



(51) International Patent Classification:
G06F 3/0488 (2013.01) *G06F 3/14* (2006.01)
G06F 3/0486 (2013.01)

(21) International Application Number:
PCT/US2017/015033

(22) International Filing Date:
26 January 2017 (26.01.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15/063,216 7 March 2016 (07.03.2016) US

(71) Applicant: INTEL CORPORATION [US/US]; 2200 Mission College Blvd., Santa Clara, California 95054 (US).

(72) Inventors: KUMAR, Arvind; 291 NW Pacific Grove Drive, Beaverton, Oregon 97006 (US). VALAVI, John J.; 7178 NE Rockridge Place, Hillsboro, Oregon 97124 (US). RAMADOSS, Murali; 1021 Kirby Court, Folsom, California 95630 (US).

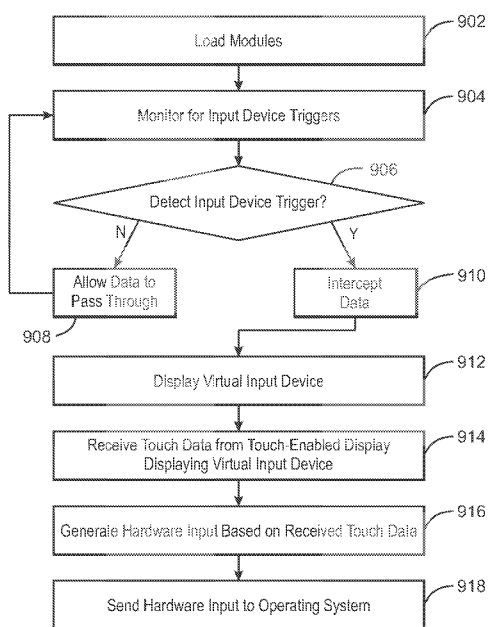
(74) Agents: PYTER, Wiktor J. et al.; International Ip Law Group, P.L.L.C., 13231 Champion Forest Dr., Suite 410, Houston, Texas 77069 (US).

(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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, 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, ST, 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,

[Continued on next page]

(54) Title: VIRTUAL INPUT DEVICE USING SECOND TOUCH-ENABLED DISPLAY



900
FIG. 9

(57) Abstract: An example system for generating hardware device input includes a gesture detector to detect an input device trigger from one of two coupled touch-enabled displays. The example system also further includes a redirector to intercept touch data from a triggered touch-enabled display. The example system further includes an emulator to generate hardware input data based on the intercepted touch data and send the hardware input data to an operating system. The example system also includes a user interface to display a virtual input device on the triggered touch-enabled display and receive touch data via the virtual input device.



SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ,
GW, KM, ML, MR, NE, SN, TD, TG).

— *as to applicant's entitlement to apply for and be granted
a patent (Rule 4.17(ii))*

Declarations under Rule 4.17:

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

Published:

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

VIRTUAL INPUT DEVICE USING SECOND TOUCH-ENABLED DISPLAY

Cross Reference to Related Application

[0001] The present application claims the benefit of the filing date of United States Patent Application Serial No. 15/063,216, by Kumar et al., entitled "Virtual Input Device using Second Touch-Enabled Display," filed March 7, 2016, and is incorporated herein by reference.

Background Art

[0002] Many mobile devices such as laptops and mobile phones include touch-enabled displays that enable a user to input data using a finger or stylus. In some examples, two touch-enabled displays can be combined to use one touch-enabled display as a virtual input device. For example, the virtual input device can be a virtual keyboard or a virtual touchpad, corresponding to hardware input devices such as keyboard and touchpad among other hardware input devices.

Brief Description of the Drawings

[0003] Fig. 1 is a diagram illustrating an example mobile computing device that can display a virtual input device at a second touch-enabled display and generate hardware input;

[0004] Fig. 2 is a schematic diagram illustrating a basic process for generating a virtual input device on a second touch-enabled display;

[0005] Fig. 3 is a schematic block diagram illustrating an example process for receiving input device data from a virtual input device on a second touch-enabled display;

[0006] Fig. 4 is a block diagram illustrating an example system for generating hardware input from a virtual input device;

[0007] Fig. 5 is a sequence diagram illustrating a process for generating a virtual input device on a second touch-enabled display;

[0008] Fig. 6 is a sequence diagram illustrating a process for disabling a virtual input device on a second touch-enabled display;

[0009] Fig. 7 is a block diagram illustrating an example computing device that can be used to generate hardware input from a virtual input device on a second touch-enabled display;

[0010] Fig. 8 is a flow chart illustrating a process for displaying a virtual input device on a second touch-enabled display;

[0011] Fig. 9 is a flow chart illustrating a process for generating hardware input from a virtual input device; and

[0012] Fig. 10 is a block diagram showing computer readable media that stores code for generating hardware input data.

[0013] The same numbers are used throughout the disclosure and the figures to reference like components and features. Numbers in the 100 series refer to features originally found in Fig. 1; numbers in the 200 series refer to features originally found in Fig. 2; and so on.

Description of the Aspects

[0014] The present disclosure relates generally to techniques for generating and receiving input from a virtual input device displayed on a display of one of two coupled touch-enabled displays. As the phrase is used herein, a virtual input device is a graphical user interface that mimics the appearance and functionality of a physical input device, such as a keyboard, touchpad or mouse. Specifically, the techniques described herein include a computer-readable medium, method and system for generating and receiving input from a virtual input device displayed on a second touch-enabled display. The techniques include disabling a display of the second touch-enabled display at an operating system (OS) during the display of the virtual input device. As discussed above, some laptops currently use two touch-enabled displays to form laptop designs. In some examples, one touch-enabled display can be used as a virtual keyboard. For example, OS soft keyboards can be used to provide a virtual keyboard interface for entering keyboard data. Likewise, OS-based touchpads can also be used.

[0015] However, OS-based soft keyboards and touchpads may not provide user-friendly experiences. For example, an operating system keyboard may appear on

the bottom portion of the screen where a touch happens. Thus, if a user interacts with content appearing on a top screen, the OS-based keyboard may appear on the top screen and cover almost half of the content on the screen, while the bottom screen is left unused. In some examples, one or more windows may be hidden behind the OS-based soft keyboard or applications may show up on the same screen as the OS-based soft keyboard, making the OS-based soft keyboard difficult to use. Moreover, using an OS-based soft keyboard may take away focus from applications if the OS-based soft keyboard is created as another application window. The techniques described herein enable a hardware input device experience on a second touch-enabled display. Thus, a user can use the second touch-enabled display in place of a hardware keyboard, touchpad, or other input device.

[0016] Referring now to Fig 1, illustrates an example mobile computing device that can display a virtual input device at a second touch-enabled display and generate hardware input. The example mobile computing device is generally referred to by the reference number 100.

