



(43) International Publication Date
13 February 2014 (13.02.2014)

(10) International Publication Number
WO 2014/025910 A1

(51) International Patent Classification:
G09G 5/00 (2006.01)

(21) International Application Number:
PCT/US2013/053989

(22) International Filing Date:
7 August 2013 (07.08.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/569,048 7 August 2012 (07.08.2012) US

(71) Applicant: CYPRESS SEMICONDUCTOR CORPORATION [US/US]; 198 Champion Court, San Jose, CA 95134 (US).

(72) Inventors; and

(71) Applicants : WRIGHT, David [US/US]; 15316 23rd Avenue NE, Woodinville, WA 98077 (US). SEGUINE, Ryan [US/US]; 9217 3rd NW, Seattle, WA 98117 (US). KOLOKOWSKY, Steve [US/US]; 7915 Rufus Ct, San Diego, CA 92129 (US). YOUNG, David [US/US]; 2691 W Cedar Grove St, Meridian, ID 83646 (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, BN, 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, 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, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, 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, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, 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, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— as to the identity of the inventor (Rule 4.17(i))

Published:

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

(54) Title: SENSOR SYSTEM AND METHOD FOR MAPPING AND CREATING GESTURES

(57) Abstract: A computing system includes a sensor configured to detect user inputs. The system further includes a processor configured to receive a detected first user input from the sensor. The processor further receives a detected second user input from the sensor. In response, the processor assigns a command to the first user input based on the second user input.



WO 2014/025910 A1

SENSOR SYSTEM AND METHOD FOR MAPPING AND CREATING GESTURES

RELATED APPLICATIONS

[0001] This application is a continuation-in-part application of U.S. Patent Application No. 12/702,930 filed on January 25, 2011 which claims the benefit of U.S. Provisional Application No. 61/150,835 filed on February 9, 2009, both of which are hereby incorporated by reference herein.

TECHNICAL FIELD

[0002] The present disclosure relates generally to input methods and particularly characteristic detection for sensor devices.

BACKGROUND

[0003] Computing devices, such as notebook computers, personal data assistants (PDAs), kiosks, and mobile handsets, have user interface devices, which are also known as human interface devices (HID). One user interface device that has become more common is a touch-sensor pad (also commonly referred to as a touchpad). A basic notebook computer touch-sensor pad emulates the function of a personal computer (PC) mouse. A touch-sensor pad is typically embedded into a PC notebook for built-in portability. A touch-sensor pad replicates mouse X/Y movement by using two defined axes which contain a collection of sensor elements that detect the position of a conductive object, such as a finger. Mouse right/left button clicks can be replicated by two mechanical buttons, located in the vicinity of the touchpad, or by tapping commands on the touch-sensor pad itself. The touch-sensor pad provides a user interface device for performing such functions as positioning a pointer, or selecting an item on a display.

These touch-sensor pads may include multi-dimensional sensor arrays for detecting movement in multiple axes. The sensor array may include a one-dimensional sensor array, detecting movement in one axis. The sensor array may also be two dimensional, detecting movements in two axes.

[0004] Another user interface device that has become more common is a touch screen. Touch screens, also known as touchscreens, touch panels, or touchscreen panels are display overlays. The effect of such overlays allows a display to be used as an input device, removing the keyboard and/or the mouse as the primary input device for interacting with the display's content. Such displays can be attached to computers or, as terminals, to networks. There are a number of types of touch screen technologies, such as optical imaging, resistive, surface acoustical wave, capacitive, infrared, dispersive signal, piezoelectric, and strain gauge technologies. Touch screens have become familiar in retail settings, on point-of-sale systems, on ATMs, on mobile handsets, on kiosks, on game consoles, and on PDAs where a stylus is sometimes used to manipulate the graphical user interface (GUI) and to enter data. A user can touch a touch screen or a touch-sensor pad to manipulate data. For example, a user can apply a single touch, by using a finger to press the surface of a touch screen, to select an item from a menu.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] Embodiments are illustrated by way of example and are not intended to be limited by the figures of the accompanying drawings, in which like references indicate similar elements and in which:

[0006] **Figure 1** illustrates a system for detecting contacts and assigning gestures and executing commands according to an embodiment;

[0007] **Figure 2** illustrates a workshop GUI for gesture definition according to an embodiment.

[0008] **Figure 3** illustrates a control panel GUI for gesture parameterization according to an embodiment;

[0009] **Figure 4** illustrates a heads-up display for gesture display according to an embodiment;

[0010] **Figure 5** illustrates a method for assigning and maintaining gestures according to an embodiment;

[0011] **Figure 6** illustrates a method for executing commands and triggers according to the present invention;

[0012] **Figure 7** illustrates method for selecting from a list of possible gestures according to an embodiment;

[0013] **Figure 8A** illustrates a horizontal slider according to an embodiment;

[0014] **Figure 8B** illustrates a vertical slider according to an embodiment;

[0015] **Figure 8C** illustrates a radial slider or control knob according to an embodiment;

[0016] **Figure 8D** illustrates a plurality of buttons according to an embodiment;

- [0017] **Figure 8E** illustrates a single contact geometric shape according to an embodiment;
- [0018] **Figure 8F** illustrates a two-contact geometric shape according to an embodiment;
- [0019] **Figure 8G** illustrates a three-contact geometric shape according to an embodiment;
- [0020] **Figure 9A** illustrates a compass needle for two contacts for rotate gestures according to an embodiment;
- [0021] **Figure 9B** illustrates a compass needle for two contacts for rotate gestures according to an embodiment;
- [0022] **Figure 9C** illustrates a compass needle for three contacts for rotate gestures according to an embodiment;
- [0023] **Figure 10A** illustrates a move gesture for two contacts according to an embodiment;
- [0024] **Figure 10B** illustrates a move gesture for three contacts according to an embodiment;
- [0025] **Figure 11A** illustrates a expand/contract gesture for two contacts according to an embodiment;
- [0026] **Figure 11B** illustrates a expand/contract gesture for three contacts according to an embodiment;
- [0027] **Figure 12** illustrates a method for defining and applying gestures to contact locations according to an embodiment;

[0028] **Figure 13** illustrates absolute and relative display of movement on a sensor array according to an embodiment;

[0029] **Figure 14** illustrates a method for teaching a processor which gestures apply to detected characteristics according to an embodiment;

[0030] **Figure 15** illustrates a method for recording gestures according to an embodiment;

[0031] **Figure 16** illustrates a touchscreen device for receiving user input according to an embodiment;

[0032] **Figure 17** is a block diagram illustrating a computing device for implementing user creatable gestures and gesture mapping, according to an embodiment;

[0033] **Figure 18A** is a flow diagram illustrating a gesture mapping method, according to an embodiment;

[0034] **Figure 18B** is a diagram graphically illustrating the gesture mapping method of Figure 18A;

[0035] **Figure 19A** is a flow diagram illustrating a gesture mapping method, according to an embodiment;

[0036] **Figure 19B** is a diagram graphically illustrating the gesture mapping method of Figure 19A;

[0037] **Figure 20A** is a flow diagram illustrating a gesture mapping method, according to an embodiment;

[0038] **Figure 20B** is a diagram graphically illustrating the gesture mapping method of Figure 20A;

[0039] **Figure 20C** is a flow diagram illustrating a gesture mapping method, according to an embodiment;

[0040] **Figure 20D** is a diagram graphically illustrating the gesture mapping method of Figure 20C;

[0041] **Figure 21A** is a flow diagram illustrating a method for user creatable gestures, according to an embodiment;

[0042] **Figure 21B** is a diagram graphically illustrating the gesture mapping method of Figure 21A;

[0043] **Figure 22A** is a flow diagram illustrating a method for user creatable gestures, according to an embodiment; and

[0044] **Figure 22B** is a diagram graphically illustrating the gesture mapping method of Figure 22A;

DETAILED DESCRIPTION

[0045] In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of embodiments of the present invention. It will be evident, however, to one skilled in the art that the embodiments may be practiced without these specific details. In other instances, well-known circuits, structures, and techniques are not shown in detail or are shown in block diagram form in order to avoid unnecessarily obscuring an understanding of this description.

[0046] Reference in the description to "one embodiment" or "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the invention. The appearances of the phrase "in one embodiment" in various places in the specification do not necessarily all refer to the same embodiment.

[0047] A system, method and apparatus are described for detecting a user input on a sensor array and defining and executing commands based on that user input. The commands are defined using a configuration tool and through feedback with either a developer implementing gestures for a user interface or by the user of that interface. A display device for displaying user input, commands and parameters is described as either a stand-alone application or a heads-up display (HUD) visible during typical operation of an operating system.

[0048] Gesture detection and detection method development methods and systems are described. Gestures include interactions of an activating element, such as a finger, with an input device that produce an output readable by a controller or processor.

Gestures can be single point interactions, such as tapping or double-tapping. Gestures can be prolonged interactions such as motion or scrolling. Gestures can be interactions of a single contact or multiple contacts.

[0049] The response of a GUI to user inputs may be defined during development. Developers employ usability studies and interface paradigms to define how a sensing device interprets user input and outputs commands to a host, application processor or operating system. The process for developing and defining gestures and other interactions with a sensing device that cause a feedback event, such as a command to an application or display change, has been hidden to the user of the product. Each gesture may be built from the ground up or constructed from pieced-together lines of code from a library.

[0050] Embodiments of the present invention allow for the definition of gestures and other interactions with a GUI through an input device.

[0051] A gesture is an end-to-end definition of a contacts interaction with a movement with regard to a sensor array through the execution of a user's intent on a target application or program. The core of a gesture's purpose is to derive semantic meaning and detail from a user and apply that meaning and detail to a displayed target. A "gidget" is a control object which is located in a location relative to a sensor array. A gidget's location may be the entire sensor, such as in a motion gesture, or it may be a specific location or region, such as in buttons activation gestures or scrolling. Gidgets implement metaphoric paradigms for creating and implementing gestures. Metaphoric paradigms represent motions that a user would naturally take in response to and in an effort to control display targets. Such motions include, but are not limited to, rotation, panning, pinching and tapping.

[0052] Multiple gadgets can be associated with a sensor array depending on the application specifications. Gadgets are capable of operating independently, each tracking its own state and producing gestures according to its own set of rules. Multiple gadgets are also capable of working in concert to produce gestures based on a combination of cascading rules discussed herein. In either case, single gadgets or multiples of gadgets send control information to targets, such as cursors or menu items, in an application or operating system. To streamline and prioritize the interactions of gadgets where and when they overlap, a hierarchy may be defined to allow top-level gadgets to optionally block inputs to and outputs from low-level gadgets. In an embodiment, low-level gadgets may be buttons and high-level gadgets may be vertical and horizontal sliders. In this embodiment, a motion on the sensor would not activate buttons if the horizontal or vertical slider gadgets are active.

[0053] Figure 1 shows a system 100 for detecting a contact or contacts, interpreting that contact or contacts into a gesture and providing feedback for the definition and development of the gesture detection and interpretation. Contact 102 is detected on sensor array 104 by sensor controller 106. Sensor array 104 may be a capacitive sensor array. Methods for detecting contact 102 on sensor array 104 by sensor controller 106 are described in "Methods and Circuits of Measuring the Mutual and Self Capacitance" Serial No. 12/395,462, filed 27-Feb-2009, and "Capacitance To Code Converter With Sigma-Delta Modulator" Serial No. 11/600,255, filed 14-Nov-2006, the entire contents of each are incorporated by reference herein. Sensor controller 106 reports contact information, such as capacitance counts for sensor array 104 to track pad controller 108. Track pad controller receives contact information from sensor controller

106 and calculates contact position for each contact. Track pad controller 108 sends contact position information for each contact to operating system 112 through track pad drivers 110A and 110B. Trackpad drivers 110A and 110B communicate position information to application 114 and gadget controller 116.

[0054] Application 114 comprises the current program for which the contact interaction with the input sensor array 104 applies. Application 114 may also comprise a control panel GUI 120, a heads-up display (HUD) 122 and a workshop GUI 124, the workshop GUI allows a designer or a user to define gadgets and gadget sets. In one embodiment, the control panel GUI 120, HUD 122 and workshop GUI 124 may be the entirety of the application. In another embodiment, the control panel GUI 120, HUD 122 and workshop GUI 124 may be present alone or in combination in simultaneous operation of a current program 118. Current program 118 may be a photo editing program, a word processing program, a web browsing program or any program for which user interaction is applicable and for which gestures are detected. Gadget controller 116 accesses a memory 126 on which are stored at least one gadget set 131-136. While six gadget sets are shown in Figure 1, it would be obvious to one of ordinary skill in the art to implement a solution with fewer or more gadget sets based on needs of the system 100. Gadget sets are a collection of definitions for gadgets and of real-time HUD options for HUD 122. Gadget sets 131-136 are assigned to groups 141-144. A group is a category of gadget sets and can be stored in a different memory location or implemented through naming conventions for gadget sets, which is understood by gadget controller 116. While four groups are shown in Figure 1, it would be obvious to one of ordinary skill in the art to implement a solution with fewer or more group sets based on specifications of the system 100. In one

embodiment, three group sets may comprise a total of six gidget sets, with at least three gidget sets in each group.

[0055] Groups are assigned to gidget libraries (A and B) 128 and 129. Gidget libraries are folders or memory locations which contain a number of gidget sets that are specific to an application 114 or signed in user. The gidget controller 116 accesses gidgets that are available through a gidget sets 131-136 assigned to groups 141- 144, which are contained within a gidget library 128-129. When a different application 114 is opened or a new current program 118 is selected, the gidget controller accesses a different gidget set 131-136 through gidget libraries 128-129 and groups 141-144.

[0056] Still referring to Figure 1, the gidget controller is responsible for a number of tasks including:

- monitoring which application window is open or, if no application window is open, detecting the desktop,
- implementing new gidget sets 131-136 when a new application window or desktop comes into focus,
- driving gidget animations, which display the motion of each gidget as it is detected by the sensor, for the HUD 122,
- serializing event target commands toward the application 114 through the operating system 112.
- switching HID device driver streams, data streams from the device driver to the host or operating system, "on" and "off,"
- configuring a virtual HID such as a mouse, scroll, button, joy-stick and other game control devices, and

- injecting HID reports, which summarize the inputs and displays of the HID, into the virtual HID device driver.

The gadget controller is initiated as a start-up application in the user's application space.

[0057] For each gadget, there are associated a number of events. Events relate a gadget's motion or state to an object in the application or operating system with which the user is interacting through the sensor array 102 (Figure 1). Table 1 lists types of events and their configurable filtering parameters.

Table 1. Events and Configurable Parameters

Event Type	Event	Configurable Parameters
Linear Motion (Pixels or Percent of Range)	Moving	Distance, Speed, Acceleration, Assigned Contact Location Count
	Moved	Distance, Average Speed, Max Acceleration, Assigned Contact Location Count
Rotational Motion (Degrees or Relative Measure)	Rotating	Distance, Speed, Acceleration, Assigned Contact Location Count
	Rotated	Distance, Average Speed, Max Acceleration, Assigned Contact Location Count
Expand/Contract Motion (Percent of Expansion or Contraction)	Expanding	Distance, Speed, Acceleration, Assigned Contact Location Count
	Expanded	Distance, Average Speed, Max Acceleration, Assigned Contact Location Count
	Contracting	Distance, Speed, Acceleration, Assigned Contact Location Count
	Contracting	Distance, Average Speed, Max Acceleration, Assigned Contact Location Count
Tapping Motion (Up Count / Down Count)	Down	Distance, Speed, Acceleration, Assigned Contact Location Count
	Up	Distance, Average Speed, Max Acceleration, Assigned Contact Location Count

[0058] Events may be defined in sequences, so that when motion parameters are filtered out for a higher-priority event, subsequent events in the sequence are not evaluated and do not produce triggers. A trigger is an action that the gadget controller 116 (Figure 1)

applies to an application of operating system. A trigger is stopped when filtering criteria are satisfied.

[0059] Event can also be aligned to create a set of overlapping filter requirements and form a series of AND conditions. One embodiment of a set of overlapping filter requirements is an event for "growing" may block lower priority events for "moving" or "rotating" on the same gidget.

[0060] Figure 2 shows an exemplary layOut for workshop GUI 124 (Figure 1). The workshop GUI 124 allows a designer or a user to define gidgets and gidget sets. The workshop GUI contains regions for storage and configuration management 202 and gidget and event definition 204. The contact's (102) interaction with sensor array 104 is shown in the HUD simulation window 206, where the contacts are shown as visual representations and indication of the output are also displayed. The HUD simulation window 206 is configured to display characteristics of the contact 102 based on selections in the HUD simulation options window 208. Real-time events and parameters that are visually apparent to the user are displayed and controlled in the real-time HUD options window 210. In an embodiment, real-time events may be contact detection and identification, gesture outputs, or contact location calculation. In another embodiment, parameters that may be visually apparent to the user may be the sensitivity of the sensor or the latency of gesture detection. Triggers for events and gidgets are displayed in the real-time trigger feedback window 212, which displays the detected interactions and how they are interpreted by the gidget controller 116 and displayed in the HUD simulation window 206. Configurations and parameters are written to the gidget controller or the track pad controller through application target 216.

[0061] Figure 3 illustrates an embodiment for the control panel GUI 120. The control panel GUI is accessed during normal operation of the operating system 112, application 114 and/or current program 118. The control panel GUI is accessed by a menu item 312 in the operating system 112 (shown), application 114 or current program 118. The control panel GUI 120 allows a user to select from various personalities or parameter sets 302 for the track pad. The control panel also allows the user to define from among global defaults such as HID availability 304 and HID opacity 306.

[0062] Figure 4 illustrates an embodiment of HUD 122 of Figure 1. The HUD 122 is accessed by selecting a menu item 412 in the operating system 112. The HUD 122 is shown in the operating system as a separate window from the application 114 or current program 118. The HUD 122 displays the contacts 402, 404 and 406 as they contact the sensor array 102 of Figure 1. The HUD 122 shows the shape 408 and center-of-mass defined by the contacts 402, 404 and 406 if possible. That is, a single point of contact will not have shape, only a single location. Two points of contact will define a line, but not a shape. The HUD 122 also illustrates a summary 420 of the interaction. The HUD 122 may be displayed as always on top, always on bottom or some location in between depending on user settings and preferences. The HUD display is configured by the control panel GUI 120 and provides the user and/or the developer with real-time graphical feedback. In one embodiment, the HUD 122 may display individual gidget states which may be displayed as status text, positional read-outs, or a graphical outline. In another embodiment, the HUD 122 may present the location and identification of assigned contact locations. In another embodiment, the HUD 122 may present real-time, scaled gidget animation for use as visual feedback to user interaction with sensor array

102. In yet another embodiment, the HUD may display enlarged animation of gadgets for the sight-impaired, a game player or other experience enhancements, depending on the application. In still another embodiment, the HUD 122 may place a display GUI in a small "corner-of-the-eye" location for visual feedback for standard user input.

[0063] In one embodiment, HUD information may be stored in the gadget set to control the opacity of the HUD 122. In another embodiment, the stored information may be the ability of the HUD 122 to flash for a period of time when a new gadget set is activated or be always on or always off. In another embodiment, HUD settings may be set by the user in the control panel GUI 120 as well.

[0064] When a gadget has captured and is associated with a contact location (given by X, Y and Z position), it is active. Contact locations assigned to an active gadget are not available to lower-level gadgets when assigned to a higher-level gadget. Higher-level gadgets may access contact locations that are assigned to lower-level gadgets. Contact locations, once captured may be released according to Figure 5.

[0065] Figure 5 shows a method for assigning and releasing a contact location from an active gadget. The sensor array 102 (Figure 1) is scanned with sensor controller 104 (Figure 1) in block 510. Contact presence is detected in decision block 515. If no contact is detected, the sensor is scanned again in block 510. If a contact is detected, the X, Y and Z coordinates of the contact's location are calculated in block 520. If more than one contact is detected, X, Y and Z coordinates of each contact's location are calculated. The contact location or locations are assigned to a gadget in block 530. The assignment of gadgets to contact locations is determined by gadget hierarchy, which may be defined in development or by the user. The combination of a gadget and a contact location defines

the assigned contact location in block 540. The defined contact location is displayed in the HUD 122 (Figure 1) if the HUD 122 is open in block 550. If the contact is not maintained on the sensor array 102 in decision block 555, the assigned contact location is released in block 570 and the associated active gidget is no longer active. If the contact is maintained on the sensor array 102 in decision block 555, the contact's location is compared to a retention perimeter for the active gidget in decision block 565. If the contact location is within the retention perimeter for the active gidget, the assigned contact location is maintained in block 580 and contact detected again to determine if it is still present. If the contact location is outside the retention perimeter for the active gidget, the assigned contact location is released in block 570 and the associated active gidget is no longer active. After release of the contact location, the sensor array is scanned again in block 510.

[0066] Figure 6 shows the process for starting and stopping a trigger. A contact location or a plurality of contact locations are assigned to a gidget in block 610. The action, such as movement across the sensor or stationary position in an embodiment, of the contact or contacts is detected in block 620. The contact action is compared to allowable actions for the active gidget for the assigned contact location by the gidget controller 116 (Figure 1) in decision block 625. If the contact action is outside the allowed parameters for the active gidget, no action is taken in block 630. If the contact action is within the allowed parameters for the active gidget, the appropriate trigger is identified in block 640. The identified trigger is applied to the application 114 (Figure 1) or current program 118 (Figure 1) in block 650. Trigger filter criteria are applied in block 660. Trigger filter criteria are specific to the trigger and the active gidget. If the trigger

filtering criteria is determined not to be satisfied by the gadget controller 116, the trigger is maintained and continues to be applied to the application 114 or current program 118 in block 650. If the gadget controller determines that the trigger filtering criteria are satisfied, the trigger is stopped in block 680.

