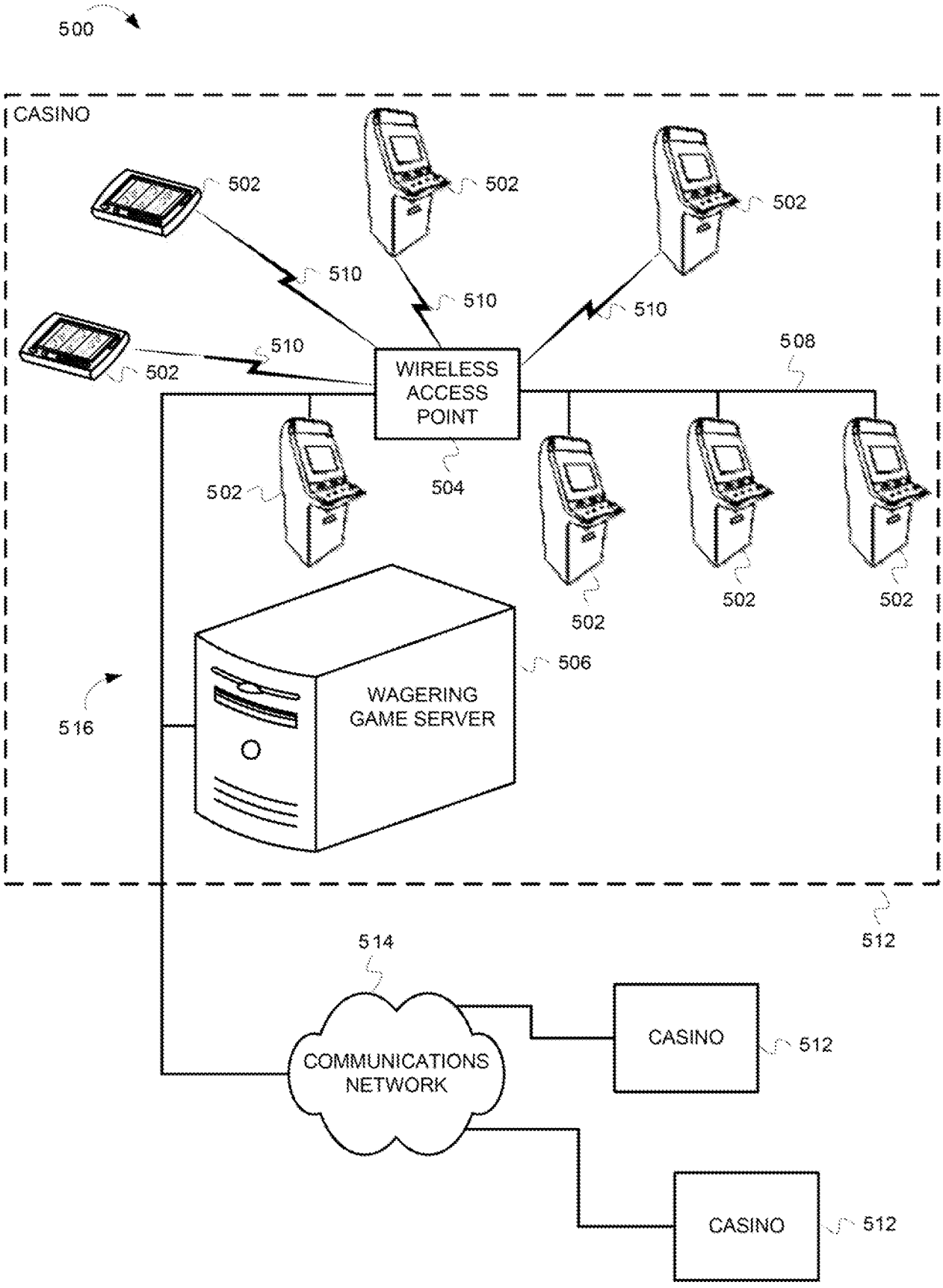

500
CASINO
502
502
502
510
510
510
502
510
WIRELESS ACCESS POINT
504
508
502
502
502
502
WAGERING GAME SERVER
506
516
512
514
COMMUNICATIONS NETWORK
CASINO
512
CASINO
512