[0017] The mobile computing device 100 shown in Fig. 1 includes a first touch-enabled display 102 coupled to a second touch-enabled display 104. The first touch-enabled display has a first display 106. For example, the first display 106 is shown displaying a web page with writing and pictures. The second touch-enabled display 104 includes a second display 108. The second display 108 includes a virtual keyboard 110. A pair of hands 112 are shown in contact with the surface of the second touch-enabled display 104.

[0018] In some examples, the mobile computing device 100 may have a number of modes for displaying content on the two touch-enabled displays 102, 104. For example, the mobile computing device 100 may change display configurations based on the orientation of the mobile computing device 100.

[0019] In Fig. 1, the mobile computing device 100 is shown in a keyboard orientation with two hands 112 contacting the second display 108 of the second touch-enabled display 104. In some examples, a virtual input device 110 can be displayed on the second touch-enabled display 104 in response to detecting a trigger at the second touch-enabled display 104. For example, the trigger can be a gesture

or the placement of one or more hands on the surface of the touch-enabled display 104. In some examples, the trigger can be a button press on the screen of a touch-enabled display or a physical button (not shown).

[0020] Fig. 2 is a diagram illustrating a basic process for generating a virtual input device on a second touch-enabled display. The example process is referred to generally by the reference number 200 and the example stages are referred to generally by the reference numbers 202A, 202B, 202C and can be implemented using the mobile computing device 100 of Fig. 1 above.

[0021] The first stage 202A shows a mobile computing device with a first touch-enabled display 102 and a second touch-enabled display 104. A pair of hands 112 are shown touching the touch-enabled display 104. In some examples, the touching of the touch-enabled display 104 can be detected and the display from the operating system can be disabled and an input device displayed as shown in the second stage 202B.

[0022] At the second stage 202B, the touch-enabled display 104 is shown displaying a virtual input device 110. In some examples, the virtual input device 110 can include a keyboard, a touchpad, or any other input device. For example, the touch-enabled display 104 may have had any display moved to the touch-enabled display 102 and had its display replaced with the virtual input device 110. In some examples, the touch-enabled display 104 can display the virtual input device 110 in response to the hands 112 being detected on the touch-enabled display 104. For example, the palms of the hands 112 may be detected at the touch-enabled display 104. In some examples, the detection of the hands 112 may be one of any number of possible triggers that can be used to enable the virtual input device 110. In some examples, an application can be used to manage the touch-enabled display 104 while it displays the virtual input device 110. As described in detail below, the virtual input device 110 can be a pseudo-device that emulates a hardware input device. Thus, the operating system of the mobile computing device may detect the loss of one display and an addition of a hardware input device. For example, the emulated hardware input device can be a keyboard with an integrated mouse and/or touchpad.

[0023] At the third stage, the touch-enabled display 104 no longer displays the

virtual input device 110. For example, the hands 112 may no longer be detected at the touch-enabled display 104 and the virtual input device 110 may have been disabled as described below with respect to at least Figs. 6 and 8. In some examples, the virtual input device 110 can be disabled upon detecting that the trigger is no longer present. In some examples, a second trigger can be used to disable the virtual input device 110. For example, a gesture or button on the touch-enabled display 104 can be used to disable the virtual input device 110. In some examples, the display can be restored on the touch-enabled display 104.

[0024] Thus, the touch-enabled display 104 can be temporarily removed from the operating system as a display and used instead as a virtual input device. For example, the operating system may detect a connected hardware device corresponding to the virtual input device. When the touch-enabled display 104 returns to a normal state as shown in stage 202C, the touch-enabled display 104 appears to the operating system as a second touch enabled display and can be used by the operating system as normal. As the phrase is used herein, a touch enabled display is a graphical user interface that is used primarily to display content to a user, but which also can receive touch input from a user through graphical input devices such as buttons, links, and the like. Examples of touch enabled displays include Web pages that enable users to touch links to additional Web pages, video displays with various graphical buttons and slider mechanisms, document displays that enable a user to move the cursor to a selected location in the text by touching the location.

[0025] The diagram of Fig. 2 is not intended to indicate that the example process 200 is to include all of the states shown in Fig. 2. Rather, the example process 200 can be implemented using fewer or additional states not illustrated in Fig. 2.

[0026] Fig. 3 is a schematic block diagram illustrating an example system for receiving input device data from a virtual input device on a second touch-enabled display. The example system is referred to generally by the reference number 300 and can be implemented using the computing device 100 of Fig. 1 above.

[0027] The example system 300 includes touch sensors 302 coupled to a touch integrated circuit 304 as indicated by an arrow 306. The system includes a GPU 308

coupled to the touch IC 304 as indicated by an arrow 310. The system includes an operating system 312 communicatively coupled to the GPU 308 as indicated by arrows 314, 316. The GPU 308 includes a number of vendor kernels 318 and a number of post-processing kernels 320 communicatively coupled to the vendor kernels 318 as indicated by an arrow 322. The post-processing kernels 320 include a gesture detection module (GDM) 324 and a redirector and emulator module (REM) 326 communicatively coupled to the GDM 324 as indicated by an arrow 328. The operating system 312 further includes touch data 330 and input device data 332 received from the REM 326. The operating system 312 also includes a number of applications 334 shown receiving the touch data 330 and the input device data 332 as indicated by arrows 336 and 338, respectively.

[0028] As shown in Fig. 3, the touch IC 304 can collect raw touch sensor data from the touch sensors 302 and sent the raw touch sensor data to the GPU 308 via a dedicated hardware path as indicated by arrow 310. For example, the raw touch data can include touch and/or stylus sensor data. The raw touch sensor data can be received by the vendor kernels 318 of the GPU 308. The vendor kernels 318 can be touch/stylus algorithms from touch-enabled display vendors that can run on the GPU 308. For example, the vendor kernels 318 can convert the raw sensor data into screen co-ordinates depicting touch points. In addition to this, the system 300 allows for additional processing of the data at the post-processing kernels 320. The post-processing kernels 320 allow additional code to interpret multi-touch and stylus XY coordinates and associated stylus/pen data. For example, the raw data can be interpreted as gestures corresponding to pen-up, pen-down, hover, pressure, etc. The post-processing kernels 320 can then process the interpreted raw data and make additional decisions. The final processed data can then be send to the operating system 312 as indicated by arrows 314 and 316.

[0029] In some examples, the post-processing kernels 320 can include a virtual input device post-processing kernel 320. For example, the virtual input device post-processing kernel 320 can include a gesture detection module (GDM) 324 and a redirection and emulation module (REM) 326. As touch events are detected on a surface of a touch-enabled display via touch sensors 302, the touch IC 304 collects

the raw data from the touch sensors 302 and the touch vendor kernels 318 process the raw touch sensor data and produce touch point data, which includes the XY coordinates for all the touch points on the screen. In some examples, the processed data can be in a standard Human Interface Device (HID) format and the post-processing kernels 320 can easily monitor this as the data is sent to the OS 312.