[0067] As stated before, events are specific to gadgets. Gadgets can be global or specific to applications. To apply the correct event based on the user interaction with the sensor array 102 (Figure 1), the process of Figure 7 is followed. For process 700, an application window detected in decision block 705. If an application window is open, a "Top Window" focus is detected from the application window in block 710. The "Top Window" focus is the open window to which user input is applied. As the user interacts with the system, the "Top Window" focus may change as new applications are opened or windows are activated and deactivated in the display. The "Top Window" focus is applied in block 720. The "Top Window" focus from blocks 710 and 720 may instruct the gadget controller to apply installation defaults in block 730 or it may instruct the gadget controller to apply personality selections in block 740. Personality selections are made in the Control Panel GUI 120 (Figure 1) and select a gadget set for the interface between the user and the application 114 or current window 118 (Figure 1). Personality selections may be set for a specific user in one embodiment. In another embodiment, personality selections may be defined for a genre of programs or applications. After installation defaults are applied in block 730 or personality selections are applied in block 740, the gadget set is selected in block 750. The "Top Window" focus maintained or not maintained in decision block 755 by the gadget controller 116 (Figure 1) based on the selected gadget set from block 750. If the "Top Window" focus is not maintained in

decision block 755, a new "Top Window" focus is detected in block 710 again. If the "Top Window" focus is maintained in decision block 755, decision block 775 determines if an event target to switch the gadget set is present. If the event target does not specify that the gadget set be switched, the gadget set is set again in block 750. If the gadget controller 116 (Figure 1) determines that the event target does require the gadget set be switched, the gadget set is switched to the new gadget set in block 780. If, in decision block 705, it is determined that an application window is not open, the gadget set for the desktop or default screen is selected in block 760. Decision block 765 determines if an event target to switch the gadget set is present. If the event target does not specify that the gadget set be switched, the gadget set is set again in block 760. If the gadget controller 116 determines that the event target does require the gadget set be switched, the gadget set is switched to the new gadget set in block 780.

[0068] Gadget sets are assembled into gadget libraries as shown in Figure 1. Gadget libraries define user-targeted solutions for applications. The gadget controller accesses the gadget libraries for the detected application. Gadget libraries may be defined during development or by the user in real-time. The user accesses and assigns gadget libraries through the control panel GUI 120 (Figure 1). The control panel GUI specifies the preferences in which the selected gadget sets are used (shown in Figure 7) or turn on and off gadget sets for an application.

[0069] A gadget is a control object location on a sensor array. In some embodiments, gadgets may appear as horizontal sliders, vertical sliders, rotational sliders or knobs, buttons, geometric shapes or contact plane. Each gadget type may be defined multiple times. Events capture assigned contact locations for active gadgets subject to a

hierarchy and blocking rules. The workshop GUI allows the hierarchy to be rearranged and blocking rules to be redefined according to application requirements.

[0070] Examples of gadgets are shown in Figures 8A-8F. Figure 8A shows an example of a horizontal slider 800 that may be displayed on HUD 122 (Figure 1) according to one embodiment. Horizontal slider 810 tracks the position of a contact 802 or a number of contacts in one horizontal dimension 804. Slider elements 806(1) through 806(N) simulate a hardware-based horizontal slider or switch. A horizontal slider gadget may support a number of event types include, but not limited to, moving, moved, expanding, expanded, contracting, contracted, up and down.

[0071] Figure 8B shows an example of a vertical slider 820 that may be displayed on HUD 122 (Figure 1) according to one embodiment. Vertical slider 820 tracks the position of a contact 822 or a number of contacts in one vertical dimension 824. Slider elements 826(1) through 826(N) simulate a hardware-based horizontal slider or switch. A vertical slider gadget may support a number of event types include, but not limited to, moving, moved, expanding, expanded, contracting, contracted, up and down.

[0072] Figure 8C shows an example of a radial slider (rotational knob) 840 that may be displayed on HUD 122 (Figure 1) according to one embodiment. Radial slider 840 tracks the position of a contact 842 or a number of contacts in relation to a reference axis. Slider elements 846(1) through 846(N) simulate a hardware-based radial slider or control knob. A radial slider gadget may support a number of event types include, but not limited to, rotating, rotated, up and down.

[0073] Figure 8D shows examples of buttons that may be-used as gadgets according to one embodiment. Button gadgets may include, but are not limited to, up

triangle 862, down triangle 864, left triangle 866, right triangle 868, square 870, circle 872 and icon 874. The displayed "icon" button is not representative of the only icon that can be used as an icon button gadget, rather it is shown as an example only. Buttons gadgets may support event types including, but not limited to, up and down.

[0074] Figure 8E shows examples of geometric shape gadgets. Geometric shape gadgets are defined by the number and configuration of contacts. A point gadget 881 is comprised of a single contact. A line gadget 882 is comprised of two contacts, 883 and 884, and the line 885 that connects them. A triangle gadget 886 is comprised of three contacts, 887-889, and the lines 891-893 that connect them. Geometric shape gadgets comprising more than three contacts are defined by those contacts and the non-overlapping connections between them. Geometric shape gadgets may have different events assigned to each configuration based on the number of contacts or other parameters. If no events are assigned to a particular geometric shape gadget, a contact configuration in that the geometric shape may not have an output visible to the user or readable by the application or operating system. In one embodiment, events that are defined for a line gadget but not for a triangle gadget are captured and are displayed in the HUD for two contacts on the sensor array. However, a third contact on the sensor array creates a triangle gadget, which does not have associated events and are not displayed in the HUD. In another embodiment, two geometric gadgets can be defined and assigned events. In such an embodiment and with three contacts present on the sensor array, an active line gadget and an active triangle may be readable by the gadget controller and a line and triangle displayed in the HUD and available for interaction with the application or operating system. A geometric shape gadget may support a number of event types

including, but not limited to, rotating, rotated, moving, moved, expanding, expanded, contracting, contracted, up and down.

[0075] Figures 9A-C show how line and triangle geometric shape gidgets are used to execute and display "rotated" and "rotating" events. Figure 9A shows a first embodiment 900 of two contacts 902 and 904 connected by line 903 and defining compass point 905. Compass point 905 has a direction that is parallel to the line 903 between contacts 902 and 904. As contacts 902 and 904 move and line 903 rotates, compass point 905 also rotates to remain parallel to line 903. Compass point 905 points to the right in Figure 9A. In one embodiment, the direction of the compass point may be defined by which contact, 902 or 904, is higher. In another embodiment, the direction of the compass point may be defined by which contact, 902 or 904, is in most contact with the sensor.

[0076] Figure 9B shows a second embodiment 910 of contacts 902 and 904 connected by line 903 and defining compass point 906. Compass point 906 has a direction that is perpendicular to the line 903 between contacts 902 and 904. As contacts 902 and 904 move and line 903 rotates, compass point 906 also rotates to remain perpendicular to line 903. Compass point 906 points up in Figure 9B. In one embodiment, the direction of the compass point 906 may always be positive on detection of multiple contacts. In another embodiment, the direction of the compass point 906 may be defined to point in positive or negative directions based on which contact, 902 or 904 is detected first and where the contacts are relative to each other.

[0077] Figure 9C shows an embodiment of three contacts 912, 914 and 916. In this embodiment, a rotate event is defined by the relative position of the lower two

contacts, 912 and 914. The line 918 connecting contacts 912 and 914 is used to define the compass point 920. As contacts 912, 914 and 916 move, line 918 connecting contacts 912 and 914 rotates and compass point 920 also rotates to remaining parallel to line 918.

While line and triangle geometric shapes are shown here, it is evident that different geometric shapes may be used to implement rotate and rotating events.

[0078] Figures 10A and 10B show how line and triangle geometric shape gadgets are used to execute and display "moving" and "moved" events. Figure 10A shows an embodiment 1000 of two contacts 1002 and 1004 connected by line 1003 and having a center of mass 1005. A moving or moved event is detected by calculating the position of the center of mass 1005 at a first time and comparing that position to the position of the same center of mass at a second time 1006. The path 1007 followed by the center of mass defines the moving or moved event.

[0079] Figure 10B shows an embodiment 1020 of three contacts 1022, 1023 and 1024 which define a shape 1026 having a center of mass 1028. A moving or moved event is detected by calculating the position of the center of mass 1028 at a first time and comparing that position to the position of the same center of mass at a second time 1030. The path 1032 followed by the center of mass defines the moving or moved event. Center of mass 1028 and 1030 are defined by Green's theorem:

Error!

$$\oint_C (L dx + M dy) = \iint_D \left(\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y} \right) dx dy.$$

$$\oint_C (L dx + M dy) = \iint_D \left(\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y} \right) dx dy.$$

Equation 1

wherein C is a positively oriented, piecewise smooth, simple closed curve in a plane and D is the region bounded by C . L and M are functions of x and y defined in an open region containing D and have continuous partial derivatives.

[0080] Figures 11A and 11B show how line and triangle geometric shape gadgets are used to execute and display "expanding," "expanded," "contracting" and "contracted" events. Figure 11A shows an embodiment 1100 of two contacts 1102(1) and 1104(1), which are connected by a line 1103(1) having a length L_1 . As contacts 1102(1) and 1104(1) move apart, they are shown as contacts 1102(2) and 1104(2) which are connected by line 1103(2) having length L_2 . The length of line L_1 in comparison to line L_2 defines the expansion or contraction events. If L_1 is greater than L_2 , a contraction event is defined. If L_2 is greater than L_1 , an expansion event is defined.

[0081] Figure 11B shows an embodiment of three contacts 1122(1), 1124(1) and 1126(1) which define a shape 1128(1) having an area A_1 . As contacts 1122(1), 1124(1) and 1126(1) move to new positions shown as contacts 1122(2), 1124(2) and 1126(2), a larger shape 1128(2) having an area A_2 is defined. A comparison of A_1 to A_2 defines expansion or contraction events. If A_1 is greater than A_2 , a contraction event is defined. If A_2 is greater than A_1 , an expansion event is defined.

[0082] As discussed herein, an event is defined in the workshop GUI 124 (Figure 1) in the gadget and event definition region 204 (Figure 2). Figure 12 shows the process by which an event is defined in the workshop GUI 124. The shape for a geometric shape or a standard gadget is defined in block 1210 by selecting from a list of available gadgets. An event type is defined in block 1220, by selecting from a list of available event types or adding a non-standard event type in an input window. Possible event types may include,

but are not limited to, rotating, rotated, moving, moved, expanding, expanded, contracting, contracted, up and down. Event parameters are defined in block 1230 by selecting options for displayed parameters from a list or adding non-standard parameters in an input window according to a set of conventions. Event parameters may include the rate or resolution of rotation, movement and expansion/contraction. Event parameters may also include hysteresis or delay in implementation of the event. Enable criteria are defined in block 1240 by selecting options for enabling the event from a list of possible criteria or by adding a non-standard criterion in an input window according to a set of conventions. Enable criteria define what is necessary for an event to be started and ended. Event type, parameters and enable criteria are then applied to the shape and gadget in block 1250. During development, the action of the gadget may be simulated in block 1260 to ensure that the movement or action detected by the sensor array translates to the desired event. The user or developer is then able to evaluate the performance of the parameters in block 1270, and enable criteria and adjust settings accordingly. In one embodiment, the contacts, gadgets events and triggers are all displayed. The user may see this combination and verify that it is the desired combination. If it is not the desired combination, parameters may be adjusted to change the output combination to meet the specification of the application.

[0083] In one embodiment, the position of a contact or contacts on the sensor array is mapped to the display as an absolute position. Gestures that involve cursor control in drawing applications may have the ability for the application to interpret contact or movement of contact over the sensor array without any relative position.

[0084] In another embodiment, the position of a contact or contacts on the sensor array is mapped to the display as a relative position on the sensor array and the display device. That is, movement that is 50% across the sensor array will be shown as cursor movement that is 50% across the display device.

[0085] Absolute and relative position is shown in Figure 13. Contact 1302 moves across the sensor device 1310 along path 1305. This movement is equivalent to approximately 50% of the width of the sensor array. An absolute position for the movement of contact 1302 along path 1305 is shown on display 1320 as path 1315. A relative position for the movement of contact 1302 along path 1305 is shown on display 1320 as path 1325. The relative motion on the display device may be a one-to-one relation in an embodiment, that is, the movement across the sensor device 1310 is directly proportional to the displayed movement on display 1320. In another embodiment, the relative motion on the display device may be a different ratio.

[0086] A gesture that is performed by a user may be learned by the gidget controller. One embodiment for gesture learning by the gidget controller is shown in Figure 14. Contacts are detected on the sensor array in block 1410. In an embodiment, contacts may be detected on the sensory array using a the capacitance measurement circuit configured perform a variety of well-known and understood sensing methods, including charge transfer filtering, relaxation oscillator charging, differential charge sharing between multiple capacitors, and others. In another embodiment, contacts may be detected using non-capacitive sensing methods such as surface-acoustic wave, field effect sensing, or infra-red or other optically-based methods. After contacts have been detected, the position for each contact is calculated in block 1420. There may be only one contact,

there may be several contacts. The shape that is defined by the initial placement of the contacts is determined in block 1430 by connecting each contact and comparing the contact arrangement and lines to exemplary arrangements stored in memory 126 (Figure 1). This shape is then tracked over multiple scans of the sensor array with contacts present to detect movement in block 1440. The contact shape and movement is compared to a list of possible gestures stored in memory 126 in block 1450. In one embodiment, each characteristic of the shape and movement is associated with a probability of a gesture being intended by the user. That is, three contacts may be associated with a rotate more often than four contacts, so a rotate gesture may have a greater probability of selection if there are only three contacts. However the intended gesture for four contacts may be a rotate, so it is computed with a probability. A probability table for all possible gestures for contact shape and movement is created in block 1460. Table 2 shows an example probability table according to one embodiment.

Table 2. Example Probability Table

Gesture	Number of Contacts				
	1	2	3	4	5
Tap	73	15	5	1	
Rotate	1	45	30	31	31
Move	20	20	35	36	36
Expand	1	10	15	16	16
Contract	1	10	15	16	16

Each gesture is assigned a probability of intent based on the shape and movement of the contacts. For the example probability table shown in Table 2, with three contact detected, a "move" gesture is selected.

[0087] The gesture with the greatest probability of intent is selected from the probability table and applied to the application in block 1470. Feedback is received from the user, application or operating system on the applied gesture in block 1480. This feedback could be in the form of an "undo gesture" command, response to a visual or audio prompt to the user, or a lack of response within a timeout period (signifying confirmation of the intended gesture). This feedback may be given in response to a presented gesture that happens when the user pauses on the sensor array or maintains the contacts in proximity to but not in direct contact with the array. Such an action can be referred to as a "hover." When the contacts hover above the array after a gesture has been performed the probable applied gesture may be presented for approval by the user. The applied gesture is confirmed or rejected based on the feedback from the user, application or operating system in block 1490. The probabilities of each gesture corresponding to the contact shape and movement are updated based on the confirmation or rejection of the applied gesture in block 1498. In one embodiment, confirmation of the applied gesture increases the probability that the applied gesture will be applied again for a similar contact shape and movement, while other gestures' probabilities are reduced. If a gesture is confirmed to be a "rotate" gesture, the a scalar is added to the rotate gesture in the probability table that increases the proportion of actions similar to that which was detected that are interpreted as a "rotate" gesture. In another embodiment, rejection of the applied gesture reduces the probability that the applied gesture will be applied again for a

similar contact shape and movement, while other gestures' probabilities are increased. In another embodiment, rejection or verification of the applied gesture that is repeated by the user a number of times set in development may eliminate or permanently confirm the applied gesture, respectively.

[0088] Specific gestures may be defined by the user through specific action. The user may instruct the controller to apply a gesture to specific pattern of contact and movement to create new user- or application-specific gestures. This instruction may be through a "recording" operation. One embodiment for teaching a gesture to the processor is shown in Figure 15. Gesture recording is begun in block 1510. The start of a gesture recording may be through a radio button, audio command or other GUI item. Contacts are detected on the sensor array in block 1520. Positions for each contact is calculated in block 1530. The shape defined by the contacts is determined in block 1540 and movement of that shape over successive scans of the sensor array is detected in block 1550. Gesture recording is stopped in block 1560. Stopping the gesture recording may be through a radio button, key strike audio command or other GUI item. Contact shape and movement are saved to memory in block 1570. The save contact shape and movement may be displayed for confirmation of intended motion. A list of possible gestures is then presented to the user for selection and application to the saved contact shape and movement and the user selects one of the presented gestures for application to the saved contact shape and movement in block 1580. The list of gestures can be presented while the contacts remain in direct contact with the sensor array or hovering over the sensor array. The selected gesture is then saved to memory in block 1590.

[0089] Another embodiment of the present invention is shown in Figure 16. A touchscreen device 1600, such as a LCD monitor or tablet computer has a touchscreen 1605 for user input. The touchscreen functions as a normal touchscreen, but cursor 1640 control is not through direct input but through a software touchpad 1610 displayed on the touchscreen 1605. The software touchpad 1610 is accessed through menu item 1612 by touching the touchscreen 1605 at the location of menu item 1612.

[0090] While gestures in the present application have been described as having only two up to dimensions, the system and methods described could be applied to three-dimension gestures. In such cases contact locations are defined by their X, Y and Z values relative to the sensor array. The addition of a third dimension adds possible gestures and interaction with the user that may not be described here but would be clear to one of ordinary skill in the art to use the described methods for detection and application to the system.

[0091] Figure 17 is a block diagram illustrating a computing device for implementing user creatable gestures and gesture mapping, according to an embodiment of the present invention. In one embodiment, the computing device 1700 is controlled by an operating system 1712. In one embodiment, operating system 1712 may be representative of operating system 112, described above with respect to Figure 1. Computing device 1700 may further include several computer application programs, such as applications 1720 and 1722. Applications 1720 and 1722 may be representative of application 114, described above with respect to Figure 1. In addition computing device 1700 may include gesture library 1730 and command library 1735, stored in a memory, such as memory 126. Gesture library 1730 may include a data structure storing

characteristics of one or more gestures which may be received by computing device 1700 as user input. The user input may be received by sensor array 1701, which may be representative of track pad 101, described above with respect to Figure 1. In certain embodiments, sensor array 1701 may include a track pad, touch screen, or other form of input device. The characteristics may include a number of contacts, the position of those contacts, relative and absolute motion of the contacts, etc. Command library 1735 may include a data structure storing a number of commands which may be executed by operating system 1712 or applications 1720 and 1722. The commands in command library 1735 may or may not be mapped to a gesture from gesture library 1730, so that when the gesture is received as a user input, the corresponding command may be executed.

[0092] Connected to computing device 1700 may include one or more peripheral devices, such as sensor array 1701, keyboard 1706 and display device 1708. In one embodiment some or all of these devices may be externally connected to computing device 1700, however, in other embodiments, some or all may be integrated internally with computing device 1700. Operating system 1712 of computing device 1700 may include drivers corresponding to each peripheral, including sensor array driver 1710, keyboard driver 1716 and display driver 1718. For example, a user input may be received at sensor array 1701. Sensor array driver 1710 may interpret a number of characteristics of the user input to identify a gesture from gesture library 1730. Sensor array driver 1710 may also determine if the identified gesture corresponds to a command from command library 1735 and may send a signal to an application 1720, causing application 1720 to execute the command.

[0093] Figure 18A is a flow diagram illustrating a gesture mapping method, according to an embodiment of the present invention. The method 1800 may be performed by processing logic that comprises hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (e.g., instructions run on a processing device to perform hardware simulation), or a combination thereof. The processing logic is configured to provide a method for gesture mapping to allow mapping of a received input gesture to a command to be performed by a computer application program. In one embodiment, method 1800 may be performed by computing device 1700, as shown in Figure 17. Figure 18B is a diagram graphically illustrating the gesture mapping method 1800 of Figure 18A.

[0094] Referring to Figure 18A, at block 1810, method 1800 receives a user input. In one embodiment, the user input may include a gesture performed by a user on an input device, such as sensor array 1701. The gesture may be identified by a number of characteristics stored, for example, in an entry in gesture library 1730 corresponding to the gesture. The received gesture may be associated with one or more commands stored, for example, in command library 1735. The commands may include operations to be performed by operating system 1712 or applications 1720 and 1722. For example, as illustrated in Figure 18B, the user input 1862 may include a gesture such as one or more fingers being swiped across the sensor array 1701 to form the shape of a “check mark” or the letter “V.” In one embodiment, this gesture may be associated with an “copy and paste” command that makes a copy of a previously selected object 1864 displayed by an application or the operating system and pastes 1866 the copy of the object into the displayed workspace 1870.

[0095] At block 1820, method 1800 activates a software-implemented keyboard. In one embodiment, the software-implemented keyboard may be a logical representation of physical or touch-screen keyboard 1706. The software-implemented keyboard may be stored in a memory of computing device 1700 and used to generate keyboard strings associated with various commands. In one implementation the software-implemented keyboard may comprise a filter driver configured to generate data inputs to the operating system (in response to a request from the gesture processing software) which are functionally equivalent to the data inputs created when a user. At block 1830, method 1800 may identify a corresponding command (e.g., from command library 1735) and associate the received user input 1862 with a keyboard string 1872 for the corresponding command. The keyboard string 1872 may include, for example, a sequence of one or more characters or function keys which may normally be entered by a user in a keyboard 1706. In the example mentioned above with respect to Figure 18B, where the identified command was the “copy and paste” command, there may be an associated keyboard string 1872. In one embodiment, the keyboard string may include the sequence of pressing the control (“CTRL”) key and the letter “C” followed by the control key again and the letter “V”. Thus, method 1800 may associate the “check mark” gesture with the keyboard string “CTRL C CTRL V” 1872.