FIG. 5

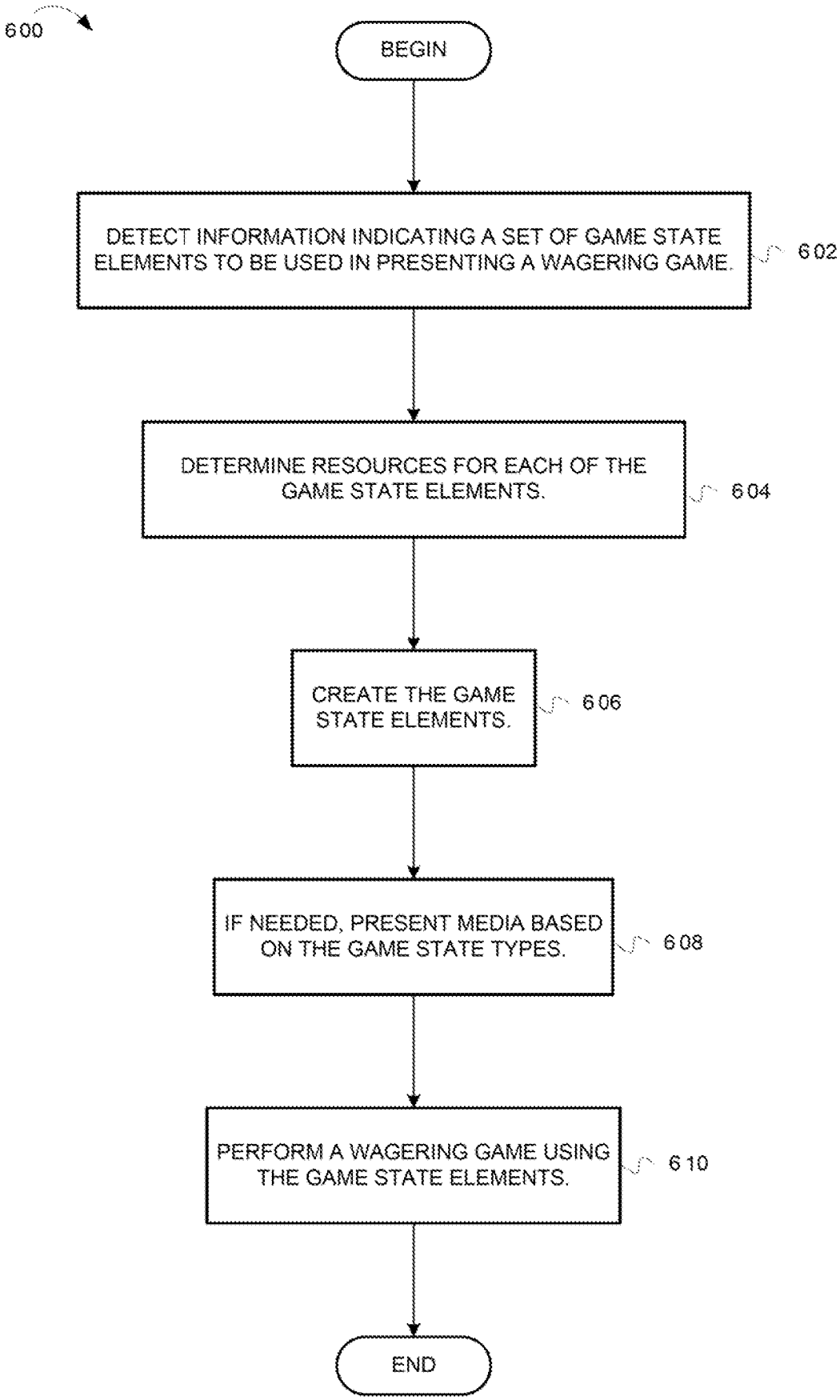

600
BEGIN
DETECT INFORMATION INDICATING A SET OF GAME STATE ELEMENTS TO BE USED IN PRESENTING A WAGERING GAME.
602
DETERMINE RESOURCES FOR EACH OF THE GAME STATE ELEMENTS.
604
CREATE THE GAME STATE ELEMENTS.
606
IF NEEDED, PRESENT MEDIA BASED ON THE GAME STATE TYPES.
608
PERFORM A WAGERING GAME USING THE GAME STATE ELEMENTS.
610
END