[0030] In some examples, the GDM 324 can constantly monitor the touch point data. For example, the GDM 324 can run a gesture detection algorithm for turning on this monitoring capability. In some examples, a specific trigger to display the virtual input device on the touch-enabled display can be predefined. For example, a trigger can be a gesture. In some examples, the gesture can be a detected placement of both hands on the screen, with both palms resting at bottom part of the screen, and all 10 fingers touching the screen. Since this is a normal gesture for people to type on a physical keyboard, and is therefore an intuitive gesture to bring up a virtual keyboard on the touch-enabled display. Another example of a gesture can be the user explicitly touching a seam where two screens meet and dragging down a virtual input device from the 'seam' onto the lower touch-enabled display. In some examples, the virtual input device can be dragged from any other edge of the touch-enabled display. Yet another gesture could be the user drawing a figure that can be used as a trigger. For example, the user can draw a letter "K" to bring up a virtual keyboard or a letter "T" to bring up a virtual touchpad. In some examples, the triggers can be configurable. For example, once a gesture is decided, it can be programmed in the GDM 324. In some examples, more than one trigger can also be programmed. Once the trigger to start the virtual input device is detected, the GDM module 324 can set an Input Device Active flag. The Input Device Active flag can then be shared with the REM 326.

[0031] In some examples, the (REM) 326 can operate in two modes based on the Input Device Active flag. For example, if the Input Device Active flag is not set, the REM 326 can work in pass-through mode and does not modify any touch data. Touch data can be delivered to the OS as touch HID reports. In some examples, the REM 326 can be a zero impact pass-through module. If the Input Device Active flag is set, the REM 326 can intercept all touch data and convert it into keyboard key-

strokes or other input device data and send keyboard HID reports (or other HID reports specific to the emulated input device) to the OS 312 through a driver. For example, the driver can be the Intel Precise Touch Driver. This input device data can be delivered to the window in focus as determined by the OS 312. Thus, input device data can be delivered to the OS 312 using a similar mechanism to that used when hard keyboard data gets delivered to the OS 312.

[0032] In some examples, the REM 326 can also interact with a user level component (not shown) using a private HID interface. For example, a user interface module (UIM) can be an OS-based application 334 that can render a virtual input device layout user interface (UI) on a touch-enabled display. The private HID interface and messages are described in greater detail below.

[0033] The diagram of Fig. 3 is not intended to indicate that the example system 300 is to include all of the components shown in Fig. 3. Rather, the example system 300 can be implemented using fewer or additional components not illustrated in Fig. 3 (e.g., additional sensors, integrated circuits, kernels, GPUs, applications, etc.).

[0034] Fig. 4 is a block diagram illustrating an example system for generating hardware input from a virtual input device. The system is generally referred to by the reference number 400 and can be performed by the mobile computing device 100 of Fig. 1.

[0035] The system 400 includes software 402, a central processing unit (CPU) 404, memory 406, a Platform Controller Hub (PCH) 408, and two touch integrated circuits (ICs) 410, 412. The software 402 includes touch enabled applications 414, legacy applications 416, a Human Interface Device (HID) class driver 418, a user interface module 420, a touch driver 422, and a graphic driver 424. The CPU 404 includes a graphics processing unit (GPU) 428. The GPU 428 includes a redirector and emulator module (REM) 430, a gesture detector module (GDM) 432, and vendor touch algorithms 434. The REM 430 is communicatively coupled to the memory 406 as indicated by an arrow 436. The vendor touch algorithms 434 are also communicatively coupled to the memory 406 as indicated by an arrow 438. The memory 406 includes Human Interface Device (HID) data and Raw Data 442. The PCH 408 includes CSME hardware 444 that is communicatively coupled to the GPU

428 as indicated by an arrow 446. The CSME hardware 444 includes a Direct Memory Access (DMA) Handler 448, a DMA module 450, a hardware bus interface (HW Bus I/F) 452 coupled to the DMA 450 as indicated by an arrow 454. The DMA 450 is further coupled to the memory 406 as indicated by an arrow 456. The hardware bus interface 452 is coupled to the two touch ICs 410, 412 as indicated by arrow 458.

[0036] In some examples, the vendor touch algorithms 434, GDM 432 and REM 430 can be loaded into system memory by the touch driver 422 during a system startup phase. For example, the touch driver 422 can be the Intel Precise Touch driver. Thereafter, the vendor touch algorithms 434, GDM 432 and REM 430 can execute every time a touch data comes in from touch sensors through Touch ICs 410, 412 and the DMA 450. This OS-independent execution allows the vendor touch algorithms 434, GDM 432 and REM 430 to operate in such a way that the OS can detect a real input device such as a keyboard or touchpad.

[0037] In some examples, an additional feature can ensure that OS applications do not interfere with the virtual input device user interface that is displayed by the UIM 420. For example, no application windows may be allowed beneath a keyboard UI or above a keyboard UI. In some examples, when a lower touch-enabled display is being used as a keyboard, the UIM 420 can manage the contents of the touch-enabled display entirely. This allows the input device user interface to provide a clobber-free experience. As used herein, clobbering refers to an interruption in the operation of a virtual input device by applications or operating system and any associated visual distractions. For example, the operation of the virtual input device may continue, but the OS may display a notification window where a user's palms may be at the touch-enabled display.

[0038] In some examples, the graphics driver 424 as shown in the figure can be used to assist the implementation of the UIM. For example, when an event is sent to the UIM 420 to display the virtual keyboard user interface, a special interrupt is also sent to the graphics driver 424 by the REM 430. This special interrupt can instruct the graphics driver 424 to issue a Display Detach event to the OS. The Display Detach event can make the lower touch display (where the input device is to appear)

disappear from OS enumeration. Thus, upon receiving the Display Detach event, the OS may be left with one display, and the OS may immediately move all the windows from the display of the lower touch-enabled display to the only remaining display on the upper touch-enabled display.

[0039] In some examples, the graphics driver 424 can provide a private interface to the UIM 420 for the UIM 420 to display a specific bitmap on the lower display. The UIM 420 can use this private interface to display the contents of the virtual keyboard. For example, the UIM 420 can be executed upon powering on the device 400 and can run in background and listen for HID events from the REM 430 on the private HID interface. The UIM 420 can receive a notification from the REM 430 on when to start and stop the virtual input device UI display. The UIM 420 can also inform the REM 430 about map layout using an HID set feature interface. For example, the map can enable the REM 430 to run in an autonomous fashion and convert XY co-ordinates or handwritten gestures to a specific key code without interacting with the UIM 420 on every co-ordinate translation or handwritten stroke. For example, the REM 430 can include a character recognition module that automatically detects and converts touch data received at a writing window into keyboard stroke data. Thus, the UIM 420 can be said to fully own displaying everything on the display of the second (in this case, lower) touch-enabled display. The OS therefore has no way to clobber this display.