[0096] At block 1840, method 1800 provides the keyboard string 1872 to the software-implemented keyboard driver. In one embodiment, this may be the same driver as keyboard driver 1716, however in other embodiments, it may be a separate driver. At block 1850, method 1800 instructs the operating system to perform the command associated with the keyboard string. In one embodiment, computing device 1700 may

enter the keyboard string (e.g., “CTRL C CTRL V”) using the software-implemented keyboard generated at block 1820. The entry of the keyboard string 1872 may cause a signal to be sent to operating system 1712 or applications 1720 and 1722 which may cause the corresponding command (e.g., the copy and paste command) to be executed or performed by the operating system 1712 or applications 1720 and 1722. As a result a selected object 1866 may be copied and pasted 1868 into the displayed workspace 1870 or other location. In another embodiment, the operating system 1712 may provide features making the software-implemented keyboard unnecessary. For example, sensor array driver 1710 may identify a received gesture 1862 and determine a command associated with that gesture. Sensor array driver 1710 may provide a signal to operating system 1712 or applications 1720, 1722 indicating that the associated command should be performed without entering a keyboard string 1872 using a software-implemented keyboard.

[0097] In another embodiment, the commands associated with different gestures may be dependent upon the context in which they are received. Depending on whether an application is currently active or whether only the operating system is running, or which of several different applications are active, certain gestures may be recognized and those gestures may have different associated commands. For example, the “check mark” gesture may only be recognized by certain application such as, applications 1720 and 1722, however operating system 1712 may not recognize the gesture if no applications are running. In addition, the “check mark” gesture may be associated with the “copy and paste” command when performed in application 1720, however, in application 1722, the gesture may have some other associated command (e.g., an undo command). Thus, the

gesture library 1730 and command library 1735 may have a context indication associated with certain entries and or may be divided into context-specific sections. In other embodiments, other factors may be considered to identify the proper context for a gesture, such as an identity of the user or a location of the gesture on the sensor array 1701.

[0098] Figure 19A is a flow diagram illustrating a gesture mapping method, according to an embodiment of the present invention. The method 1900 may be performed by processing logic that comprises hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (e.g., instructions run on a processing device to perform hardware simulation), or a combination thereof. The processing logic is configured to provide method for gesture mapping to associate a command with a received input gesture. In one embodiment, method 1900 may be performed by computing device 1700, as shown in Figure 17. Figure 19B is a diagram graphically illustrating the gesture mapping method 1900 of Figure 19A.

[0099] Referring to Figure 19A, at block 1910, method 1900 receives a first user input. In one embodiment, the first user input may include a gesture performed by a user on an input device, such as sensor array 1701. For example, as illustrated in Figure 19B, the gesture 1962 performed on sensor array 1701 may include a “back and forth” swipe with one or more fingers. The gesture 1962 may be identified by a number of characteristics stored, for example, in an entry in gesture library 1730 corresponding to the gesture. At block 1920, method 1900 compares the first user input to one or more entries in command library 1735. The received gesture 1962 may be associated with one or more commands stored, for example, in command library 1735. The commands may include operations to be performed by operating system 1712 or applications 1720 and

1722. Sensor array driver 1710 may identify a command associated with the received gesture from command library 1735 and at block 1930, method 1900 may perform the command associated with the first user input. For example, the gesture 1962 may be interpreted as the “copy and paste” command and the keyboard string “CTRL C CTRL V” 1972 may be entered. Performing the command may result, for example, in the execution of an action or function within operating system 1712 or applications 1720 and 1722. For example, a selected object 1966 may be copied and pasted 1968 into the displayed workspace 1971 or other location.

[00100] At block 1940, method 1900 receives a second user input. In certain embodiments, the second user input may include, for example, the same or a different gesture received at sensor array 1701, a keystroke or keyboard string received at keyboard 1706, the selection of an item in a user interface, such as an interface presented on display device 1708, or some other form of user input. In one embodiment, the second user input may be any indication that the command performed at block 1930 was not the command that the user intended or desired to be performed. For example, the second user input may include the keyboard string “CTRL Z” (which may implement an “undo” function) 1974, which may be entered by the user on keyboard 1706.

[00101] At block 1950, method 1900 may undo 1969 the command associated with the first user input that was performed at block 1930. In one embodiment, the operating system 1712 or application 1720 in which the command was performed may revert back to a state prior to the command being performed. In the example illustrated in Figure 19B, undoing the command 1969 may include removing the pasted copy 1968 of the selected object 1966. At block 1960, method 1900 may indicate the incorrect or outdated

association of the command with the first user input in the command library 1735. For example, sensor array driver 1710 may flag the entry in command library 1735 that associates a certain command with the gesture received as the first user input, remove the association, increment or decrement a counter, or otherwise indicate that the given command should not (or is less likely to) be performed in response to the received gesture in the future.

[00102] At block 1970, method 1900 receives a third user input indicating an intended or desired command to be associated with the first user input. The third user input may include, for example, a keystroke or keyboard string 1976 received at keyboard 1706, the selection of an item in a user interface, such as an interface presented on display device 1708, or some other form of user input. The third user input may actually perform the desired command or may indicate the desired command. In one embodiment, the keystroke 1976 may include the “Delete” key. The desired command may include placing the selected object 1966 in the Recycle Bin 1978 or Trash Can. At block 1980, method 1900 associates the command indicated by the third user input (i.e., the “Delete” key) at block 1970 with the gesture 1962 of the first user input received at block 1910. This may include, for example, linking an entry in gesture library 1730 with an entry in command library 1735 for the desired command, or otherwise associating the gesture and command. Thus, in the future, when the gesture 1962 is received as user input, the newly associated command (i.e., placing the object in the Recycle Bin) may be performed in response.

[00103] Figure 20A is a flow diagram illustrating a gesture mapping method, according to an embodiment of the present invention. The method 2000 may be

performed by processing logic that comprises hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (e.g., instructions run on a processing device to perform hardware simulation), or a combination thereof. The processing logic is configured to provide a method for gesture mapping to associate a command with a received input gesture. In one embodiment, method 2000 may be performed by computing device 1700, as shown in Figure 17. Figure 20B is a diagram graphically illustrating the gesture mapping method 2000 of Figure 20A.

[00104] Referring to Figure 20A, at block 2010, method 2000 receives a first user input. In one embodiment, the first user input may include a gesture performed by a user on an input device, such as sensor array 1701. The gesture may be identified by a number of characteristics stored, for example, in an entry in gesture library 1730 corresponding to the gesture. For example, as illustrated in Figure 20B, the gesture 2062 may include swiping one or more fingers in a “U” shaped motion across sensor array 1701. At block 2020, method 2000 compares the first user input to one or more entries in command library 1735. The received gesture may be associated with one or more commands stored, for example, in command library 1735. The commands may include operations to be performed by operating system 1712 or applications 1720 and 1722.

[00105] At block 2030, method 2000 determines if the gesture is recognized in the library 1735 and associated with a certain command. If so, at block 2040, method 2000 performs the command associated with the gesture. If at block 2030, method 2000 determines that the gesture is not already associated with a command, at block 2050, method 2000 may provide an interface 2072 with a list of one or more available commands. In one embodiment, the interface may be provided as a graphical user

interface displayed on a display device, such as display device 1708. In the example illustrated in Figure 20B, interface 2072 may include the following commands: (1) Delete; (2) Copy and Paste; (3) Rotate 90°; (4) Rotate 180°; and (5) Save.

[00106] At block 2060, method 2000 may receive a second user input indicating a desired command. In one embodiment, the interface may include all known commands or a selectively chosen subset of commands, from which the user may select a desired command. In another embodiment, the user may input the desired command into a designated field in the user interface or simply perform the command (e.g., via a keystroke or keyboard string). In one embodiment, for example, the second user input may include a keystroke 2074 including a number key (e.g., “3”) associated with one of the listed commands (e.g., Rotate 90°). The command may rotate a selected object 2066 by 90 degrees. At block 2070, method 2000 may associate the command indicated by the second user input 2074 at block 2060 with the gesture 2062 received as the first user input at block 2010. This may include, for example, linking an entry in gesture library 1730 with an entry in command library 1735 for the desired command, or otherwise associating the gesture 2062 and command.

[00107] Figure 20C is a flow diagram illustrating a gesture mapping method, according to an embodiment of the present invention. The method 2005 may be performed by processing logic that comprises hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (e.g., instructions run on a processing device to perform hardware simulation), or a combination thereof. The processing logic is configured to provide a method for gesture mapping to associate a command with a received input gesture. In one embodiment, method 2005 may be performed by

computing device 1700, as shown in Figure 17. Figure 20D is a diagram graphically illustrating the gesture mapping method 2005 of Figure 20C.

[00108] Referring to Figure 20C, at block 2015, method 2005 receives a first user input. In one embodiment, the first user input may include a gesture performed by a user on an input device, such as sensor array 1701. The gesture may be identified by a number of characteristics stored, for example, in an entry in gesture library 1730 corresponding to the gesture. For example, as illustrated in Figure 20D, gesture 2063 may include a swiping motion on the sensor array 1701 that is similar to a “check mark” gesture, but not exactly right. At block 2025, method 2005 compares the first user input to one or more entries in command library 1735. The received gesture 2063 may be associated with one or more commands stored, for example, in command library 1735. The commands may include operations to be performed by operating system 1712 or applications 1720 and 1722.

[00109] At block 2035, method 2005 determines if the gesture 2063 is recognized in the library 1735 and associated with a certain command. If so, at block 2045, method 2005 performs the command associated with the gesture 2063. If at block 2035, method 2005 determines that the gesture 2063 is not already associated with a command, at block 2055, method 2005 identifies a likely command from the library based on the gesture characteristics. Since the gesture 2063 was not exactly the same as of a recognized gesture, the gesture 2063 may not be recognized. If, however, the characteristics of the gesture 2063 are similar to the characteristics of a recognized gesture, or within in a certain defined tolerance of allowed characteristics (e.g., as illustrated by gesture 2065), method 2005 may make an “educated guess” (i.e. infer that the user intended to make a

gesture with characteristics which are similar to the motion detected) based on the commands that are associated with other similar gestures as to what command is most likely to be associated with the gesture 2063 received as the first and second user inputs. At block 2065, method 2005 associates the command with the gestures and performs the newly associated command. In one embodiment, performing the command may include copying a selected object 2078 and pasting 2080 the copy into the displayed workspace or other location.

[00110] Figure 21A is a flow diagram illustrating a method for user creatable gestures, according to an embodiment of the present invention. The method 2100 may be performed by processing logic that comprises hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (e.g., instructions run on a processing device to perform hardware simulation), or a combination thereof. The processing logic is configured to provide method for implementing a new gesture and an associated command in a computing system. In one embodiment, method 2100 may be performed by computing device 1700, as shown in Figure 17. Figure 21B is a diagram graphically illustrating the gesture mapping method 2100 of Figure 21A.

[00111] Referring to Figure 21A, at block 2110, method 2100 receives a first user input. In one embodiment, the first user input may include a gesture performed by a user on an input device, such as sensor array 1701. The gesture may be identified by a number of characteristics stored, for example, in an entry in gesture library 1730 corresponding to the gesture. For example, as illustrated in Figure 21B, gesture 2162 may include a swiping motion on the sensor array 1701 that is similar to a “check mark” gesture, but not exactly right. At block 2120, method 2100 compares the first user input to one or more

entries in command library 1735. The received gesture 2162 may be associated with one or more commands stored, for example, in command library 1735. The commands may include operations to be performed by operating system 1712 or applications 1720 and 1722.

[00112] At block 2130, method 2100 determines if the gesture 2162 is recognized in the library 1735 and associated with a certain command. If so, at block 2140, method 2100 performs the command associated with the gesture 2162. If at block 2130, method 2100 determines that the gesture 2162 is not already associated with a command, at block 2150, method 2100 receives a second user input. Since the first gesture 2162 was not exactly the same as (or within a certain tolerance) of a recognized gesture, the gesture may be repeated 2164, as a second user input. In one embodiment, this second user input is the same gesture that was received as the first user input at block 2110. The second user input may be similarly received by sensor array 1701. For example, gesture 2164 may be a more accurate “check mark” gesture.

[00113] At block 2160, method 2100 compares the first and second user inputs to the command library 1735. In one embodiment, this may include identifying characteristics of the gestures 2162 and 2164, such as a number of contacts, the position of those contacts, relative and absolute motion of the contacts, or other characteristics and comparing the identified characteristics to characteristics of the commands stored in command library 1735. At block 2170, method 2100 identifies a likely command from the library based on the gesture characteristics. Method 2100 may make an “educated guess” based on the commands that are associated with other similar gestures as to what command is most likely to be associated with the gesture received as the first and second

user inputs. At block 2180, method 2100 associates the command with the gestures and performs the newly associated command. In one embodiment, method 2100 may adjust the characteristics of the “Copy and Paste” command to include slight variations 2166 in the gestures associated with the command. This adjustment may allow either gesture 2162 or gesture 2164 to be recognized as the gesture 2166 associated with the command in the future. Performing the command may include copying a selected object 2168 and pasting 2169 the copy into the displayed workspace or other location.

[00114] Figure 22A is a flow diagram illustrating a method for user creatable gestures, according to an embodiment of the present invention. The method 2200 may be performed by processing logic that comprises hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (e.g., instructions run on a processing device to perform hardware simulation), or a combination thereof. The processing logic is configured to provide method for implementing a new gesture and an associated command in a computing system. In one embodiment, method 2200 may be performed by computing device 1700, as shown in Figure 17. Figure 22B is a diagram graphically illustrating the gesture mapping method 2200 of Figure 22A.

[00115] Referring to Figure 22A, at block 2210, method 2200 initializes gesture recording. In one embodiment, the user may select (e.g., through a user interface displayed on display device 1708) gesture recording. Gesture recording may include receiving a user input on a touch pad, where the gesture is to be added to a gesture library 1730 storing saved gestures. For example, as illustrated in Figure 22B, gesture recording may be initialized by a keyboard string 2262 entered on keyboard 1706. In one embodiment, the keyboard string is “CTRL R”. At block 2220, method 2200 receives a

first user input. In one embodiment, the first user input may include a gesture performed by a user on an input device, such as sensor array 1701. The gesture may be identified by a number of characteristics stored, for example, in an entry in gesture library 1730 corresponding to the gesture. For example, as illustrated in Figure 22B, the gesture 2264 performed on sensor array 1701 may include a “back and forth” swipe with one or more fingers. At block 2230, method 2200 compares the first user input to one or more entries in gesture library 1730 and command library 1735. The received gesture 2264 may be associated with one or more commands stored, for example, in command library 1735. The commands may include operations to be performed by operating system 1712 or applications 1720 and 1722.

[00116] At block 2240, method 2200 determines if the gesture 2264 is recognized in the gesture library 1730 and associated with a certain command in command library 1735. If so, at block 2250, method 2200 performs the command associated with the gesture 2264. If at block 2240, method 2200 determines that the gesture 2264 is not known in gesture library 1730 or already associated with a command, at block 2260, method 2200 stores the received gesture 2264 in the gesture library 1730. In one embodiment, method 2200 creates an entry for the received gesture 2264 in library 1730 and identifies the gesture 2264 according to one or more characteristics of the gesture, as described above.

[00117] At block 2270, method 2200 may receive a second user input indicating a desired command. In one embodiment, the interface may include all known commands or a selectively chosen subset of commands, from which the user may select a desired command. In another embodiment, the user may input the desired command into a

designated field in the user interface or simply perform the command (e.g., via a keystroke or keyboard string). In one embodiment, for example, the user may enter a keystroke 2266 including the “Delete” key on keyboard 1706. At block 2280, method 2200 may associated the command indicated at block 2270 with the gesture 2266 received as the first user input at block 2220. This may include, for example, linking an entry in gesture library 1730 with an entry in command library 1735 for the desired command, or otherwise associating the gesture and command. In one embodiment, the “Delete” command may include placing a selected object 2072 in the Recycle Bin 2074 or Trash Can.

[00118] Although the present invention has been described with reference to specific example embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of the invention as set forth in the claims. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

[00119] In the foregoing specification, the invention has been described with reference to specific example embodiments thereof. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A system comprising:
a sensor configured to detect user inputs; and
a processor configured to:
receive a detected first user input from the sensor;
receive a detected second user input from the sensor; and
assign a command to the first user input based on the second user input.
2. The system of claim 1, wherein the first user input comprises a gesture.
3. The system of claim 2, further comprising a memory configured to store a library of commands, wherein the processor is further configured to determine if the first user input has an associated command in the library.
4. The system of claim 3, wherein if the first user input has an associated command in the library, the processor is further configured to execute the associated command.
5. The system of claim 4, wherein the second user input comprises an indication that the associated command is incorrect, and wherein the processor is further configured to indicate in the library that the associated command should not be associated with the first user input.

6. The system of claim 3, wherein the second user input comprises the gesture, and wherein the processor is further configured to identify the command based on a plurality of characteristics of the gesture.
7. The system of claim 1, wherein assigning a command comprises:
generating a software-implemented keyboard; and
associating the first user input with a string input to the software-implemented keyboard
8. A system comprising:
a sensing device configured to determine one or more characteristics of at least one of a plurality of user inputs; and
a processor configured to:
receive the determined one or more characteristics;
identify at least one of a plurality of commands based on the determined one or more characteristics; and
assign the at least one of the plurality of commands to the at least one of the plurality of user inputs.
9. The system of claim 8, wherein the determined one or more characteristics comprises a gesture performed by the at least one of the plurality of user inputs.

10. The system of claim 9, wherein the determined one or more characteristics is received during a gesture recording period.

11. The system of claim 9, further comprising a memory configured to store a library of the plurality of commands, wherein identifying the at least one of the plurality of commands comprises identifying the at least one from the library based on the determined one or more characteristics.

12. The system of claim 8, further comprising:
a display device configured to display a graphical user interface, wherein the processor is further configured to present, in the graphical user interface, a list of available commands.

13. The system of claim 12, wherein identifying the at least one of the plurality of commands comprises receiving a second user input indicating a command from the list of available commands to be associated with the determined characteristic.

14. The system of claim 11, wherein the processor is further configured to store the at least one of the plurality of commands and the at least one of the plurality of user inputs in the library.

15. A method, comprising:
receiving a first user input detected by a sensor;

identifying one or more characteristics of the received first user input;
determining, by a processor, if the one or more characteristics matches a characteristic of a known gesture; and
if the one or more characteristics does not match a characteristic of a known gesture, generating a new gesture based on the one or more characteristics of the first user input.

16. The method of claim 15, further comprising:

receiving a second user input detected by the sensor; and
determining a command associated with the first user input based on the second user input.

17. The method of claim 15, further comprising:

determining a command associated with the first user input based on the identified one or more characteristics of the first user input.

18. The method of claim 15, wherein the identified one or more characteristics comprises one of a number of contacts with the sensor, a position of the contacts on the sensor, a relative motion of the contacts, and an absolute motion of the contacts on the sensor.

19. The method of claim 15, further comprising:

displaying on a display device, a graphical user interface including a list of available commands to be associated with the first user input.