FIG. 6

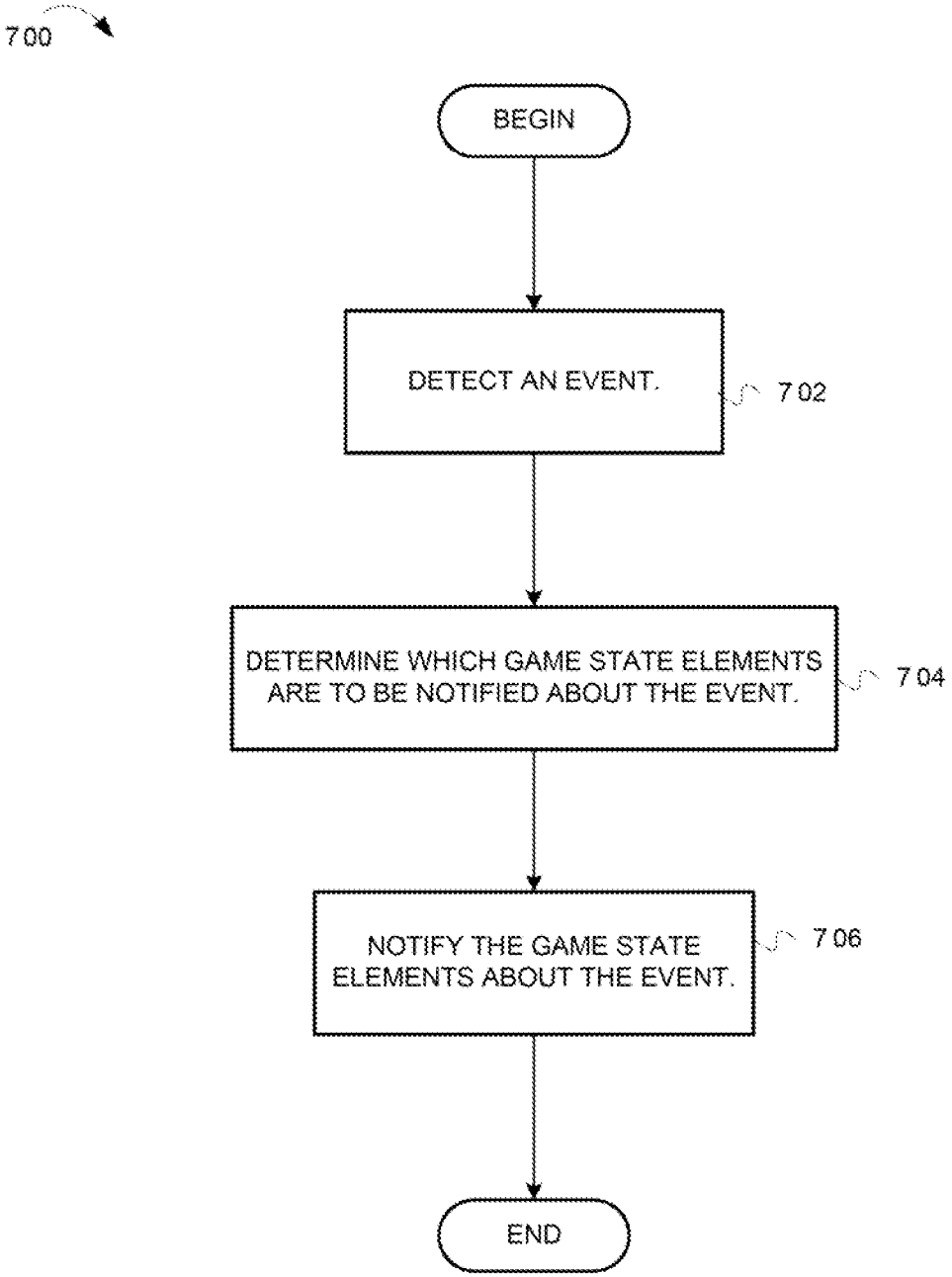

700
BEGIN
DETECT AN EVENT.
702
DETERMINE WHICH GAME STATE ELEMENTS ARE TO BE NOTIFIED ABOUT THE EVENT.
704
NOTIFY THE GAME STATE ELEMENTS ABOUT THE EVENT.
706
END