[0040] In some examples, when the redirection mode is ended, the graphics driver 424 can notify the OS of the second display, and the two touch-enabled displays can then be fully functional displays for the operating system. For example, in a redirection mode the display of the upper touch-enabled display and upper screen touch may be available to the OS, while the display of the lower touch-enabled display and the lower touch-enabled display functionalities may be hidden from the OS. Instead, an emulated input device can be detected by the OS as a normal hardware input device. In normal mode, the upper and lower displays, as well as the upper and lower touch screen functionalities, are available to the OS. The emulated input device is thus not available to the OS in normal mode. The emulated input device can include, for example, a keyboard, a touchpad, a mouse

key, and other special function keys in a similar fashion as they are available on modern hardware keyboards, in addition to a writing window. A writing window, as used herein, refers to an area in the virtual input device in which a user can write text with a stylus or finger.

[0041] Thus, from an architecture perspective, the methods described herein are performed seamlessly when integrated with existing methods. The only change when using different types of input devices comes from what the UIM user interface displays, and what specific HID messages are sent from the REM to the host OS. For example, the HID messages can differ based on keyboard, mouse, or touchpad messages.

[0042] The diagram of Fig. 4 is not intended to indicate that the example system 400 is to include all of the components shown in Fig. 4. Rather, the example system 400 can be implemented using fewer or additional components not illustrated in Fig. 4 (e.g., additional touch ICs, memory, GPUs, modules, applications, etc.).

[0043] Fig. 5 is a control flow diagram illustrating a process for generating a virtual input device on a second touch-enabled display. The example process is generally referred to by the reference number 500 and can be implemented using the computing device 100 of Fig. 1.

[0044] The control flow diagram of Fig. 5 includes a GDM 502, an REM 504, a UIM 506, a graphics driver 508, and an OS 510. At time 512, the GMD 502 detects a trigger such as a gesture and sends a notification of the detected gesture to the REM 504. At time 514, the REM sends the UIM 506 a start notification. The start notification informs the UIM 506 that it is time to display the virtual input device user interface and send a KeyMap in the case of a keyboard. For example, the KeyMap can be sent using a Set Feature request. The REM 504 sends the Start notification when the GDM 502 sets the Input Device Active flag based on detection of a trigger such as a gesture. For example, a specific gesture may have been predefined or configured to be used to enable the virtual input device.

[0045] At time 516, the UIM sends a StartingInputDevice notification to the graphics driver 508. In some examples, the StartingInputDevice notification can be a special interrupt that is sent to the graphics driver 508. At time 518, the graphics

driver 508 sends a MonitorDetached notification to the OS 510. As MonitorDetached notification is sent to the OS, the graphics driver 508 and SW reconfigures the display and associated memory to receive the display contents from the virtual input device.

[0046] At time 520 the UIM 506 sends a DisplayPrivateUI notification to the graphics driver 508. As indicated by arrow 522, the graphics driver 508 bypasses the OS 510 to directly display the private UI at the second touch-enabled display. In some examples, when the graphics driver 508 hides the display from the OS 510, the graphics driver 508 can allow the UIM 506 to display a bitmap on the hidden display. For example, the graphics driver 508 can provide a library application programming interface (API) for the UIM 506 to display the bitmap. This bitmap may include the input keyboard, touchpad, mouse or pen/stylus image variants. These variants can be loaded from the library and rendered using a graphics processing unit (GPU). In some examples, the UIM 506 can call the API any number of times to show different bitmaps. For example, the pressing of keys and other dynamic feedback may involve the display of different bitmaps.

[0047] At time 524, the UIM 506 sends a SetKeyMap notification to the REM 504. For example, the SetKeyMap notification can allow the UIM 506 to send a keymap to the REM 504 for an autonomous parsing thereafter. In some examples, a Key Map Table may contain the information that is provided by the UIM 506 to the REM 504 describing the keyboard key data. Every entry in the table contains a keycode and the corresponding key position. For example, language and locales can be included. In some examples, the co-ordinate system can assume a top-left corner of the screen as (0, 0). Position and size of the key can be defined by a rectangle specified by the top-left corner and bottom-right corner. This flexibility allows for keys of different sizes. For example a SHIFT key can be specified to be bigger than the character keys, and a SPACE key can be very large. The UIM 506 can define the coordinates.

[0048] In some examples, the Key Map Table can be a static map that can then can be provided to the REM 504. Any touches in the area defined by the Key Map Table can be reported to the OS as keyboard HID reports. The keyboard HID

reports can also be reported to the UIM 506 through private Data HID reports as described below.

[0049] At time 526, the UIM 506 sends a SetTouchpad notification to the REM 504. For example, the SetTouchpad notification can allow the UIM 506 to send information about the position and size of the virtual touchpad, which may accompany a virtual keyboard. In some examples, a Touchpad Map Table similar to the KeyMap Table can be used. For example, the Touchpad Map Table can contain the co-ordinates and size of the virtual touchpad rectangle, as well as the sizes and positions of the left, middle and right touchpad buttons. Any touches in the area defined by entries in the Touchpad Map Table can be reported to the OS as the Touchpad HID packets. The Touchpad HID packets can also reported to the UIM 506 through private Data HID reports.

[0050] At time 528, the UIM 506 sends a SetPrivateKeyMap notification to the REM 504. For example, the SetPrivateKeyMap notification can allow the UIM 506 to send a keymap to the REM 504 for translating touches in some special regions to be privately sent to the UIM 506. In some examples, a Private Key Map table similar to the Key Map Table above can be used. The Private Key Map table can contain the co-ordinates and size of the keys that are considered private to the UIM 506. For example, any touches in the area defined by entries in this table are not reported to OS 510, but are instead reported to the UIM 506 through private Data HID reports. This allows the UIM 506 to reserve some space for special keys that can alter the behavior of the UI itself. For example, a special EXIT key can make the virtual keyboard disappear when pressed.

[0051] At time 530, the UIM 506 sends a StartRedirection notification 530 to the REM 504. The StartRedirection notification can activate the REM 504 to start suppressing touches and start sending keystrokes or touchpad events to the OS 510.