20. The method of claim 15, wherein the first user input is received during a gesture recording period.

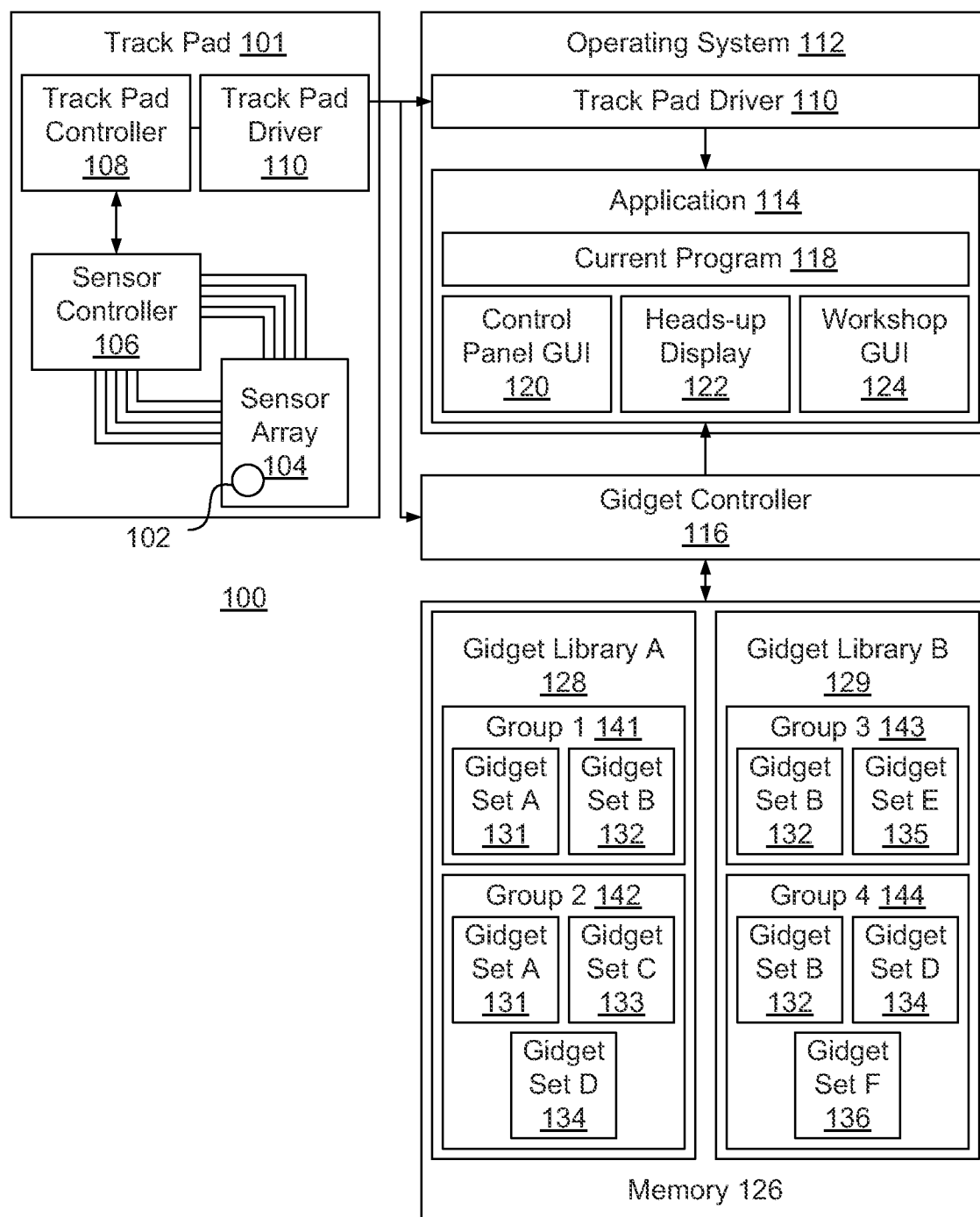
CD09033CIP1
sheet 1/30

Figure 1

200

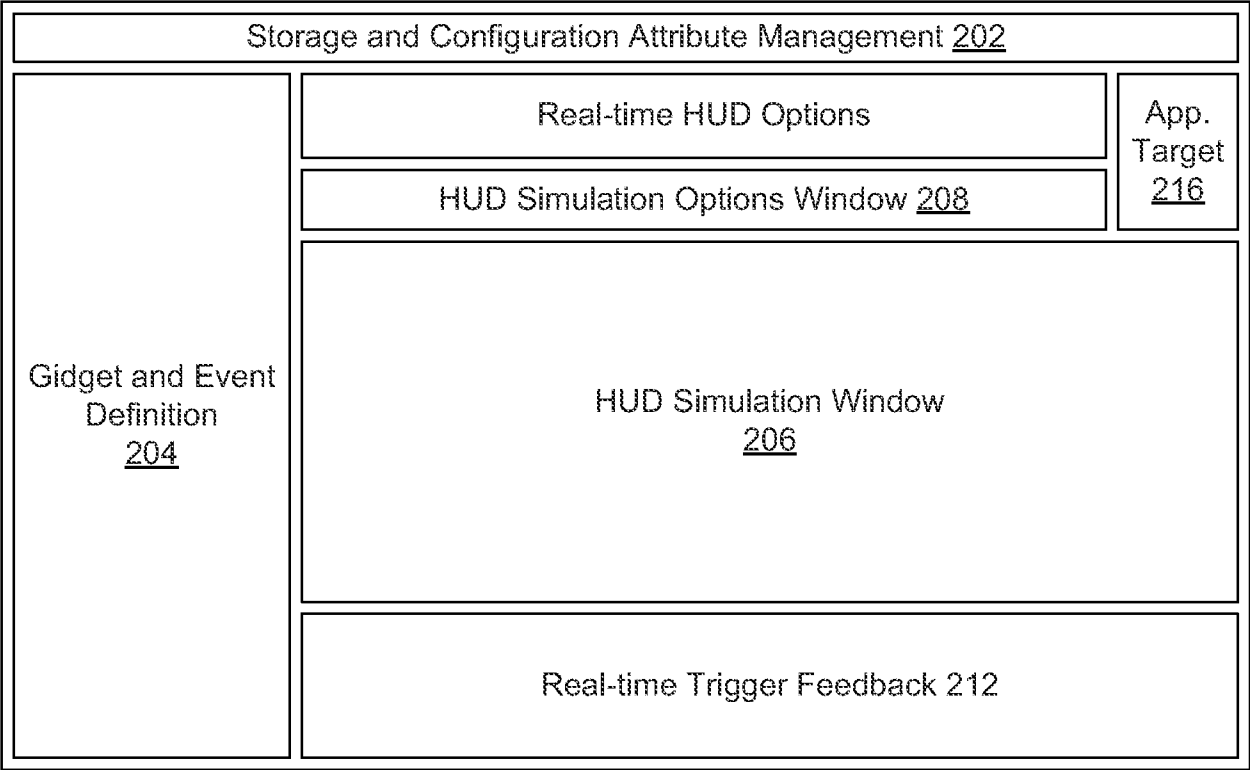


Figure 2

CD09033CIP1
sheet 3/30

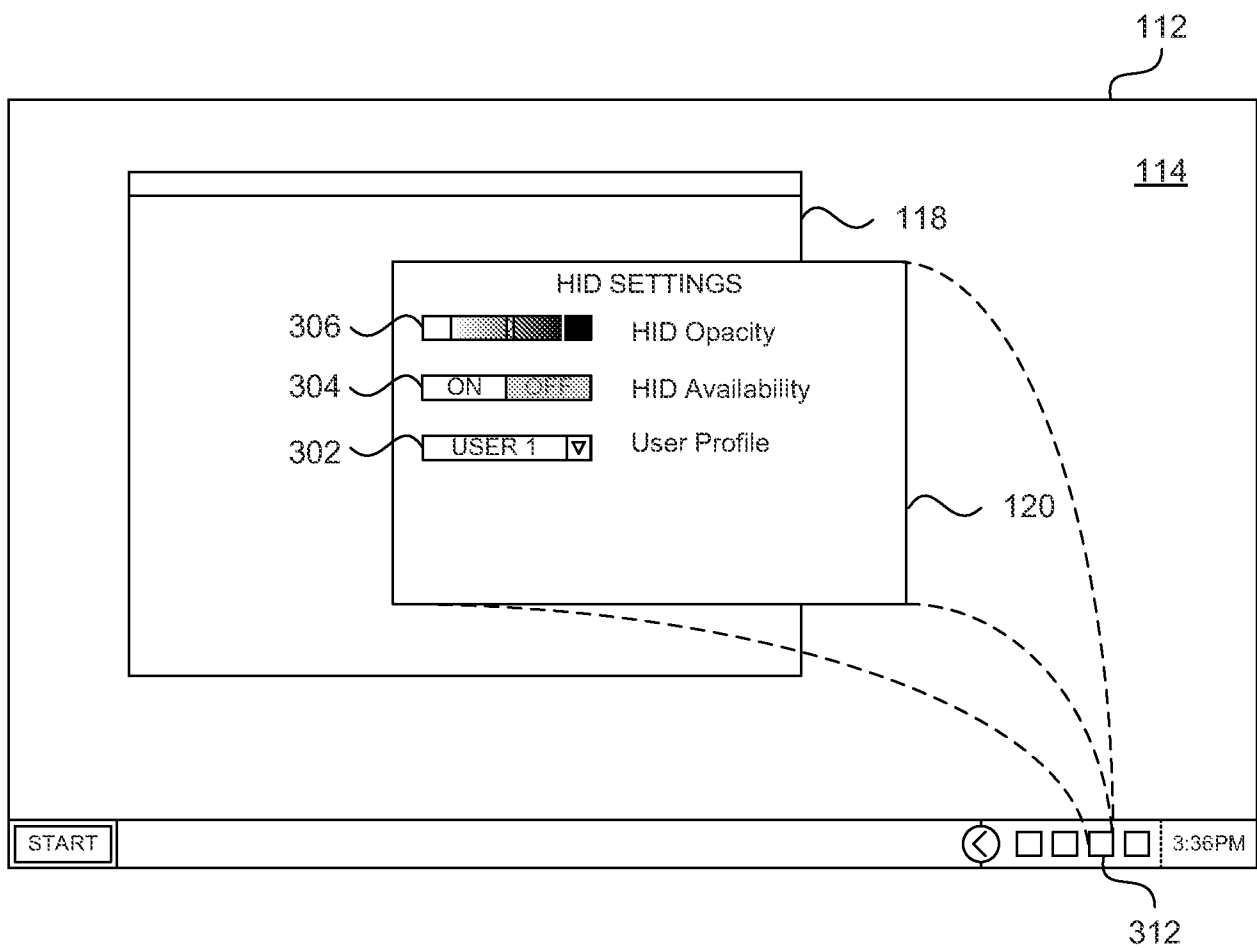


Figure 3

CD09033CIP1
sheet 4/30

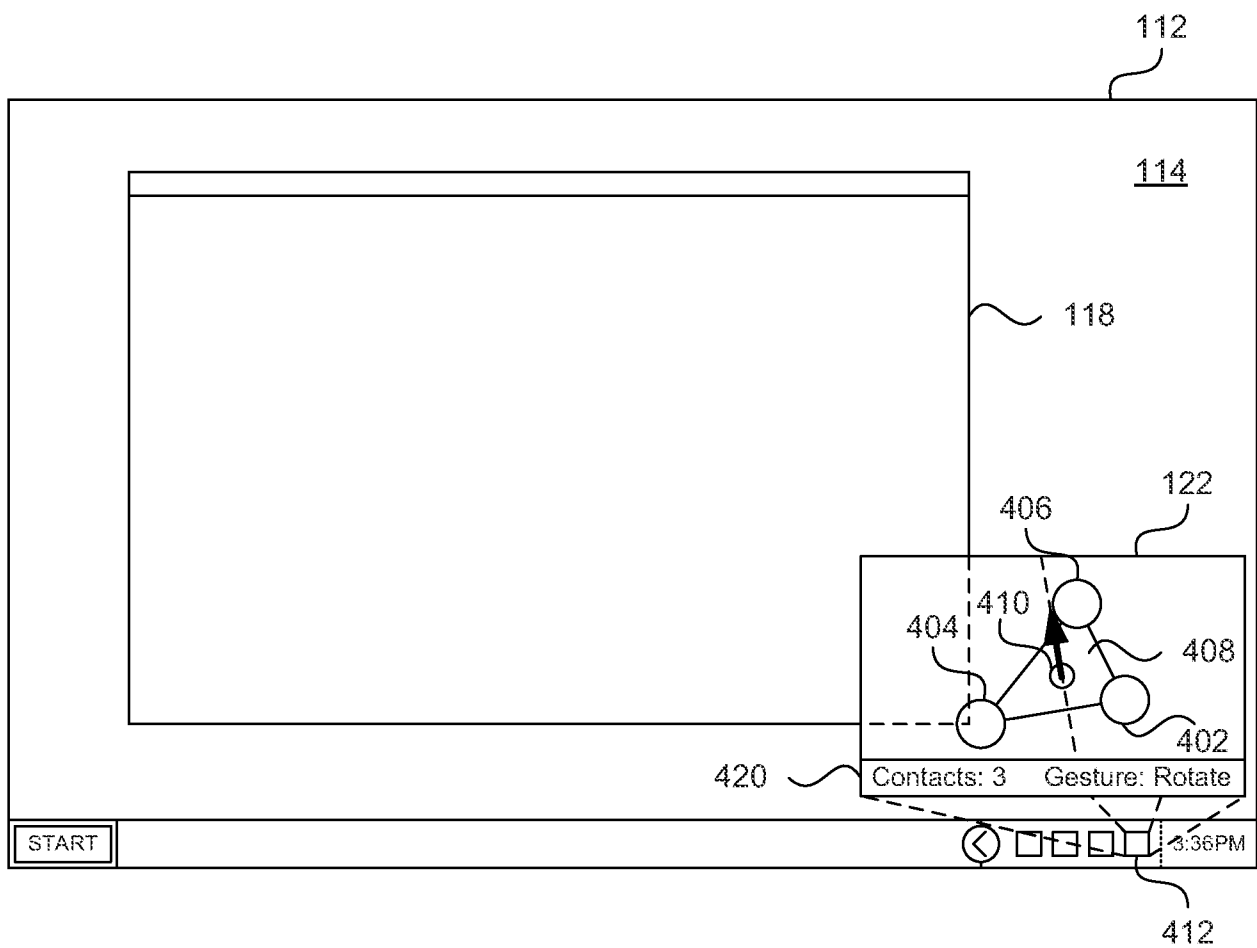


Figure 4

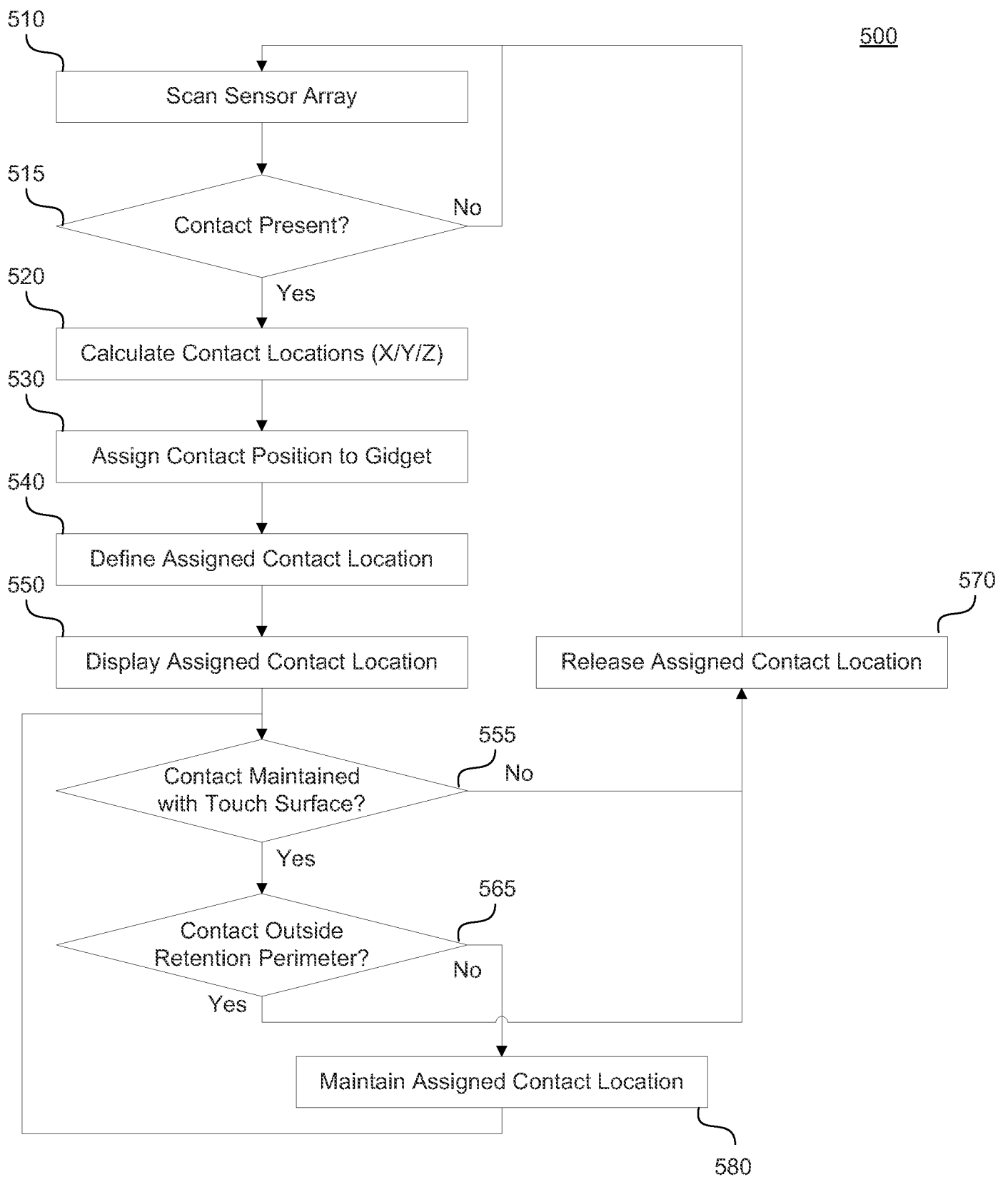
CD09033CIP1
sheet 5/30

Figure 5

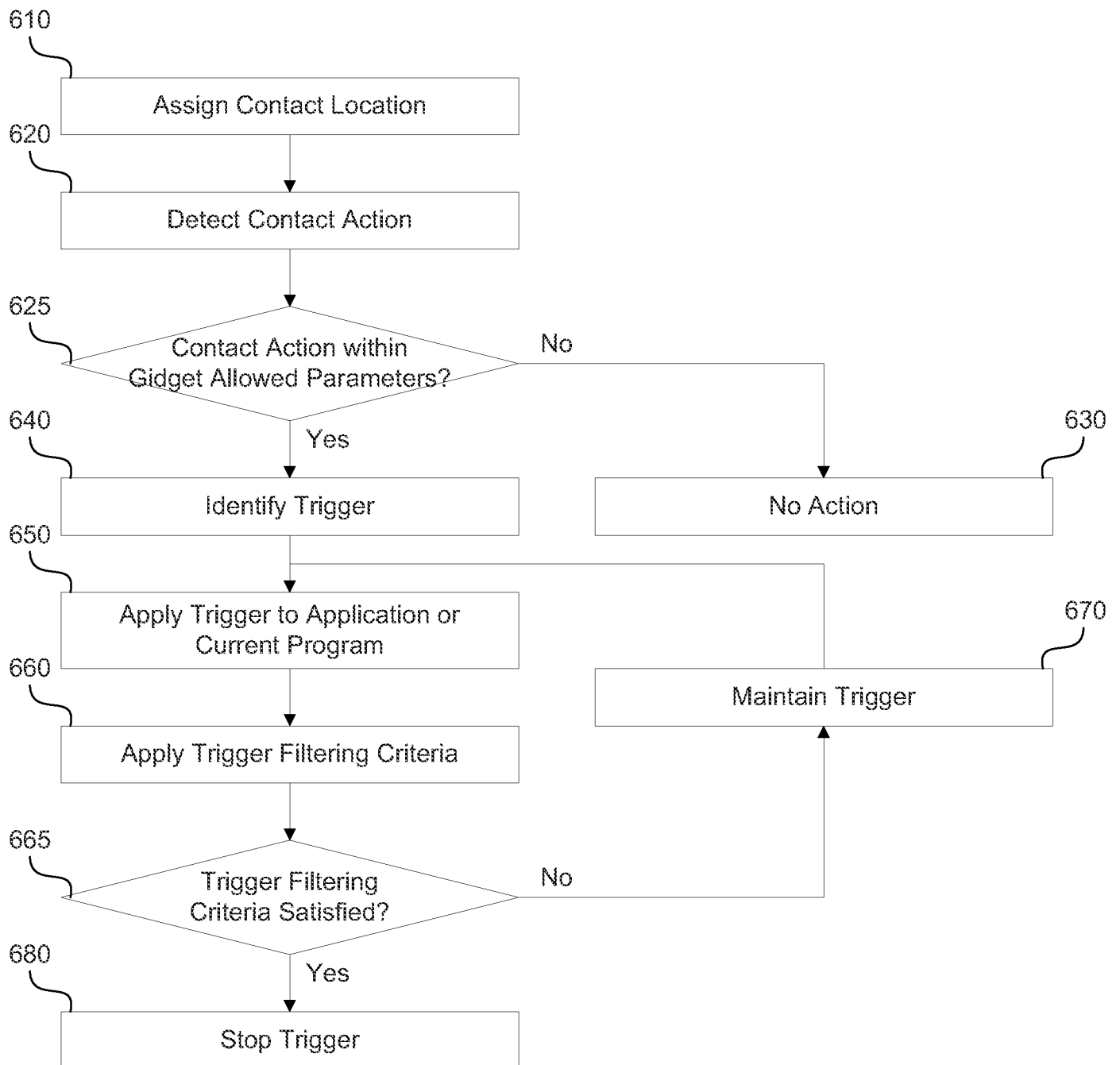


Figure 6

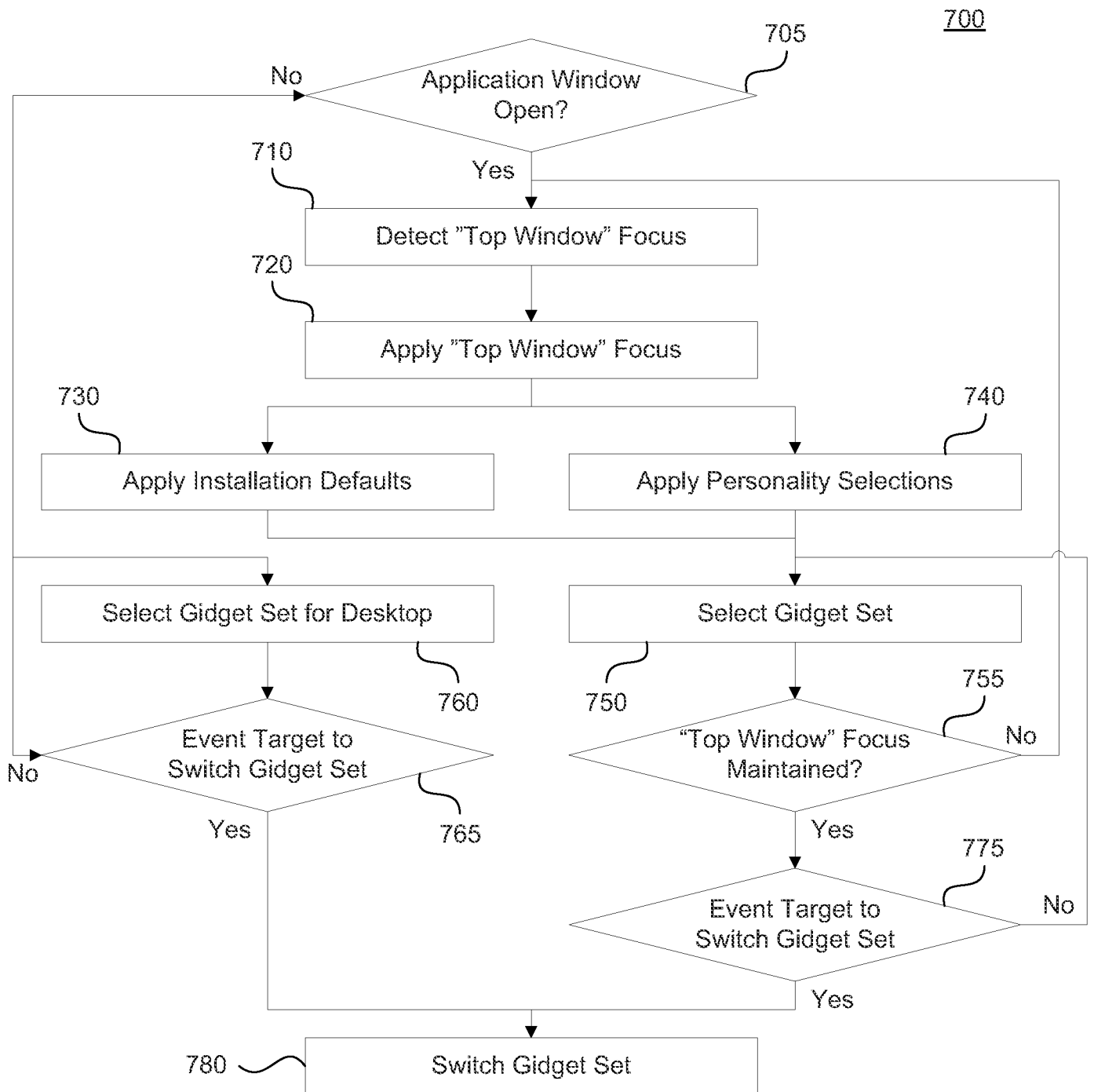
CD09033CIP1
sheet 7/30