FIG. 7

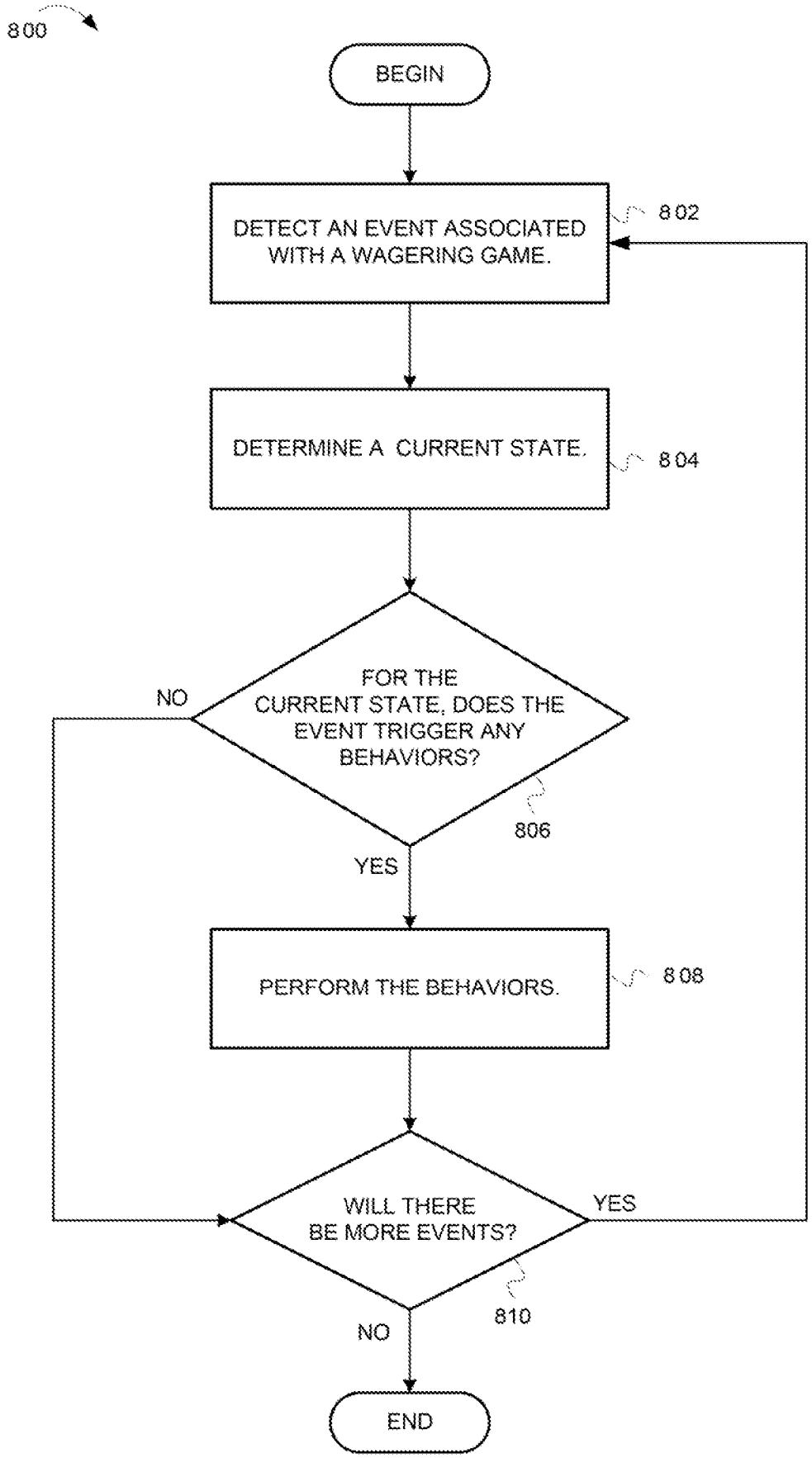

800
BEGIN
DETECT AN EVENT ASSOCIATED WITH A WAGERING GAME.
802
DETERMINE A CURRENT STATE.
804
FOR THE CURRENT STATE, DOES THE EVENT TRIGGER ANY BEHAVIORS?
806
NO
YES
PERFORM THE BEHAVIORS.
808
WILL THERE BE MORE EVENTS?
810
YES
NO
END