[0052] At time 532 a loop begins. The loop 532 includes the receiving of Touch Data at the REM 504 at time 534. In some examples, the Touch Data may have been processed by one or more vendor kernels as discussed above or by the GDM. The loop 532 further includes the sending of Device HID information from the REM

504 to the OS 510 at time 536. The loop 532 also includes the sending a PrivateDeviceData information from the REM 504 to the UIM 506 at time 538. For example, the REM 504 can send private data that shows what key(s) were pressed. Although the keystroke data also directly goes to the OS as indicated in time 536, this sideband notification allows the UIM 506 to further reflect something in the user interface. For example, when a user presses a soft keyboard button, the user interface may briefly change the color of the specific key(s) or activate a haptic feedback to provide a feeling of key press to the user. The PrivateDeviceData notification can enable such feedback.

[0053] In some examples, additional dynamic user interface adjustments can be performed. For example, a virtual touchpad can be dynamically placed keeping in mind the placement of a user's palms. The virtual touchpad can be placed in between two palms, but if user moves the palms, the virtual touchpad can be moved around. The UIM 506 can thus send SetTouchPadMap set feature requests anytime to change the behavior. In some examples, a user-draggable control can be used that allows the virtual touchpad to be manually moved by the user to a desired location. For example, private keys can be used to allow such a level of control.

[0054] This control flow diagram is not intended to indicate that the events of the example process 500 are to be executed in any particular order, or that all of the events are to be included in every case. Further, any number of additional events not shown may be included within the example process 500, depending on the details of the specific implementation. For example, although the way to enter the virtual input device redirection mode is illustrated via a detected gesture, any form of trigger can be used alternatively, such as some specific hardware buttons, or the placement of the device in certain specific way. In addition, the GDM 502 can continue to detect gestures while in the redirection mode after time 530. Although this continued detection is not shown in the flows, such continued detection can provide additional functionality of recognizing gestures. For example, a gesture including dragging the palm and fingers at a fast speed from left to right can be used make the keyboard disappear and the system return to a normal mode. Furthermore, a notification to the graphics driver 508 can also be sent directly from

the REM 504. For example, StartingInputDevice 516 message can be sent directly by REM 504 to Graphics Driver 508. This may allow the graphics driver 508 to send the MonitorDetached 518 message to the OS 510 in advance and be ready to receive the DisplayPrivateUI 520 message. Sending the StartingInputDevice 516 message directly from the REM 504 to the graphics driver 508 can make the transition to the redirection mode faster.

[0055] Fig. 6 is a control flow diagram illustrating a process for disabling a virtual input device on a second touch-enabled display. The example process is generally referred to by the reference number 600 and can be implemented using the mobile computing device 100 of Fig. 1.

[0056] The control flow diagram of Fig. 6 also includes a GDM 502, an REM 504, a UIM 506, a graphics driver 508, and an OS 510. The control flow diagram further includes the loop 532 discussed in detail with respect to Fig. 5 above. However, at time 602, the loop 532 is broken by the detection of an exit key at the UIM 506. For example, a Stop notification can allow the REM 504 to scan for specific key sequence, or detect a stop gesture in collaboration with the GDM 502 and send a Stop notification to the UIM 506 to kill the input device UI. At time 604, an ExitInputDevice notification is sent to the graphics driver 508 from the UIM 506. At time 606, a StopRedirection notification is sent from the UIM 506 to the REM 504. For example, the StopRedirection notification can allow the UIM 506 to instruct the REM 504 to return to a normal pass-through mode. Although this can be done by REM 504 itself, the StopRedirection notification allows further flexibility for an implementation to decide and allow multiple ways to return to normal mode. At the same time 608, the graphics driver 508 sends a MonitorAttached notification to the OS 510. At time 610, the REM 504 receives Touch Data. At time 612, the REM 504 sends Touch HID data to the OS 510.

[0057] This control flow diagram is not intended to indicate that the events of the example process 600 are to be executed in any particular order, or that all of the events are to be included in every case. Further, any number of additional events not shown may be included within the example process 600, depending on the details of the specific implementation. For example, the way to exit the keyboard

mode can be via another touch gesture, or via some specific hardware button, or a specific assigned key on the virtual keyboard.

[0058] Fig. 7 is a block diagram shown illustrating an example computing device that can be used to generate hardware input from a virtual input device on a second touch-enabled display. The computing device 700 may be, for example, a laptop computer, desktop computer, tablet computer, mobile device, or server, among others. The computing device 700 may include a central processing unit (CPU) 702 that is configured to execute stored instructions, as well as a memory device 704 that stores instructions that are executable by the CPU 702. The CPU 702 may be coupled to the memory device 704 by a bus 706. Additionally, the CPU 702 can be a single core processor, a multi-core processor, a computing cluster, or any number of other configurations. Furthermore, the computing device 700 may include more than one CPU 702. The memory device 704 can include random access memory (RAM), read only memory (ROM), flash memory, or any other suitable memory systems. For example, the memory device 704 may include dynamic random access memory (DRAM).

[0059] The computing device 700 may also include a graphics processing unit (GPU) 708. As shown, the CPU 702 may be coupled through the bus 706 to the GPU 708. The GPU 708 may be configured to perform any number of graphics operations within the computing device 700. For example, the GPU 708 may be configured to render or manipulate graphics images, graphics frames, videos, or the like, to be displayed to a user of the computing device 700.

[0060] The memory device 704 can include random access memory (RAM), read only memory (ROM), flash memory, or any other suitable memory systems. For example, the memory device 704 may include dynamic random access memory (DRAM). The memory device 704 may include device drivers 710 that are configured to execute the instructions for generating virtual input devices. The device drivers 710 may be software, an application program, application code, or the like.

[0061] The CPU 702 may also be connected through the bus 706 to an input/output (I/O) device interface 712 configured to connect the computing device

700 to one or more I/O devices 714. The I/O devices 714 may include, for example, a keyboard and a pointing device, wherein the pointing device may include a touchpad or a touchscreen, among others. The I/O devices 714 may be built-in components of the computing device 700, or may be devices that are externally connected to the computing device 700. In some examples, the memory 704 may be communicatively coupled to I/O devices 714 through direct memory access (DMA).

[0062] The CPU 702 may also be linked through the bus 706 to a display interface 716 configured to connect the computing device 700 to a display device 718. The display device 718 may include a display screen that is a built-in component of the computing device 700. The display device 718 may also include a computer monitor, television, or projector, among others, that is internal to or externally connected to the computing device 700.

[0063] The computing device 700 may also include a touch-enabled display interface 720 configured to connect the computing device 700 to touch-enabled displays 722. For example, the computing device 700 can receive touch or stylus input via the touch-enabled display interface.