Figure 7

CD09033CIP1
sheet 8/30

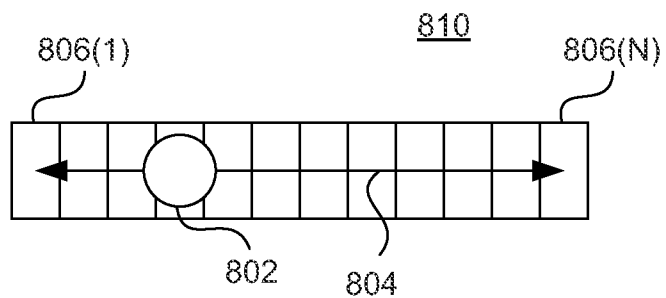


Figure 8A

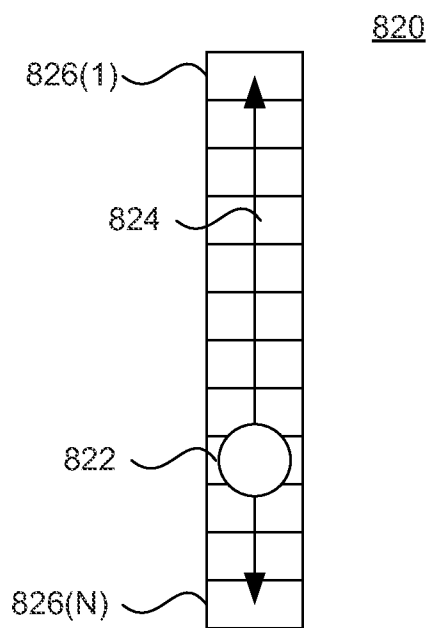


Figure 8B

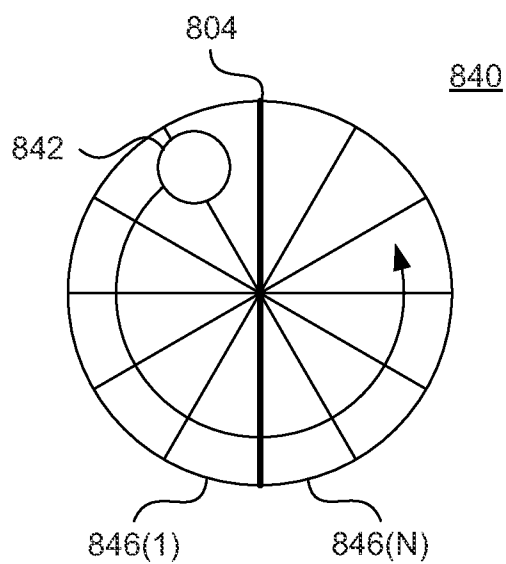


Figure 8C

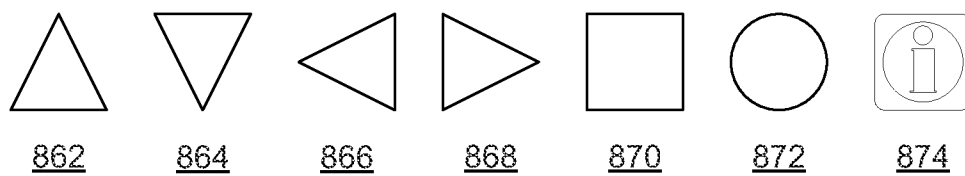


Figure 8D



Figure 8E

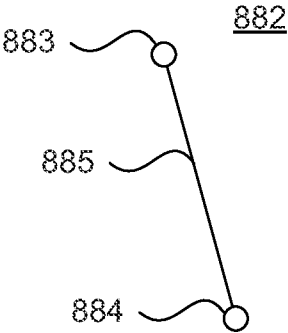


Figure 8F

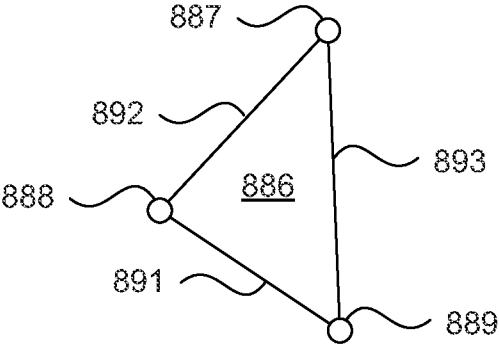


Figure 8G

CD09033CIP1
sheet 10/30

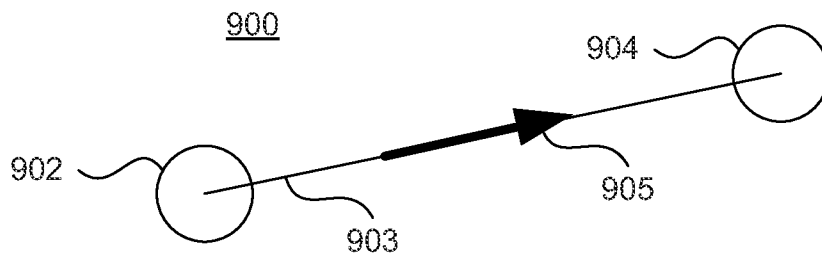


Figure 9A

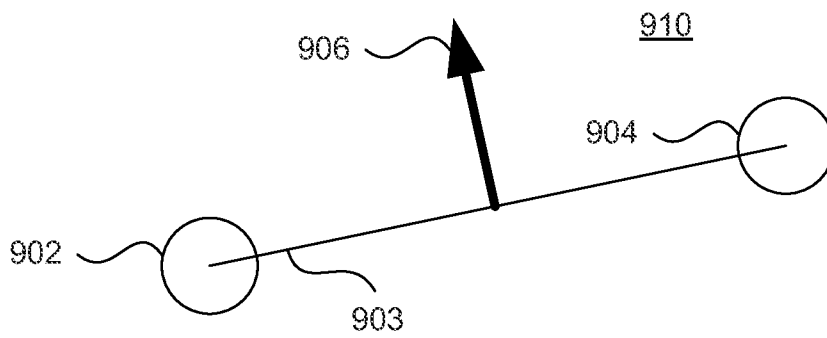


Figure 9B

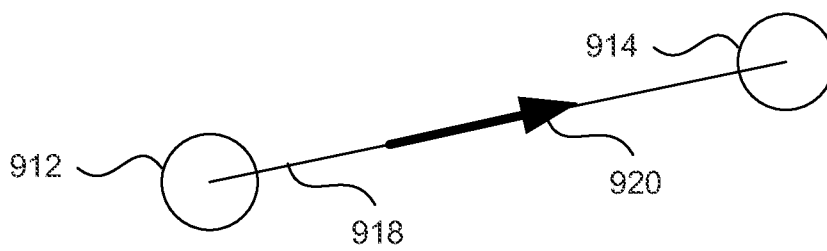


Figure 9C

CD09033CIP1
sheet 11/30