FIG. 8

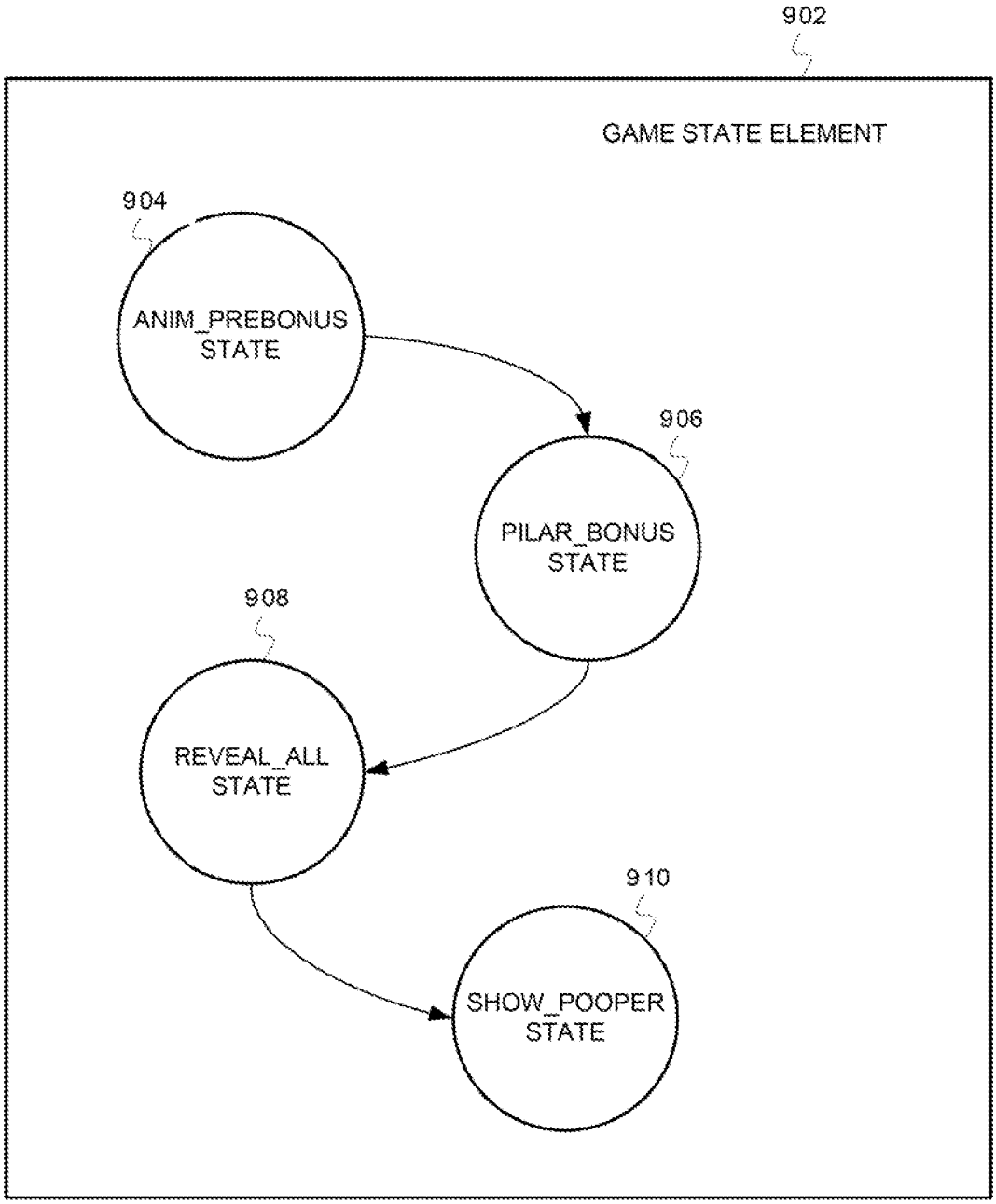

902
GAME STATE ELEMENT
904
ANIM_PREBONUS STATE
906
PILAR_BONUS STATE
908
REVEAL_ALL STATE
910
SHOW_POOPER STATE

FIG. 9

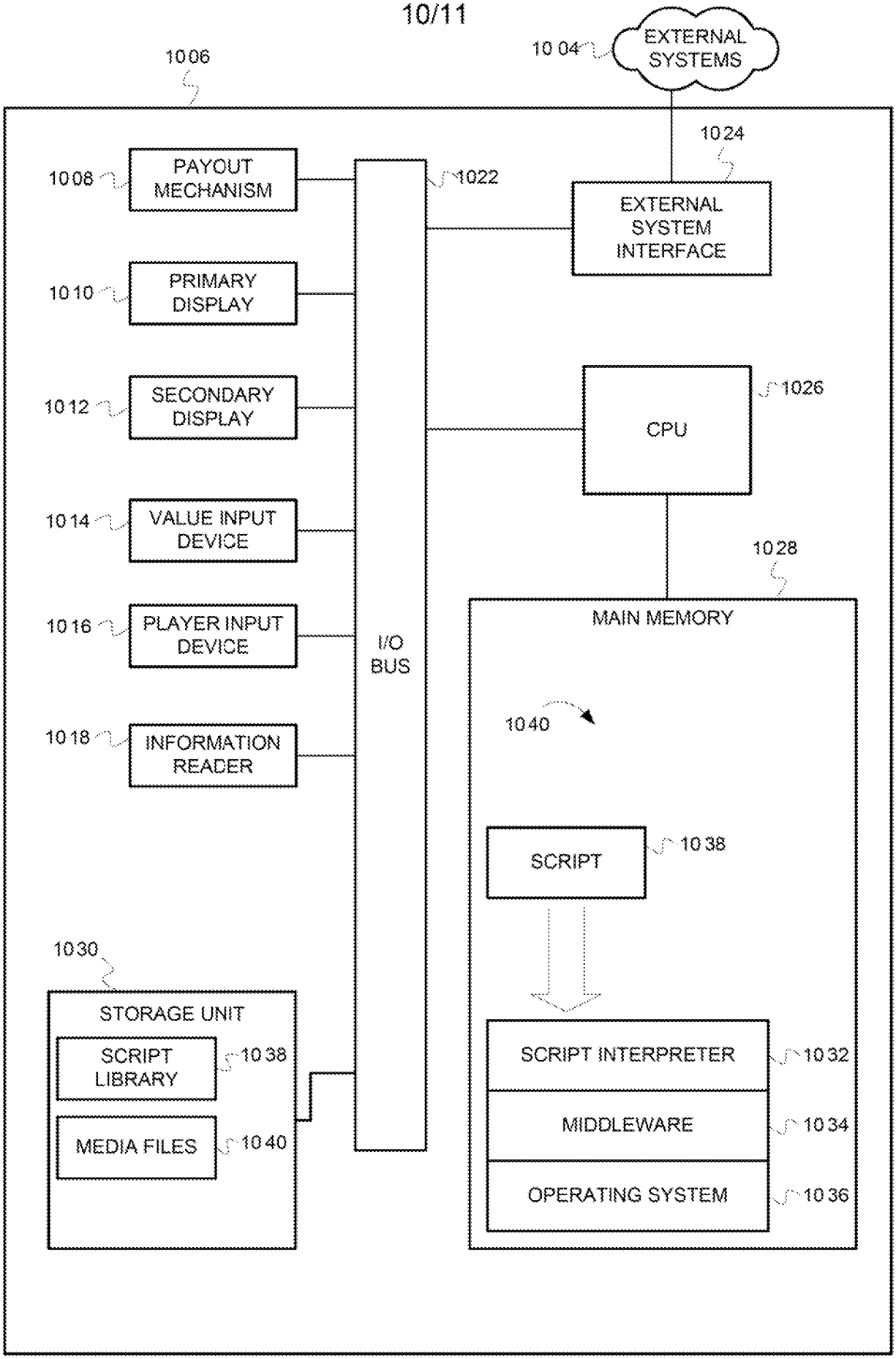

1006
1004
EXTERNAL SYSTEMS
1024
1008
PAYOUT MECHANISM
1022
EXTERNAL SYSTEM INTERFACE
1010
PRIMARY DISPLAY
1012
SECONDARY DISPLAY
1026
CPU
1014
VALUE INPUT DEVICE
1028
1016
PLAYER INPUT DEVICE
MAIN MEMORY
I/O BUS
1018
INFORMATION READER
1040
1038
SCRIPT
1030
STORAGE UNIT
SCRIPT LIBRARY
1038
SCRIPT INTERPRETER
1032
MEDIA FILES
1040
MIDDLEWARE
1034
OPERATING SYSTEM
1036