[0064] The computing device also includes a storage device 724. The storage device 724 is a physical memory such as a hard drive, an optical drive, a thumbdrive, an array of drives, or any combinations thereof. The storage device 724 may also include remote storage drives. The storage device 724 includes a gesture detector 726, a redirector 728, an emulator 730, and a user interface 732. The gesture detector 726 can detect an input device trigger from one of two coupled touch-enabled displays. The redirector 728 can intercept touch data from the triggered touch-enabled display. The emulator 730 can generate hardware input data based on the intercepted touch data and send the hardware input data to an operating system. For example, the hardware input data can include Human Interface Device (HID) packets corresponding to keyboard strokes. The user interface 732 can display a virtual input device on the triggered touch-enabled display and receive touch data via the virtual input device. For example, the virtual input device can include an emulated keyboard, touchpad, and a writing window,

among other types of input. In some examples, the user interface 732 can receive a start request from the emulator and send a start input device request to a graphics driver. For example, the graphics driver can detach a display of the triggered touch-enabled display from the operating system response to receiving the start input device request. In some examples, the user interface 732 can receive a start request from the emulator and send a request to a graphics driver to display the virtual input device. In some examples, the emulator 730 can further generate private device data based on the intercepted touch data and send the private device data to a user interface to display feedback. For example, the feedback can be a replaced bitmap of a key in response to receiving touch data corresponding to a touch of the key. In some examples, the emulator 730 can recognize characters in the intercepted touch data and generate hardware input data such as keyboard strokes corresponding to the recognized characters. For example, the intercepted touch data can include one or more handwritten strokes received inside a writing window displayed by the user interface. Since the OS and applications receive hardware keyboard data rather than touch data, the OS and applications can use the keyboard data without any character recognition at or changes to the OS or applications.

[0065] The computing device 700 may also include a network interface controller (NIC) 734. The NIC 734 may be configured to connect the computing device 700 through the bus 706 to a network 736. The network 736 may be a wide area network (WAN), local area network (LAN), or the Internet, among others. In some examples, the device may communicate with other devices through a wireless technology. For example, the device may communicate with other devices via a wireless local area network connection. In some examples, the device may connect and communicate with other devices via Bluetooth® or similar technology.

[0066] The block diagram of Fig. 7 is not intended to indicate that the computing device 700 is to include all of the components shown in Fig. 7. Rather, the computing system 700 can include fewer or additional components not illustrated in Fig. 7, such as additional engines, additional network interfaces, and the like. The computing device 700 may include any number of additional components not shown

in Fig. 7, depending on the details of the specific implementation. Furthermore, any of the functionalities of the CPU 702 may be partially, or entirely, implemented in hardware and/or in a processor. For example, the functionality of the gesture detector 726, the redirector 728, the emulator 730, and the user interface 732 may be implemented with an application specific integrated circuit, in logic implemented in a processor, in logic implemented in a specialized graphics processing unit, or in any other device.

[0067] Fig. 8 is a flow chart illustrating a process for displaying a virtual input device on a second touch-enabled display. The example process is generally referred to by the reference number 800 and can be implemented using the processor 702 of the computing device 700 of Fig. 7 above. For example, the process can be performed by the gesture detector 726, the redirector 728, emulator 730, and user interface 732 of Fig. 7.

[0068] At block 802, the gesture detector monitors a touch-enabled display for gestures. In some examples, the gesture can monitor two or more touch-enabled displays for gestures.

[0069] At block 804, the gesture detector detects an input device trigger from a touch-enabled display. For example, the input device trigger can be a gesture, a hardware button, or a placement of the touch-enabled display.

[0070] At block 806, the user interface disables the touch-enabled display at an operating system and displays virtual input device on the disabled touch-enabled display. For example, the touch-enabled display may display a virtual keyboard or a virtual touchpad. In some examples, the virtual input device can include a writing window to receive one or more handwritten strokes to be converted to keyboard strokes. In some examples, disabling the touch-enabled display can include sending a message to the operating system that the touch-enabled display has been detached.

[0071] At block 808, the user interface detects an adjustment gesture. For example, the adjustment gesture can be a movement between two detected palms. In some examples, the display may include a user-draggable control on the disabled touch-enabled display.

[0072] At block 810, the user interface adjusts a virtual input device. For example, the virtual input device can be made smaller or larger based on the detected adjustment gesture. In some examples, the virtual input device can be relocated on the touch-enabled display based on the detected adjustment gesture. In some examples, and the user interface can move the virtual input device to a new location based on touch data received at the user-draggable control.

[0073] At block 812, the gesture detector detects an absence of any input device gesture. For example, a user may have removed both hands from the touch-enabled display.

[0074] At block 814, the user interface disables the virtual input device. For example, the virtual input device may disappear from the touch-enabled display and the touch-enabled display may return to a default mode of operation. In some examples, the user interface can disable the virtual input device by detecting an exit trigger and disabling the virtual input device in response to detecting the exit trigger. In some examples, the user interface can disable the virtual input device in response to detecting a predefined placement of the touch-enabled display. For example, the touch-enabled display may be placed in a reading or book mode and the virtual input device may disappear.

[0075] This process flow diagram is not intended to indicate that the blocks of the example process 800 are to be executed in any particular order, or that all of the blocks are to be included in every case. Further, any number of additional blocks not shown may be included within the example process 800, depending on the details of the specific implementation. For example, the gesture detector can continuously monitor the touch-enabled display for additional gestures and the user interface can then disable the virtual input device in response to detecting an exit gesture.

[0076] Fig. 9 is a flow chart illustrating a process for generating hardware input from a virtual input device. The example process is generally referred to by the reference number 900 and can be implemented using the processor 702 of the computing device 700 of Fig. 7 above. For example, the process can be performed by the gesture detector 726, the redirector 728, the emulator 730, and the user interface 732 of Fig. 7.

[0077] At block 902, the processor loads modules. For example, the gesture detector, redirector, emulator and user interface may be loaded by the processor upon bootup of the computing device.

[0078] At block 904, the gesture detector monitors a touch-enabled display for input device triggers. For example, an input device trigger can be a gesture, a button press, or a placement of the touch-enabled display.

[0079] At block 906, the gesture detector determines whether an input device trigger has been detected at the touch-enabled display. If an input device trigger is detected, then the method may proceed to block 910. If an input device trigger is not detected, then the method may proceed at block 908.

[0080] At block 908, the redirector allows data from the touch-enabled display to pass through to the operating system in response to not detecting an input device trigger.

[0081] At block 910, the redirector intercepts data from the touch-enabled display in response to detecting the input device trigger. For example, raw touch data from the touch-enabled display can be captured and redirected from the operating system. In some examples, a monitor detachment notification can be sent to the operating system. For example, the operating system can move any windows from the disabled touch-enabled display to another touch-enabled display.

[0082] At block 912, the emulator displays a virtual input device on the touch-enabled display. For example, the virtual input device may be a virtual keyboard, a virtual touchpad, among other possible virtual input devices, such as a writing window. For example, a writing window can be an input box to receiving handwritten characters such as letters, digits, or other characters.