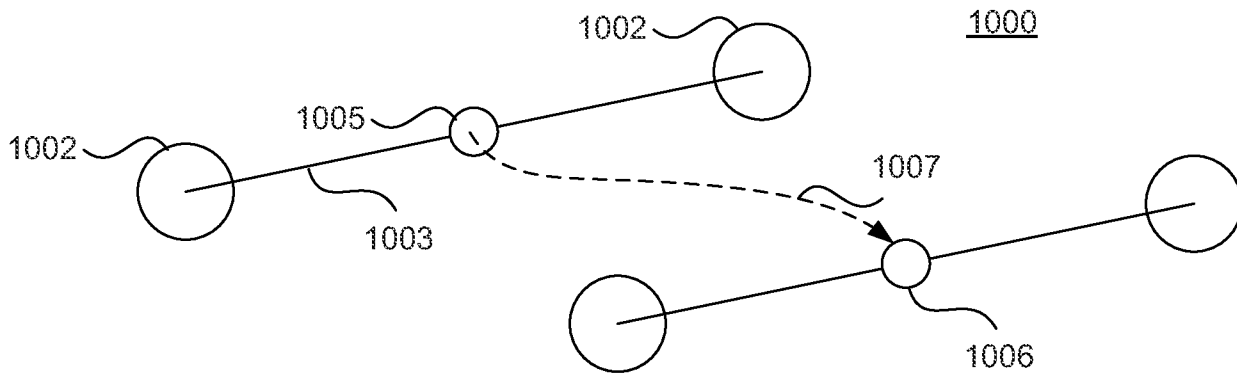


Figure 10A

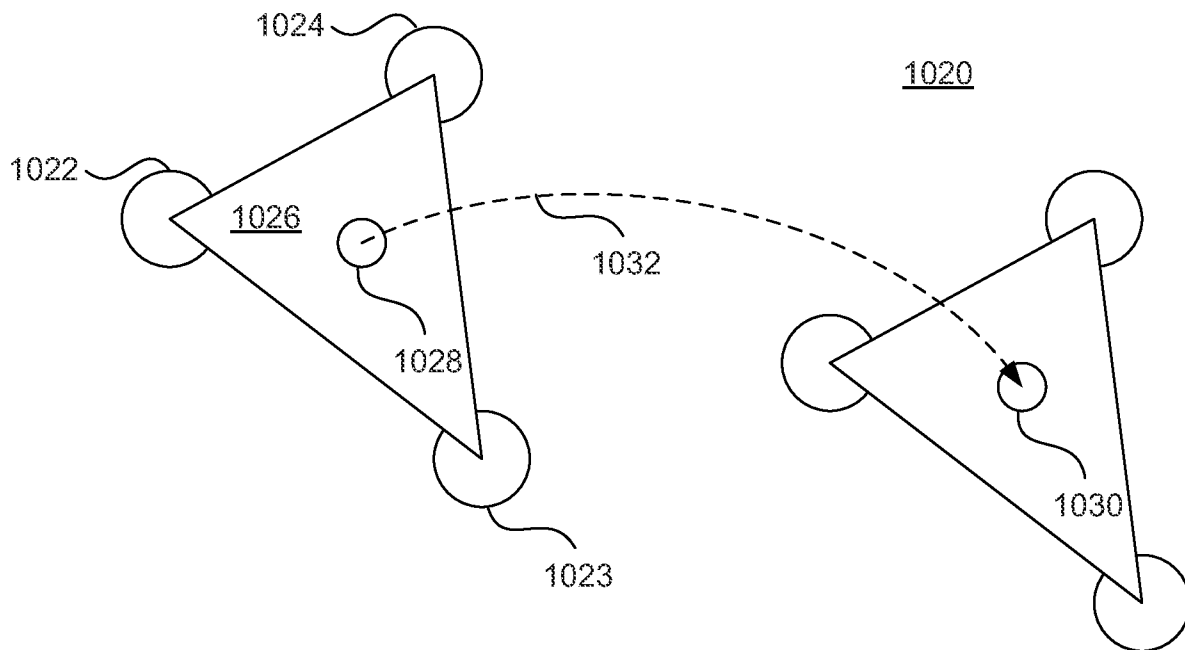


Figure 10B

CD09033CIP1
sheet 12/30

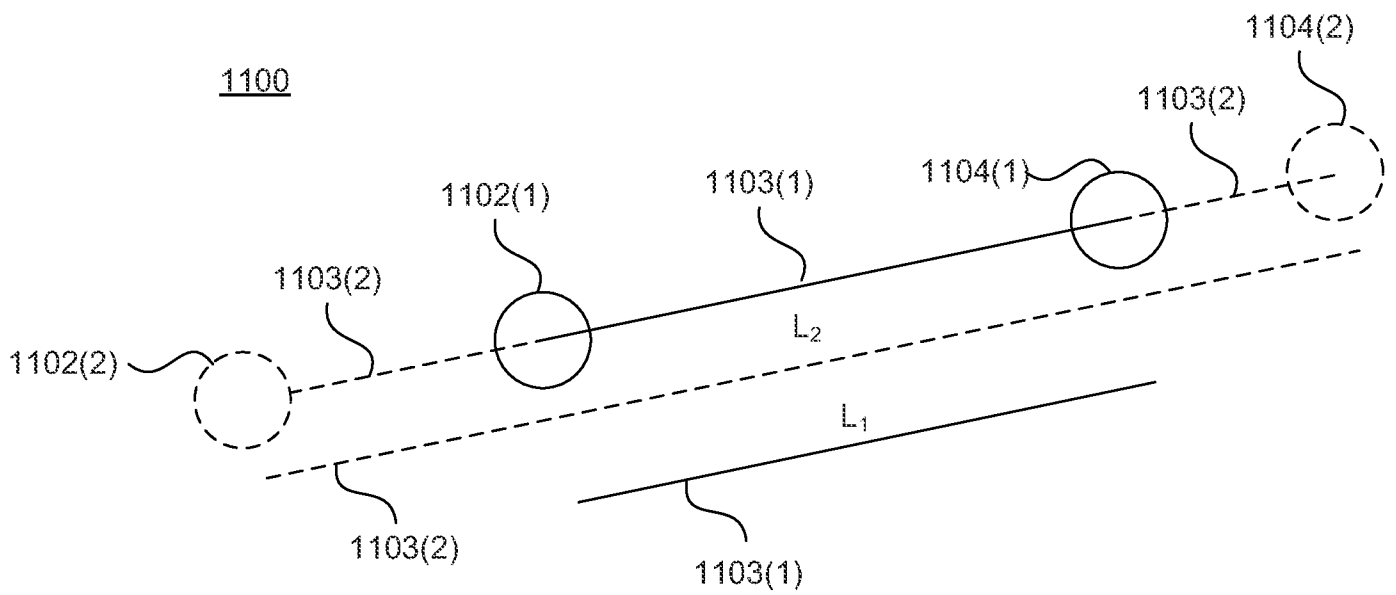


Figure 11A

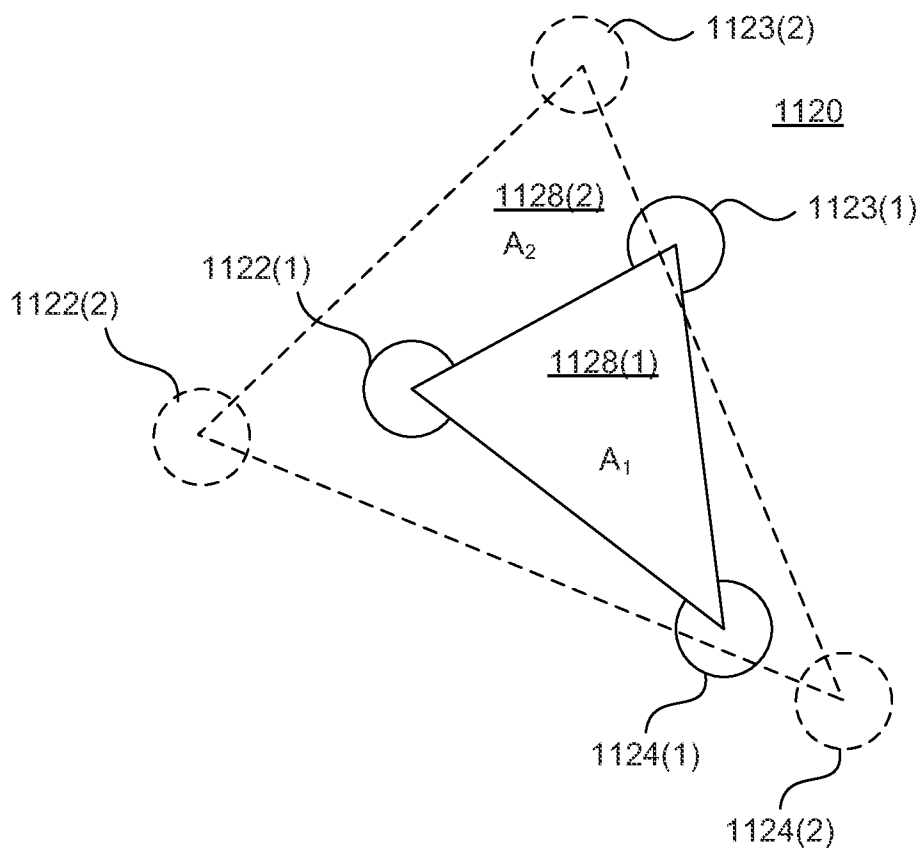


Figure 11B

CD09033CIP1
sheet 13/30

1200

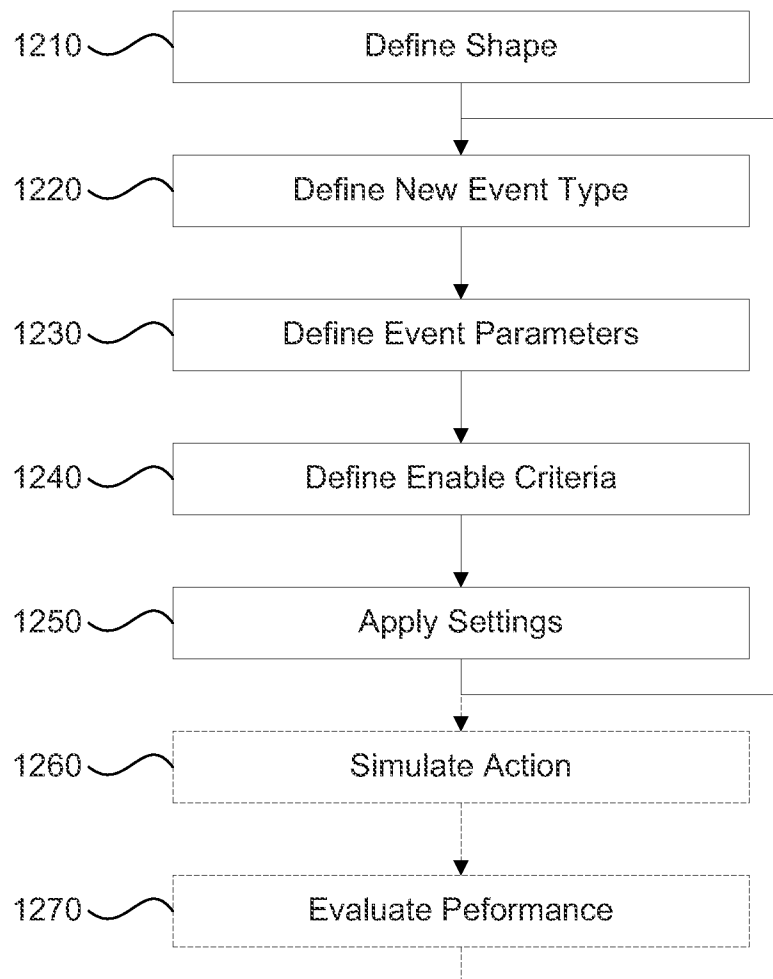


Figure 12

1300

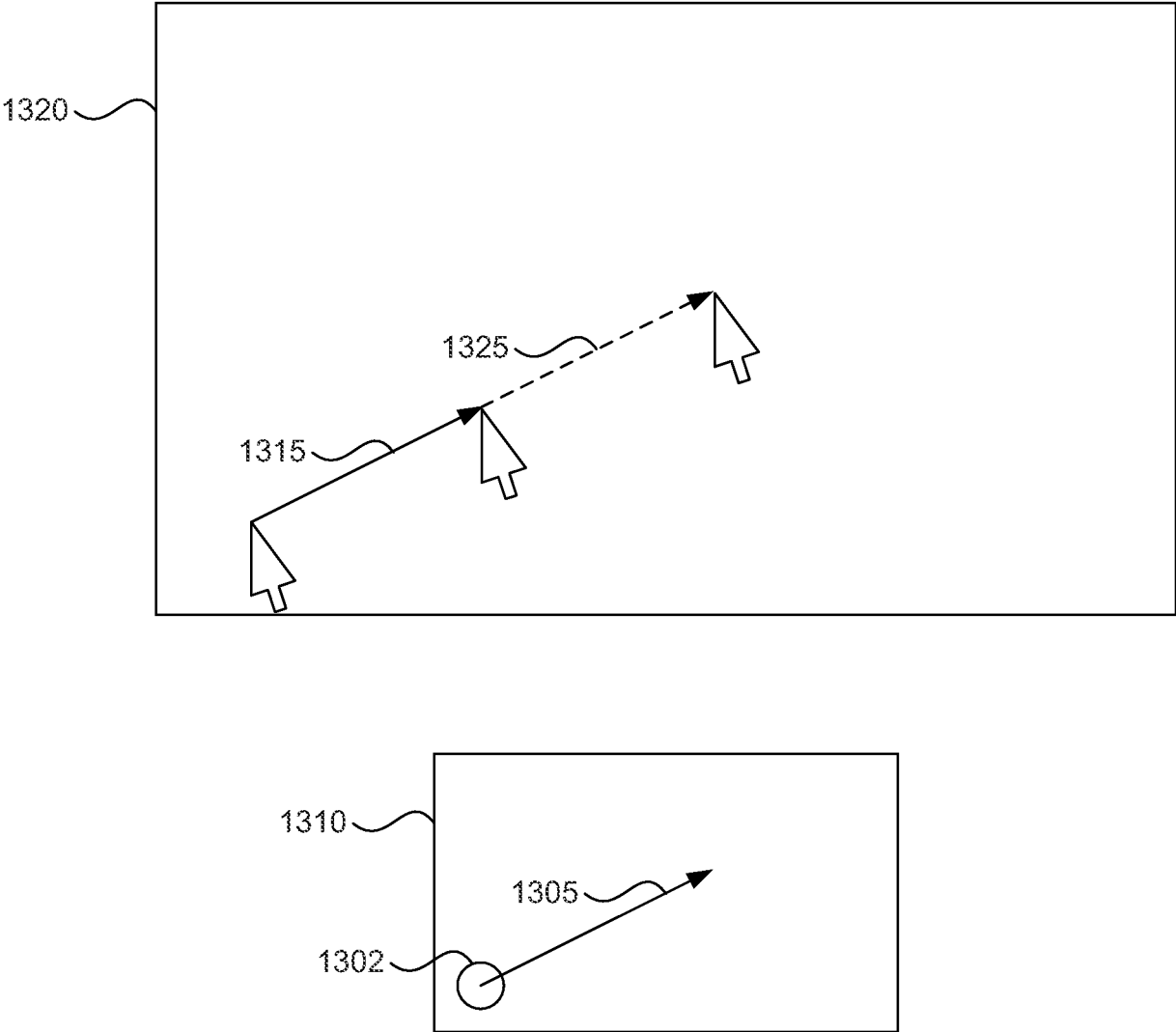


Figure 13

CD09033CIP1
sheet 15/30

1400

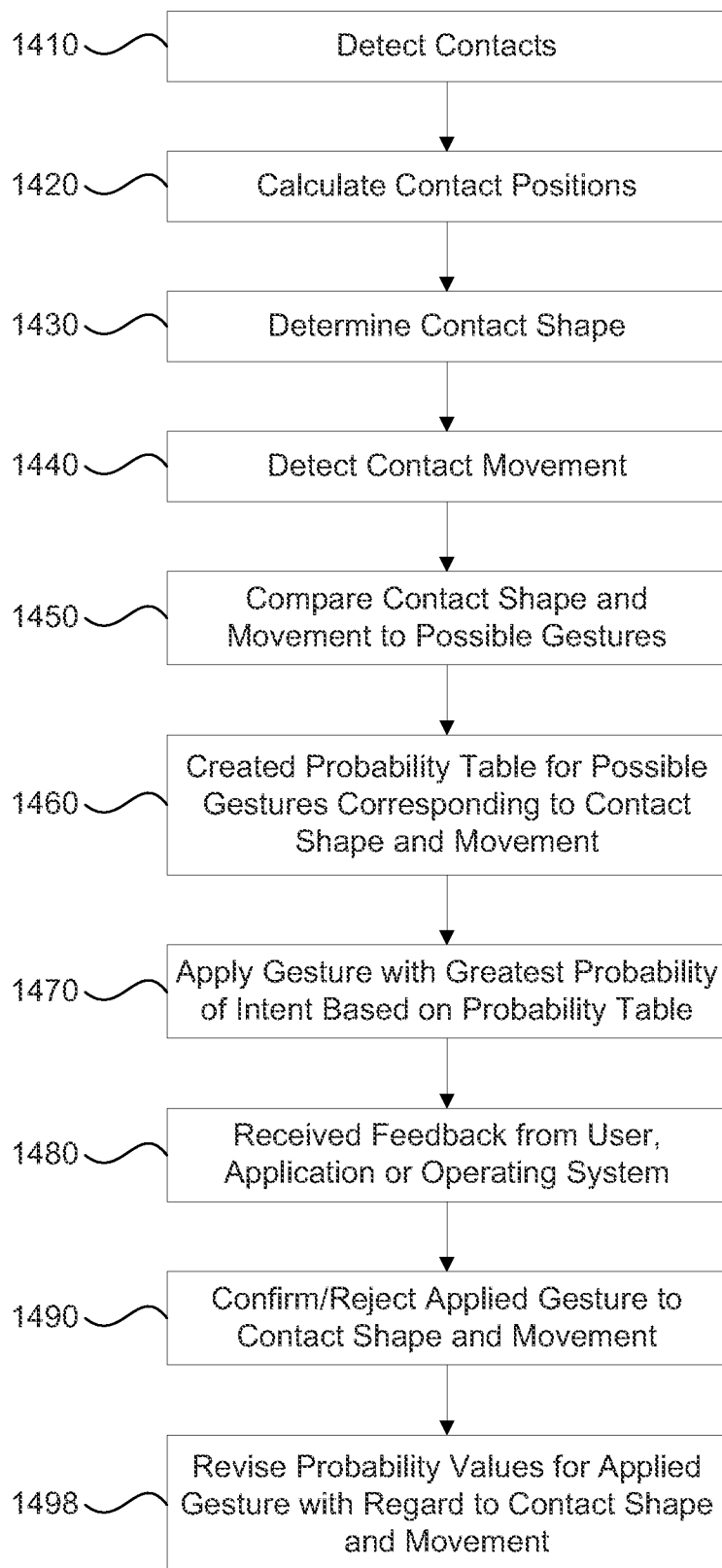


Figure 14

CD09033CIP1
sheet 16/30

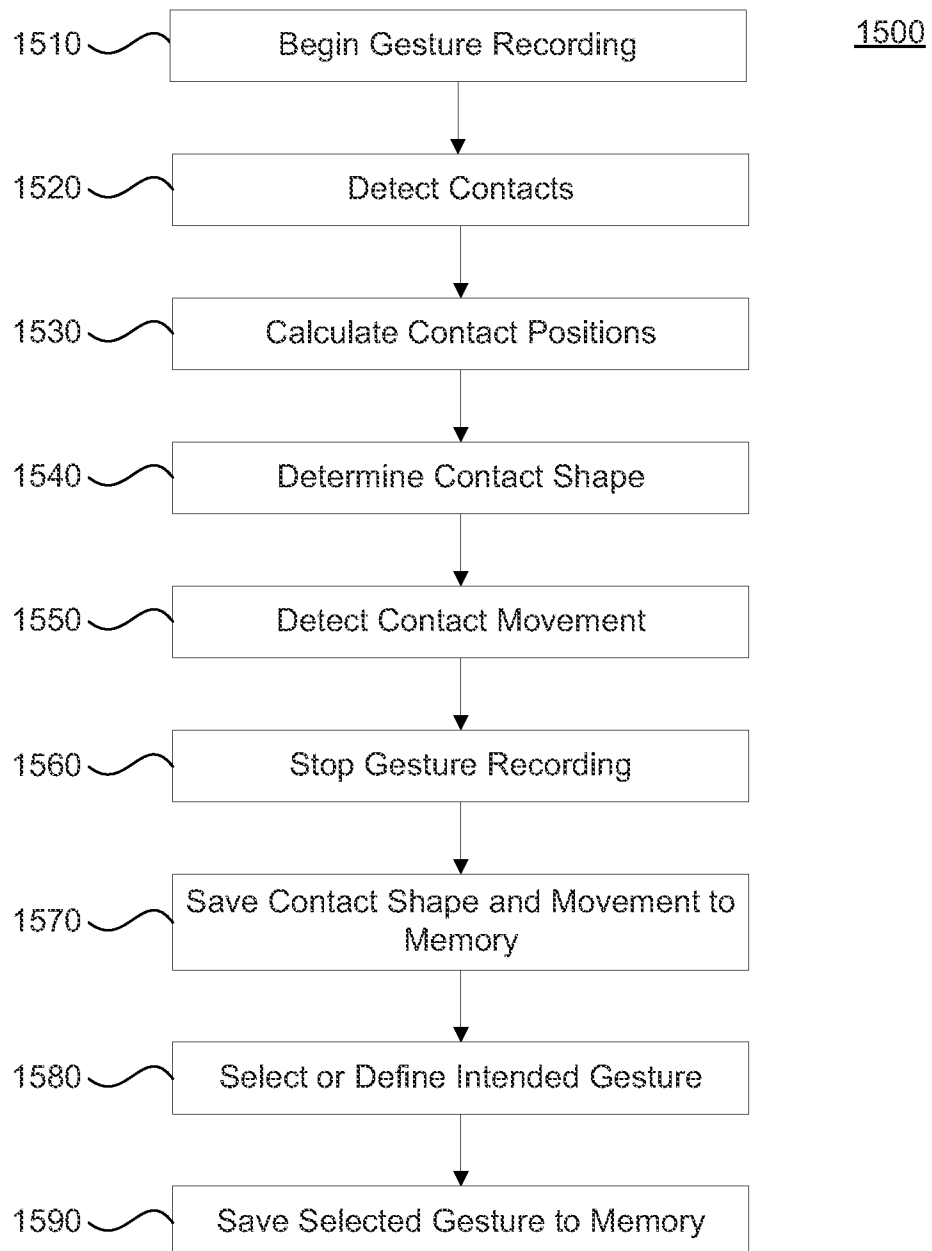


Figure 15

CD09033CIP1
sheet 17/30