FIG. 10

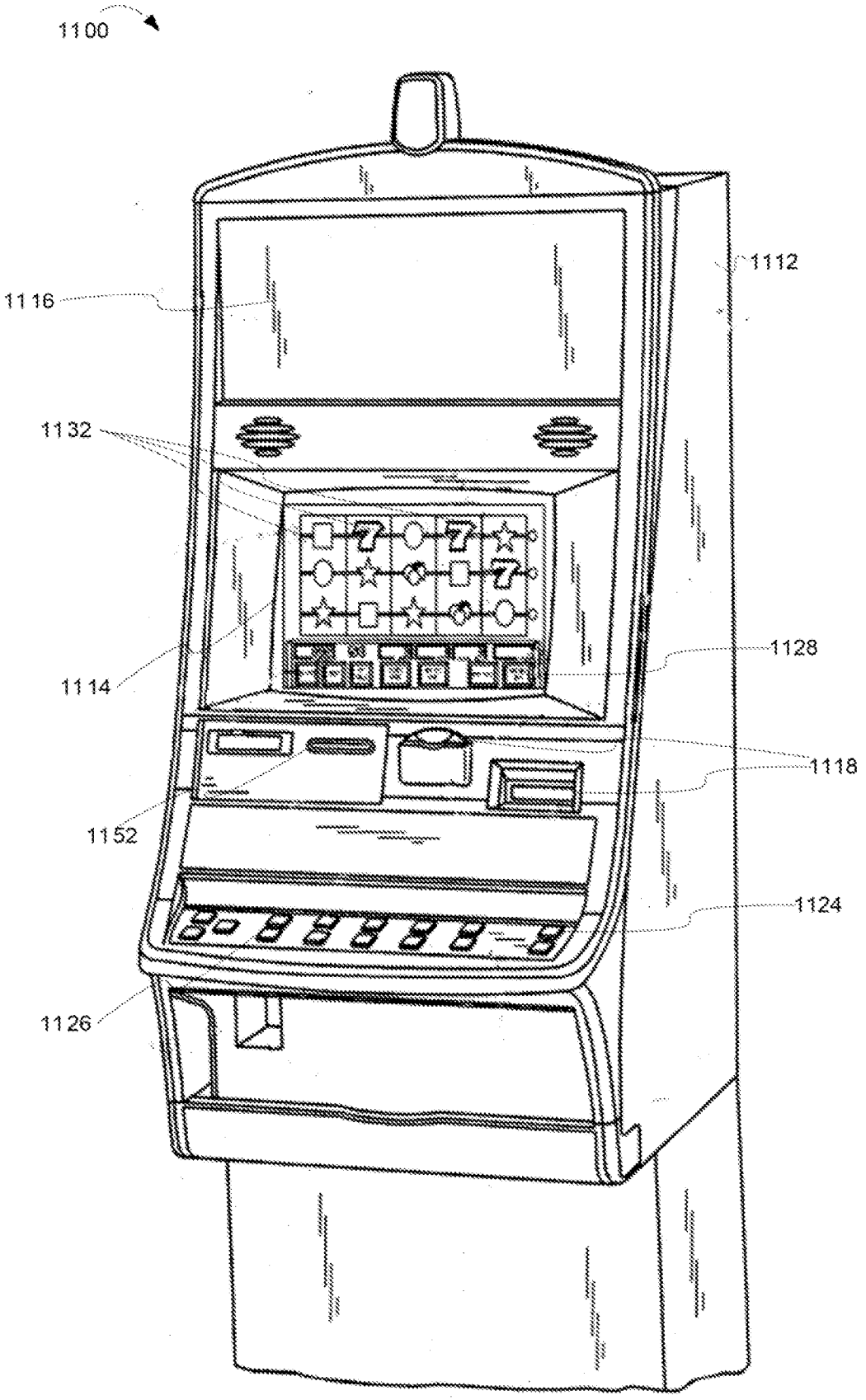


FIG. 11