[0083] At block 914, the emulator receives touch data from the touch-enabled display displaying the virtual input device. For example, the touch data can be raw touch data including coordinates of detected touches among other information.

[0084] At block 916, the emulator generates hardware input based on the received touch data. For example, the hardware input can include Human Interface Device (HID) data corresponding to hardware such as a keyboard and touchpad. In some examples, the emulator can generate private device data based on the

received touch data. In some examples, the emulator can perform character recognition on the received touch data and convert detected characters into hardware input.

[0085] At block 918, the emulator sends the hardware input to the operating system. For example, the operating system may receive the hardware input as if the emulator was a hardware keyboard or a hardware touchpad. In some examples, the emulator can also send the private device data to a virtual input device to provide feedback.

[0086] This process flow diagram is not intended to indicate that the blocks of the example process 900 are to be executed in any particular order, or that all of the blocks are to be included in every case. Further, any number of additional blocks not shown may be included within the example process 900, depending on the details of the specific implementation.

[0087] Fig. 10 is a block diagram showing computer readable media 1000 that store code for generating hardware input data. The computer readable media 1000 may be accessed by a processor 1002 over a computer bus 1004. Furthermore, the computer readable medium 1000 may include code configured to direct the processor 1002 to perform the methods described herein. In some embodiments, the computer readable media 1000 may be non-transitory computer readable media. In some examples, the computer readable media 1000 may be storage media.

[0088] The various software components discussed herein may be stored on one or more computer readable media 1000, as indicated in Fig. 10. For example, a gesture detector module 1006 may be configured to detect an input device trigger from one of two coupled touch-enabled displays. A redirector and emulator module 1008 may be configured to intercept touch data from the triggered touch-enabled display. A user interface module 1010 may be configured to display of a virtual input device on the triggered touch-enabled display. The redirector and emulator module 1008 may be configured to generate hardware input data based on the intercepted touch data. The redirector and emulator module 1008 may be configured to send the hardware input data to an operating system. In some examples, the user interface module 1010 may be configured to receive a start request and send a start

input device request to a graphics driver. For example, the graphics driver can detach a display of the triggered touch-enabled display from the operating system response to receiving the start input device request. In some examples, the user interface module 1010 may be configured to receive a start request from the emulator and send a request to a graphics driver to display the virtual input device. In some examples, the redirector and emulator module 1008 may be configured to generate private device data based on the intercepted touch data and send the private device data to a virtual input device to display feedback. For example, the feedback can be visual feedback in the form of an updated bitmap with a pressed button. In some examples, the user interface module 1010 may be configured to disable the virtual input device in response to detecting an exit trigger. For example, the user interface module 1010 may be configured to send an ExitInputDevice notification to a graphics driver in response to detecting an exit trigger.

[0089] The block diagram of Fig. 10 is not intended to indicate that the computer readable media 1000 is to include all of the components shown in Fig. 10. Further, the computer readable media 1000 may include any number of additional components not shown in Fig. 10, depending on the details of the specific implementation.

EXAMPLES

[0090] Example 1 is a system for generating hardware device input. The system includes a gesture detector to detect an input device trigger from one of two coupled touch-enabled displays. The system also includes a redirector to intercept touch data from a triggered touch-enabled display. The system further includes an emulator to generate hardware input data based on the intercepted touch data and send the hardware input data to an operating system. The system also further includes a user interface to display a virtual input device on the triggered touch-enabled display and receive touch data via the virtual input device.

[0091] Example 2 includes the system of example 1, including or excluding optional features. In this example, the user interface is to receive a start request from the emulator and send a start input device request to a graphics driver. The

graphics driver is to detach the triggered touch-enabled display from the operating system in response to receiving the start input device request.

[0092] Example 3 includes the system of any one of examples 1 to 2, including or excluding optional features. In this example, the user interface is to receive a start request from the emulator and send a request to a graphics driver to display the virtual input device.

[0093] Example 4 includes the system of any one of examples 1 to 3, including or excluding optional features. In this example, the emulator is to generate private device data based on the intercepted touch data and send the private device data to a user interface to display feedback.

[0094] Example 5 includes the system of any one of examples 1 to 4, including or excluding optional features. In this example, the system includes a graphics processing unit. The graphics processing unit includes the gesture detector, the redirector, and the emulator.

[0095] Example 6 includes the system of any one of examples 1 to 5, including or excluding optional features. In this example, the system includes a keymap to be used by the emulator module to generate the hardware input data based on the intercepted touch data.

[0096] Example 7 includes the system of any one of examples 1 to 6, including or excluding optional features. In this example, the user interface is to display private keys to be used to modify or relocate the virtual input device.

[0097] Example 8 includes the system of any one of examples 1 to 7, including or excluding optional features. In this example, the virtual input device includes a virtual keyboard, a virtual touchpad, a writing window, or any combination thereof.

[0098] Example 9 includes the system of any one of examples 1 to 8, including or excluding optional features. In this example, the virtual input device includes a virtual keyboard and the hardware input data includes Human Interface Device (HID) packets corresponding to keyboard strokes.

[0099] Example 10 includes the system of any one of examples 1 to 9, including or excluding optional features. In this example, the virtual input device includes a writing window and the emulator is to recognize characters in the intercepted touch

data and generate the hardware input data including keyboard strokes corresponding to the recognized characters.

[0100] Example 11 is a method for generating hardware device input. The method includes monitoring a touch-enabled display for an input device trigger. The method also includes detecting the input device trigger from the touch-enabled display. The method further includes disabling the touch-enabled display at an operating system in response to detecting the input device trigger. The method also further includes displaying a virtual input device on the disabled touch-enabled display in response to detecting the trigger.

[0101] Example 12 includes the method of example 11, including or excluding optional features. In this example, the method includes receiving touch data from the virtual input device. The method also includes generating hardware input based on the received touch data. The method further includes sending the hardware input to the operating system.

[0102] Example 13 includes the method of any one of examples 11 to 12, including or excluding optional features. In this example, the method includes detecting an adjustment gesture and adjusting the virtual input device based on the adjustment gesture.

[0103] Example 14 includes the method of any one of examples 11 to 13, including or excluding optional features. In this example, the method includes detecting an absence of an input device trigger and disabling the virtual input device in response to detecting the absence of the input device trigger.

[0104] Example 15 includes the method of any one of examples 11 to 14, including or excluding optional features. In this example, disabling the touch-enabled display includes sending a message to the operating system that the touch-enabled display has been detached.

[0105] Example 16 includes the method of any one of examples 11 to 15, including or excluding optional features. In this example, the method includes detecting an exit trigger and disabling the virtual input device in response to detecting the exit trigger.