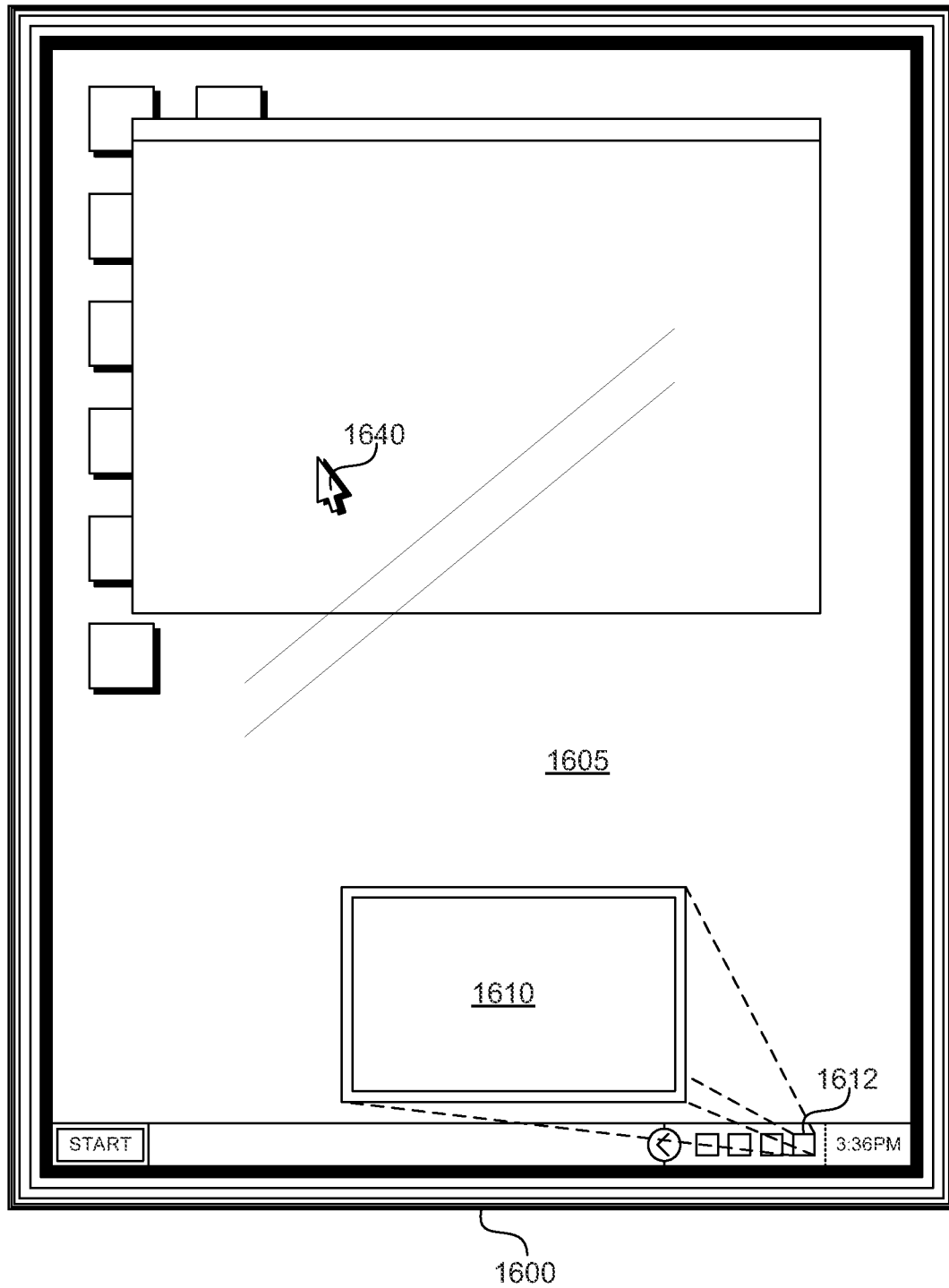


Figure 16

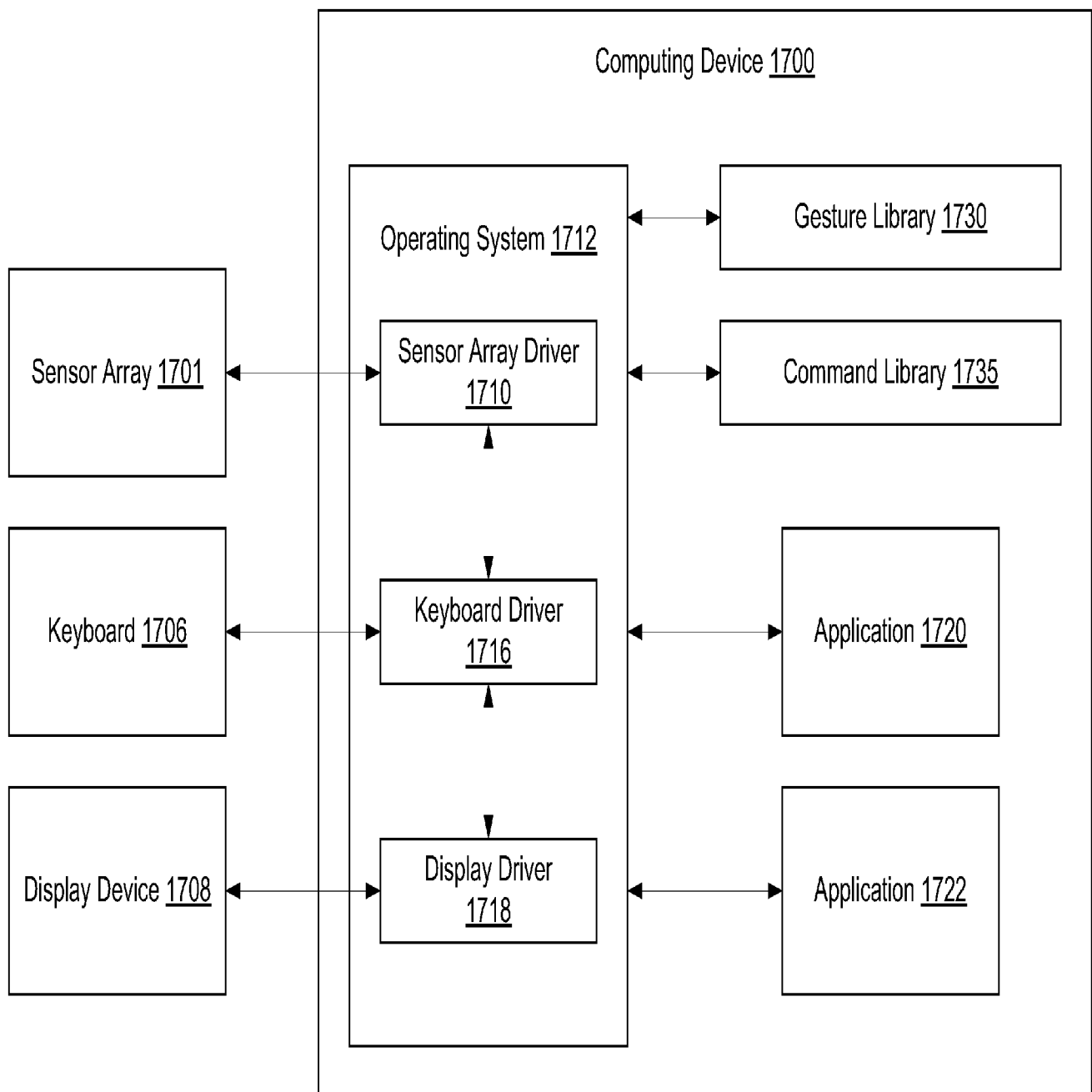
CD09033CIP1
sheet 18/30

Figure 17

CD09033CIP1
sheet 19/30

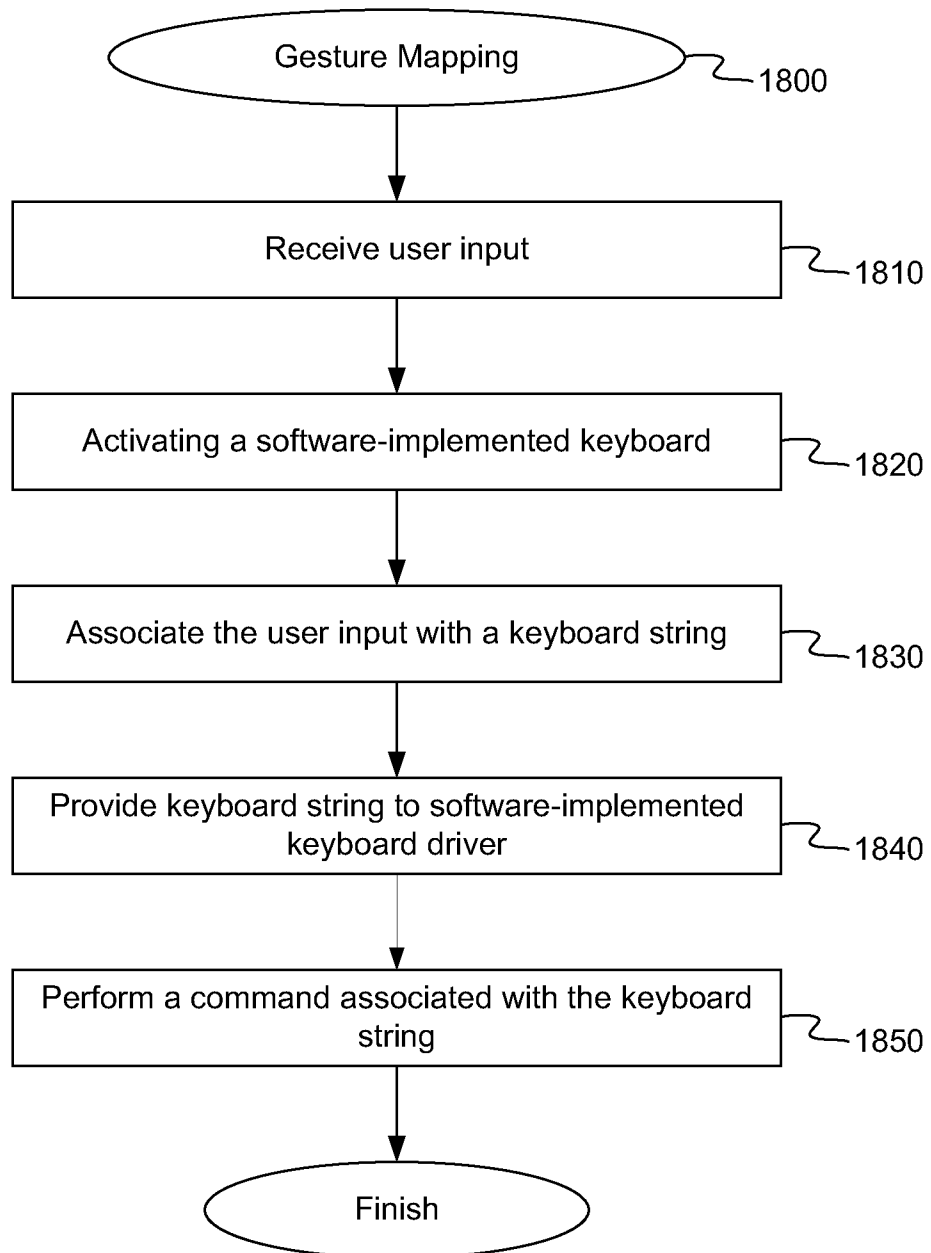


Figure 18A

CD09033CIP1
sheet 20/30

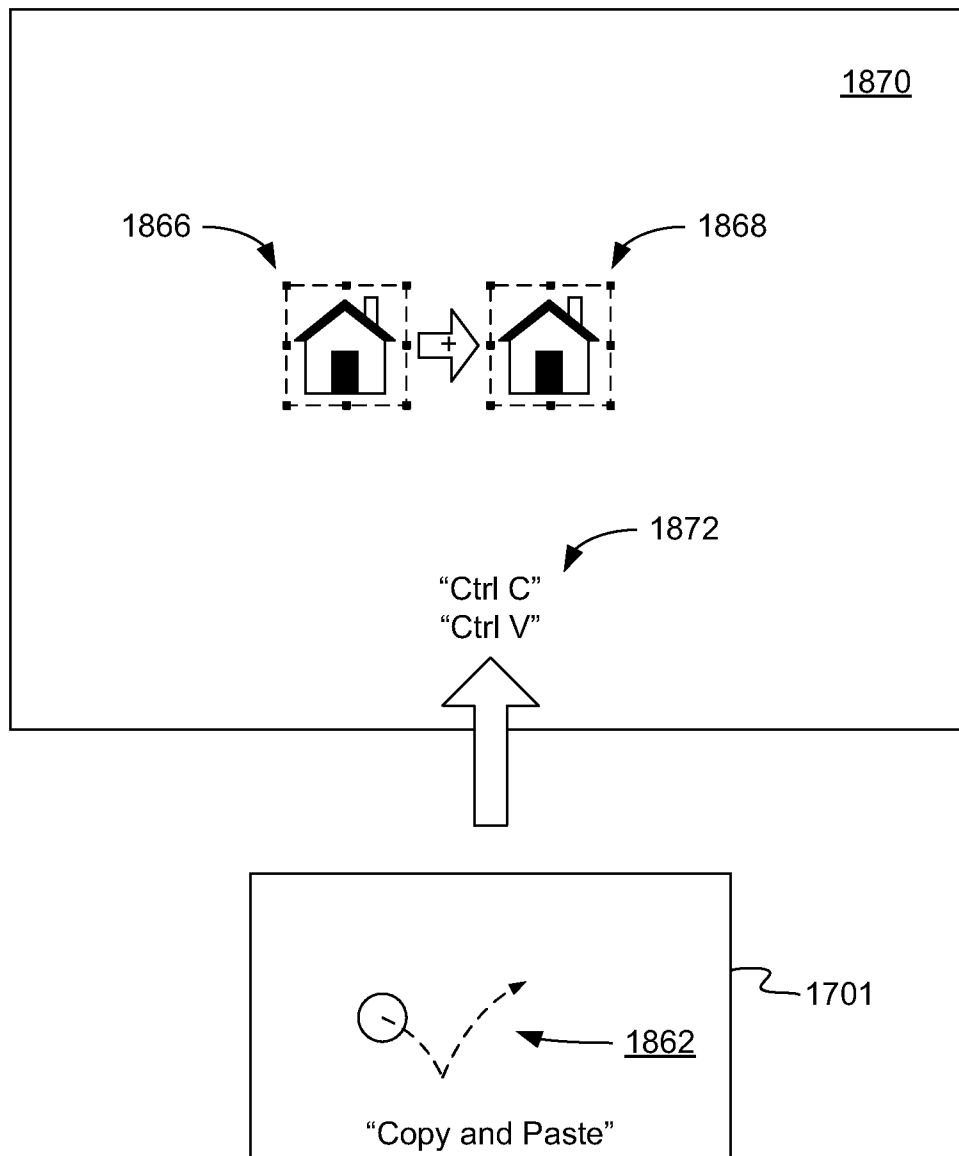


Figure 18B

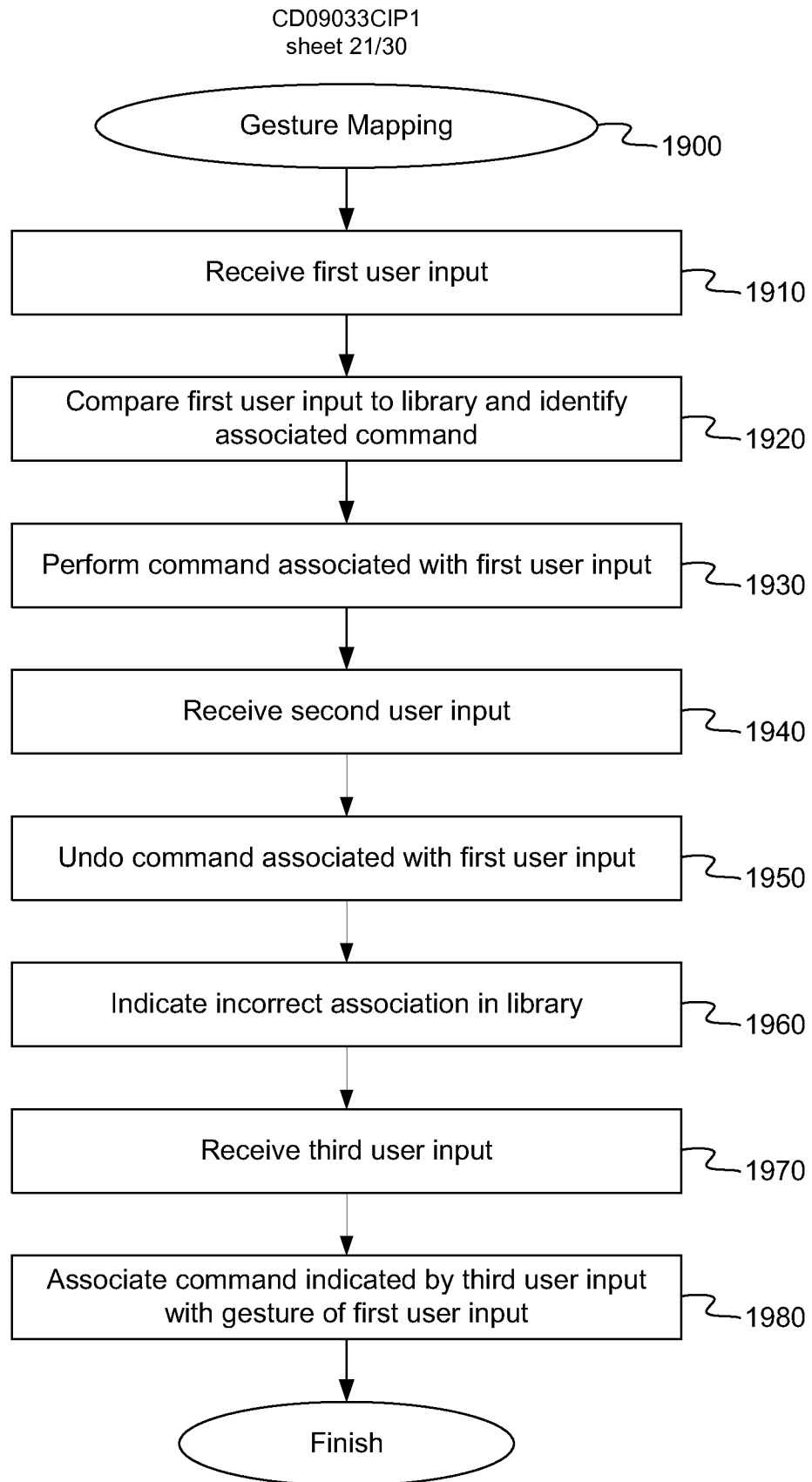


Figure 19A

CD09033CIP1
sheet 22/30

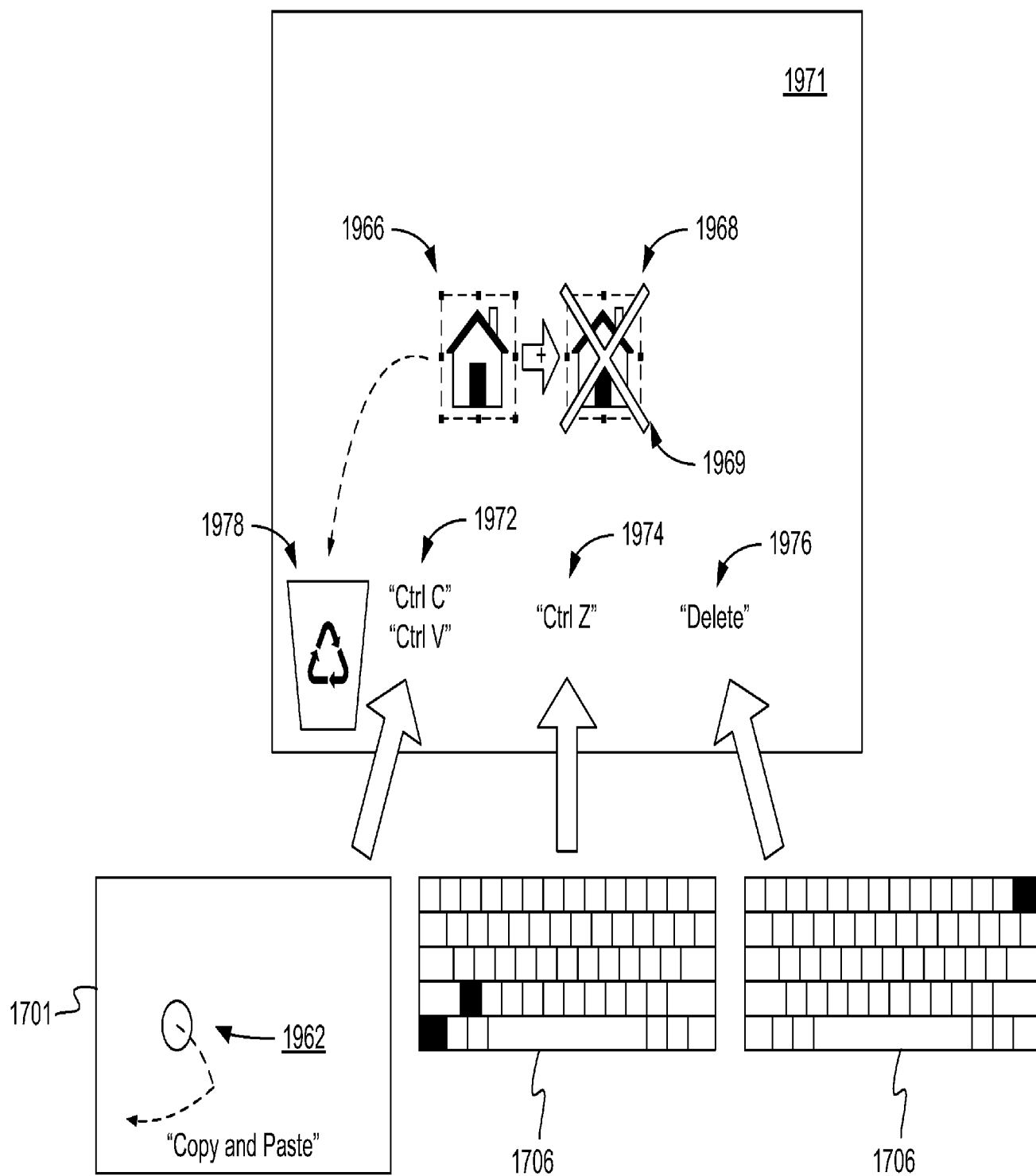


Figure 19B

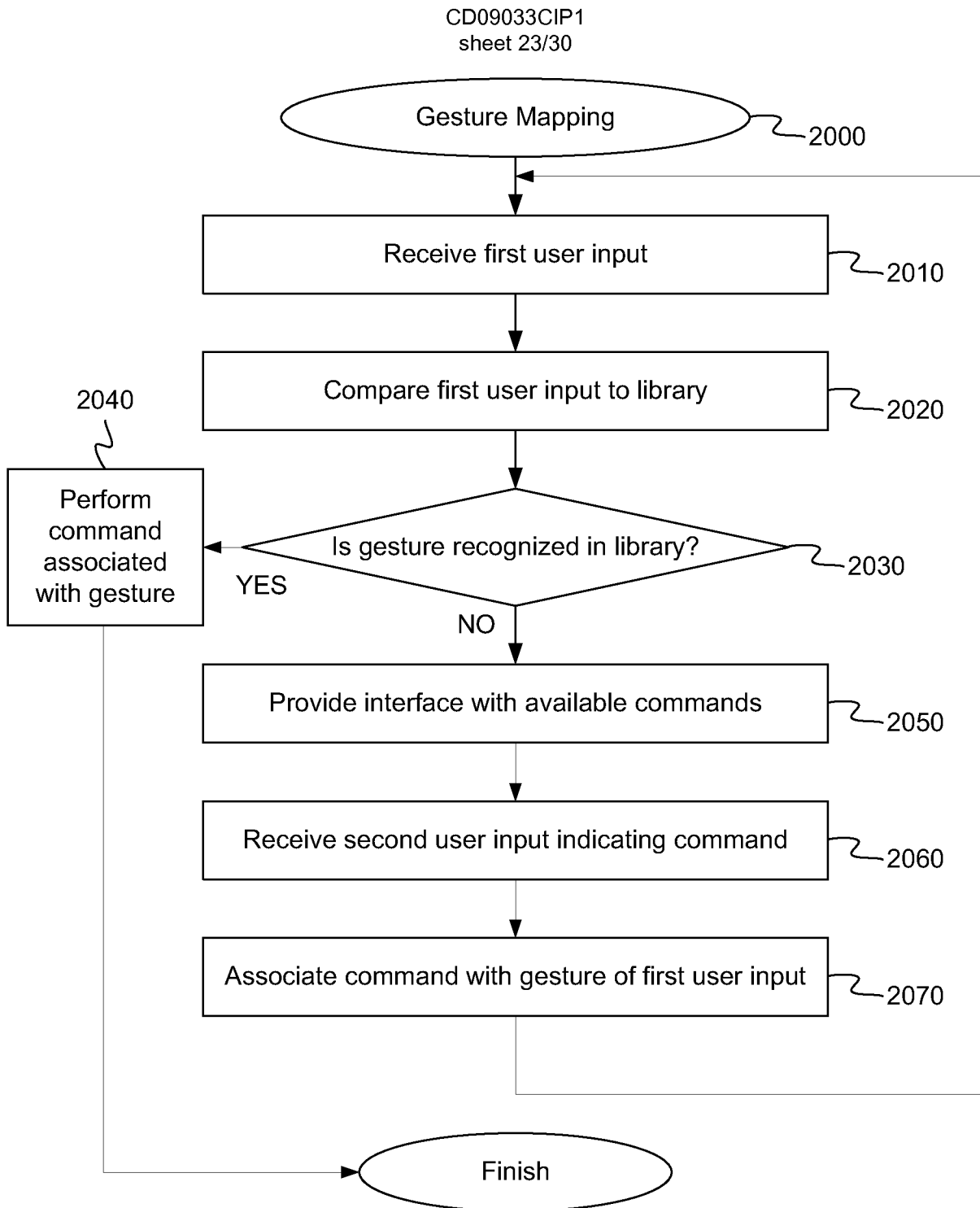


Figure 20A

CD09033CIP1
sheet 24/30

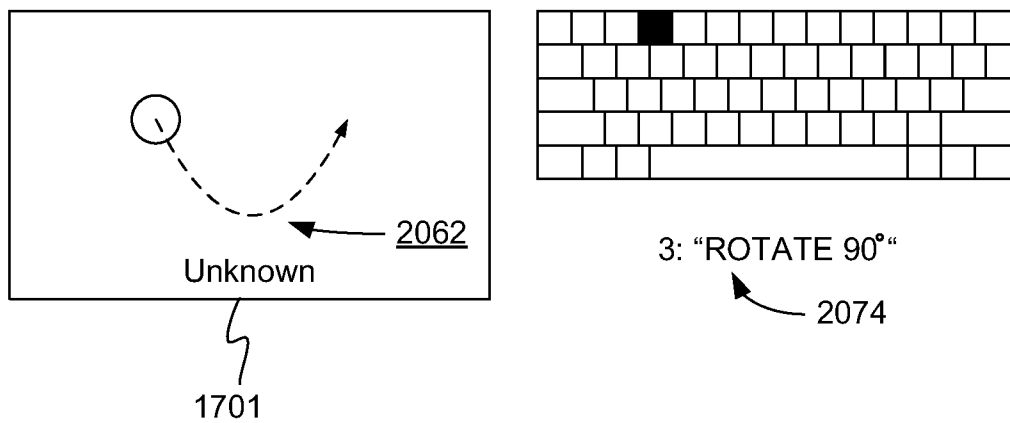
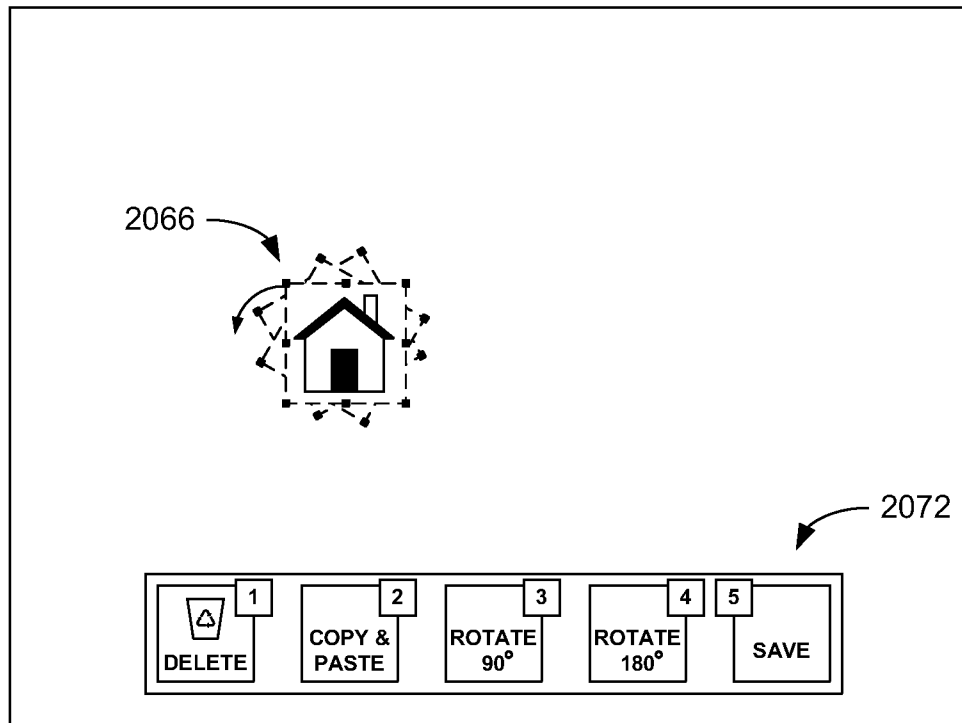


Figure 20B

CD09033CIP1
sheet 25/30

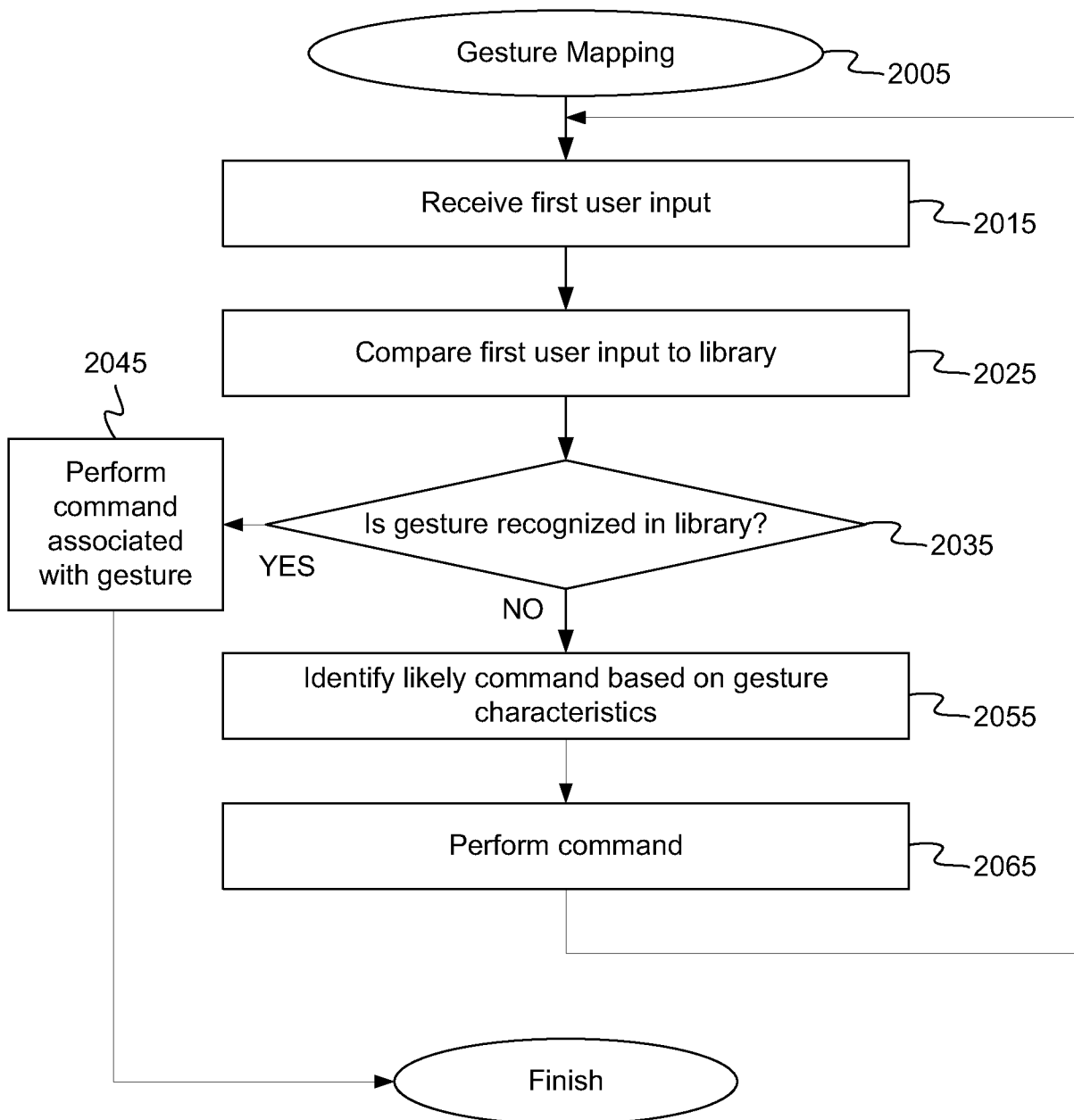


Figure 20C

CD09033CIP1
sheet 26/30

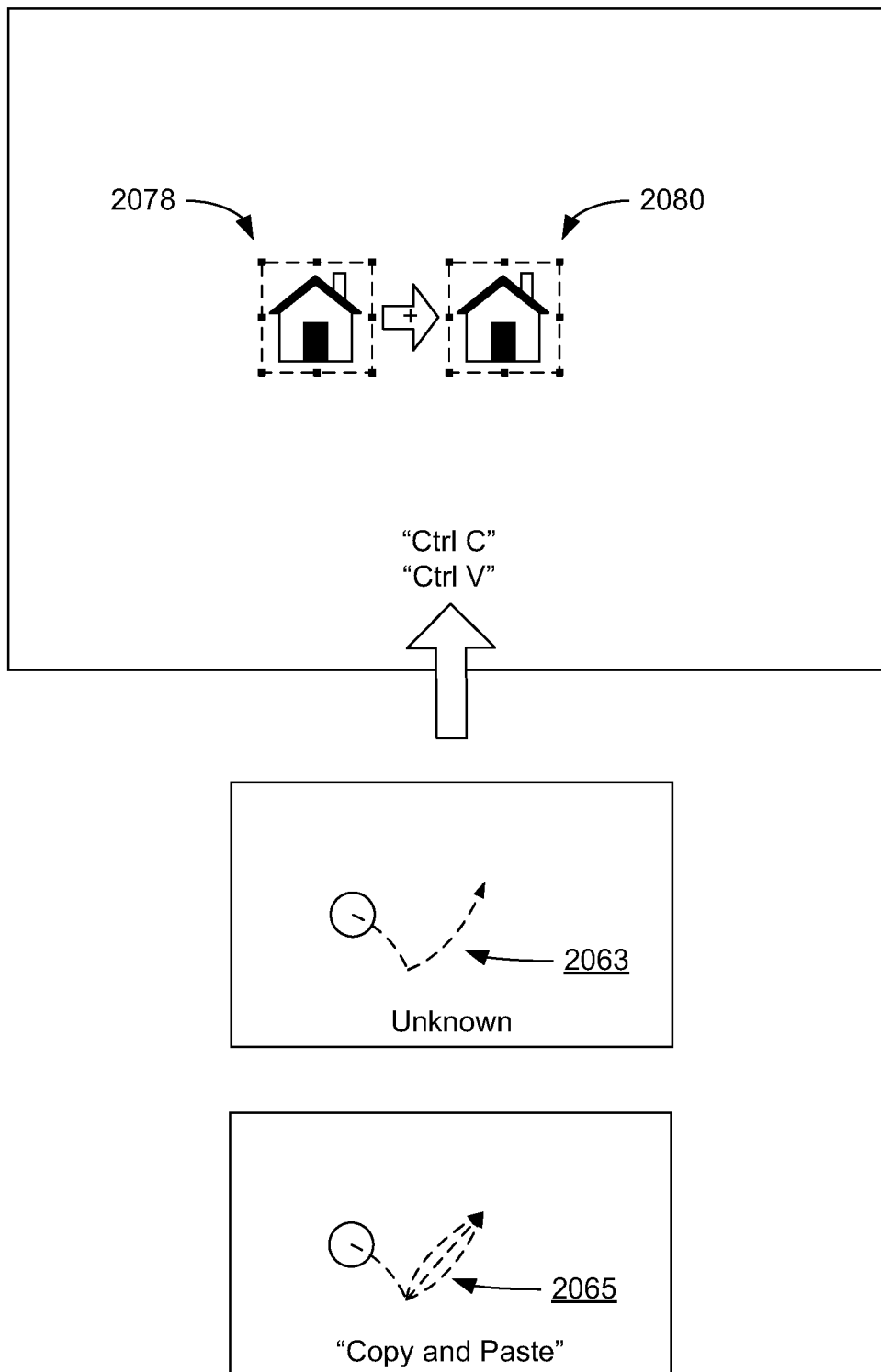


Figure 20D

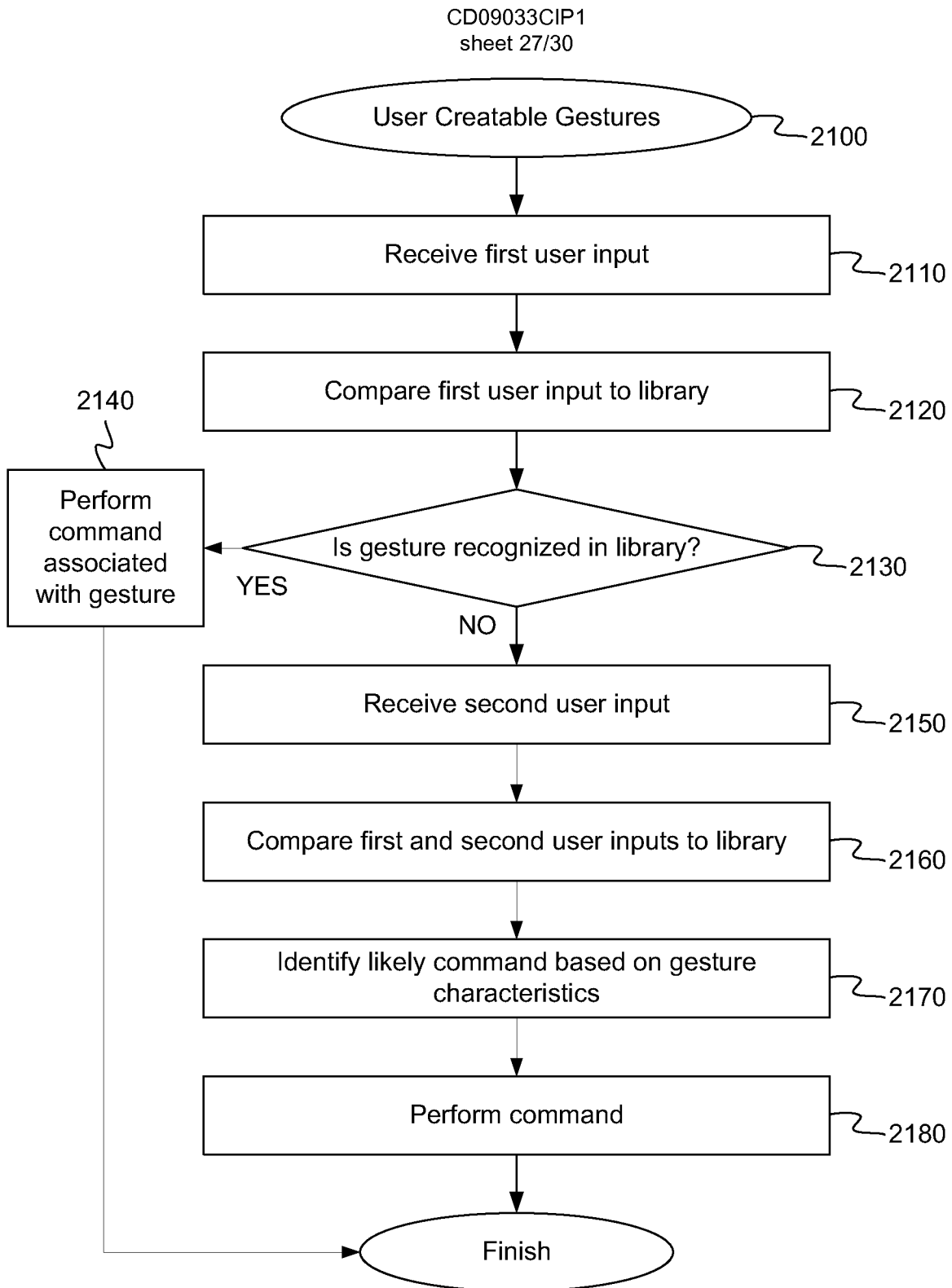


Figure 21A

CD09033CIP1
sheet 28/30

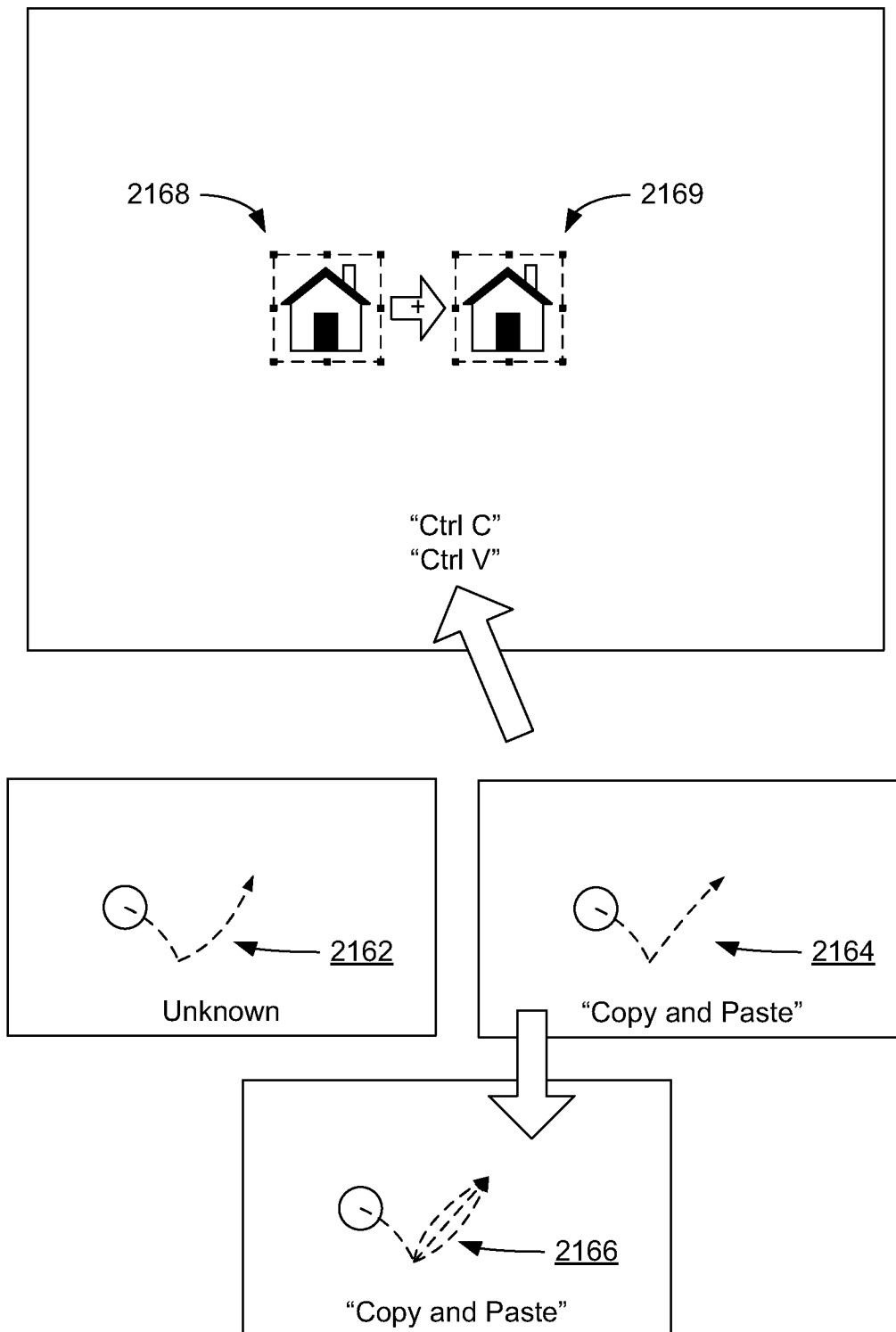


Figure 21B

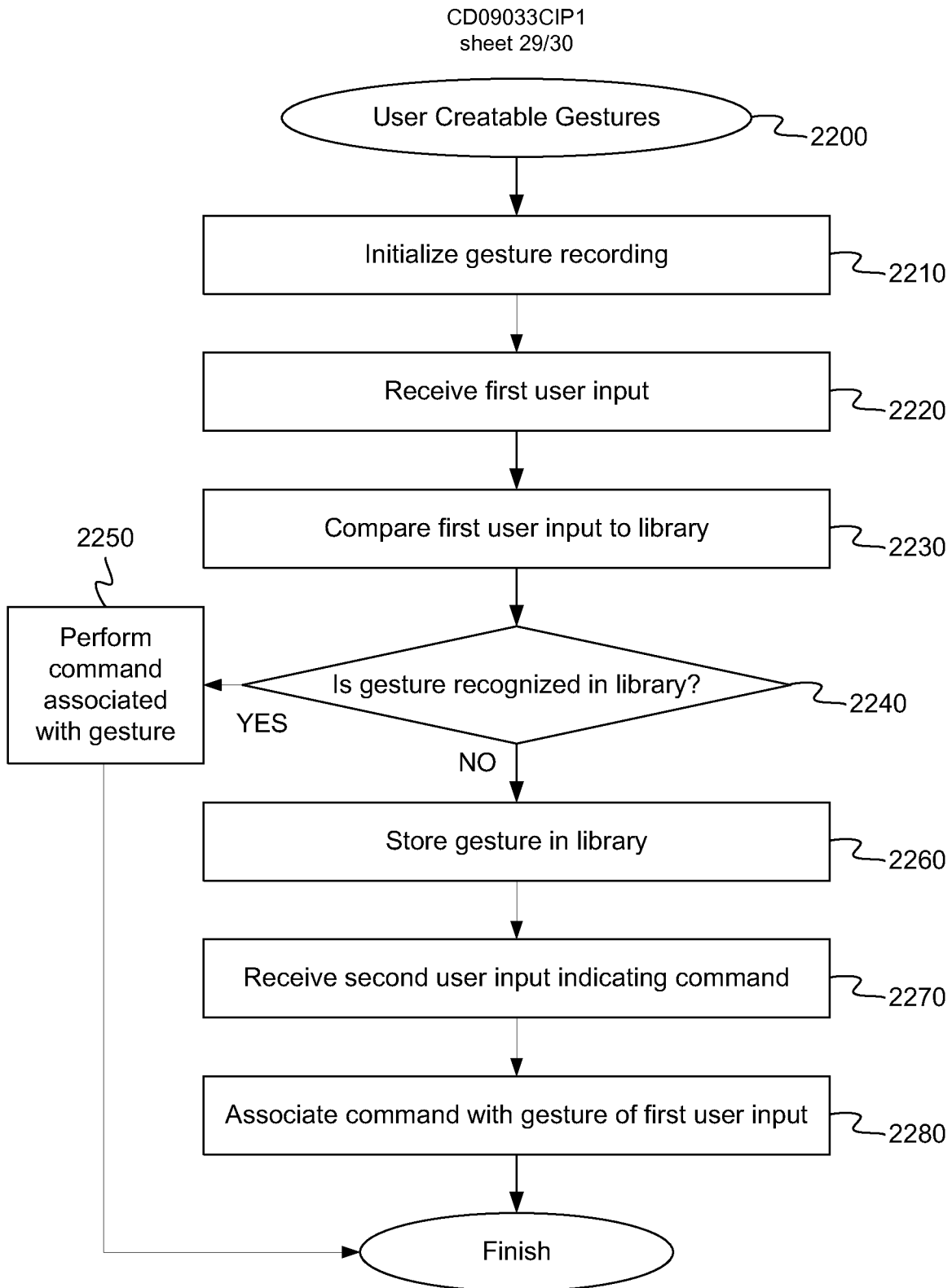


Figure 22A

CD09033CIP1
sheet 30/30

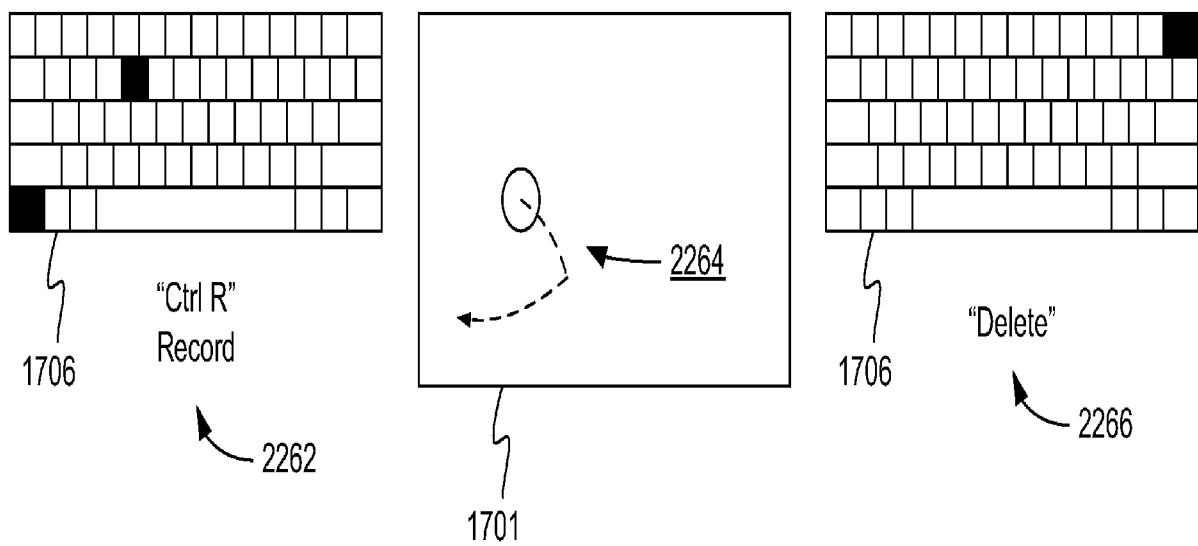
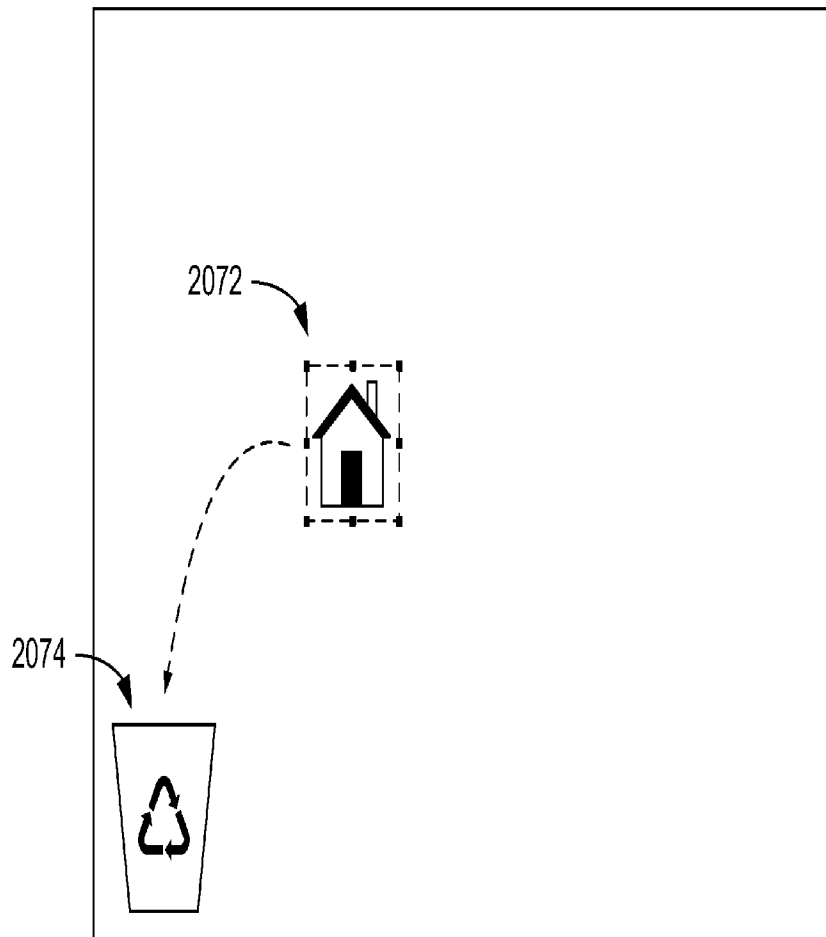


Figure 22B

INTERNATIONAL SEARCH REPORT		International application No. PCT/US13/53989		
A. CLASSIFICATION OF SUBJECT MATTER IPC: G09G 5/00(2006.01) USPC: 345/173 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) U.S. : 345/156, 173, 174 Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) EAST: same program and command, application specific with gesture				
C. DOCUMENTS CONSIDERED TO BE RELEVANT				
Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.		
X	US 2008/0036743 A1 (WESTERMAN et al) 14 February 2008(14.02.2008), see entire document.	1-20		
☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.				
<table style="width: 100%; border: none;"> <tr> <td style="width: 50%; vertical-align: top;"> * 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 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 </td> <td style="width: 50%; vertical-align: top;"> "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 </td> </tr> </table>			* 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 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
* 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 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 17 September 2013 (17.09.2013)		Date of mailing of the international search report 19 SEP 2013		
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 KEN WIEDER Telephone No. (571) 272.2691		

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US13/53989

Box No. I Nucleotide and/or amino acid sequence(s) (Continuation of item 1.c of the first sheet)

1. With regard to any nucleotide and/or amino acid sequence disclosed in the international application, the international search was carried out on the basis of:

a. type of material

☐

a sequence listing

☐

table(s) related to the sequence listing

b. format of material

☐

on paper

☐

in electronic form

c. time of filing/furnishing

☐

contained in the international application as filed

☐

filed together with the international application in electronic form

☐

furnished subsequently to this Authority for the purposes of search

2. ☐ In addition, in the case that more than one version or copy of a sequence listing and/or table relating thereto has been filed or furnished, the required statements that the information in the subsequent or additional copies is identical to that in the application as filed or does not go beyond the application as filed, as appropriate, were furnished.

3. Additional comments:

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US13/53989

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:
2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
 2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of any additional fees.
 3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
 4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:
- Remark on Protest**
- | | |
|--------------------------|---|
| ☐ | The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee. |
| ☐ | The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation. |
| ☐ | No protest accompanied the payment of additional search fees. |