[0106] Example 17 includes the method of any one of examples 11 to 16, including or excluding optional features. In this example, the method includes receiving touch data from the virtual input device. The method also includes generating private device data based on the received touch data. The method further includes sending the private device data to a user interface to provide feedback.

[0107] Example 18 includes the method of any one of examples 11 to 17, including or excluding optional features. In this example, the method includes displaying a user-draggable control on the disabled touch-enabled display and moving the virtual input device to a new location based on touch data received at the user-draggable control.

[0108] Example 19 includes the method of any one of examples 11 to 18, including or excluding optional features. In this example, the method includes monitoring the touch-enabled display for additional gestures and disabling the virtual input device in response to detecting an exit gesture.

[0109] Example 20 includes the method of any one of examples 11 to 19, including or excluding optional features. In this example, the method includes disabling the virtual input device in response to detecting a predefined placement of the touch-enabled display.

[0110] Example 21 is at least one computer readable medium for generating hardware input data having instructions stored therein that. The computer-readable medium includes instructions that direct the processor to detect an input device trigger from one of two coupled touch-enabled displays. The computer-readable medium includes instructions that direct the processor to intercept touch data from the triggered touch-enabled display. The computer-readable medium includes instructions that direct the processor to display a virtual input device on the triggered touch-enabled display. The computer-readable medium includes instructions that direct the processor to generate hardware input data based on the intercepted touch data; and send the hardware input data to an operating system.

[0111] Example 22 includes the computer-readable medium of example 21, including or excluding optional features. In this example, the computer-readable

medium includes instructions to receive a start request and send a start input device request to a graphics driver. The graphics driver is to detach the triggered touch-enabled display from the operating system response to receiving the start input device request.

[0112] Example 23 includes the computer-readable medium of any one of examples 21 to 22, including or excluding optional features. In this example, the computer-readable medium includes instructions to receive a start request from the emulator and send a request to a graphics driver to display the virtual input device.

[0113] Example 24 includes the computer-readable medium of any one of examples 21 to 23, including or excluding optional features. In this example, the computer-readable medium includes instructions to generate private device data based on the intercepted touch data and send the private device data to a user interface to display feedback.

[0114] Example 25 includes the computer-readable medium of any one of examples 21 to 24, including or excluding optional features. In this example, the computer-readable medium includes instructions to disable the virtual input device in response to detecting an exit trigger.

[0115] Example 26 includes the computer-readable medium of any one of examples 21 to 25, including or excluding optional features. In this example, the computer-readable medium includes instructions to receive touch data from the virtual input device. The computer-readable medium includes instructions that direct the processor to generate hardware input based on the received touch data. The computer-readable medium includes instructions that direct the processor to send the hardware input to the operating system

[0116] Example 27 includes the computer-readable medium of any one of examples 21 to 26, including or excluding optional features. In this example, the computer-readable medium includes instructions to monitor the touch-enabled display for additional gestures and disable the virtual input device in response to detecting an exit gesture:

[0117] Example 28 includes the computer-readable medium of any one of examples 21 to 27, including or excluding optional features. In this example, the

computer-readable medium includes instructions to detect an absence of an input device trigger and disable the virtual input device in response to detecting the absence of the input device trigger.

[0118] Example 29 includes the computer-readable medium of any one of examples 21 to 28, including or excluding optional features. In this example, the computer-readable medium includes instructions to detect an adjustment gesture and adjust the virtual input device based on the adjustment gesture.

[0119] Example 30 includes the computer-readable medium of any one of examples 21 to 29, including or excluding optional features. In this example, the computer-readable medium includes instructions to detect an exit trigger and disabling the virtual input device in response to detecting the exit trigger.

[0120] Example 31 is a system for generating hardware device input. The system includes means for detecting an input device trigger from one of two coupled touch-enabled displays. The system also includes means for intercepting touch data from the triggered touch-enabled display. The system further includes means for generating hardware input data based on the intercepted touch data. The system also further includes means for sending the hardware input data to an operating system. The system also includes means for displaying a virtual input device on the triggered touch-enabled display.

[0121] Example 32 includes the system of example 31, including or excluding optional features. In this example, the system includes means for receiving a start request from the emulator and send a start input device request to a graphics driver. The graphics driver is to detach the triggered touch-enabled display from the operating system response to receiving the start input device request.

[0122] Example 33 includes the system of any one of examples 31 to 32, including or excluding optional features. In this example, the system includes means for receiving a start request from the emulator and send a request to a graphics driver to display the virtual input device.

[0123] Example 34 includes the system of any one of examples 31 to 33, including or excluding optional features. In this example, the means for intercepting

touch data is to further generate private device data based on the intercepted touch data and send the private device data to a user interface to display feedback.

[0124] Example 35 includes the system of any one of examples 31 to 34, including or excluding optional features. In this example, the system includes means for processing graphics.

[0125] Example 36 includes the system of any one of examples 31 to 35, including or excluding optional features. In this example, the system includes means for generating the hardware input data based on the intercepted touch data.

[0126] Example 37 includes the system of any one of examples 31 to 36, including or excluding optional features. In this example, the system includes means for modifying or relocating the virtual input device.

[0127] Example 38 includes the system of any one of examples 31 to 37, including or excluding optional features. In this example, the virtual input device includes a virtual keyboard, a virtual touchpad, a writing window, or any combination thereof.

[0128] Example 39 includes the system of any one of examples 31 to 38, including or excluding optional features. In this example, the virtual input device includes a virtual keyboard and the hardware input data includes Human Interface Device (HID) packets corresponding to keyboard strokes.

[0129] Example 40 includes the system of any one of examples 31 to 39, including or excluding optional features. In this example, the virtual input device includes a writing window and the means for generating hardware input data is to recognize characters in the intercepted touch data and generate the hardware input data including keyboard strokes corresponding to the recognized characters.

[0130] Example 41 is an apparatus for generating hardware device input. The apparatus includes a gesture detector to detect an input device trigger from one of two coupled touch-enabled displays. The apparatus also includes a redirector to intercept touch data from the triggered touch-enabled display. The apparatus further includes an emulator to generate hardware input data based on the intercepted touch data and send the hardware input data to an operating system. The apparatus

also further includes a user interface to display a virtual input device on the triggered touch-enabled display and receive touch data via the virtual input device.

[0131] Example 42 includes the apparatus of example 41, including or excluding optional features. In this example, the apparatus includes a user interface to receive a start request from the emulator and send a start input device request to a graphics driver. The graphics driver is to detach the triggered touch-enabled display from the operating system response to receiving the start input device request.

[0132] Example 43 includes the apparatus of any one of examples 41 to 42, including or excluding optional features. In this example, the apparatus includes a user interface to receive a start request from the emulator and send a request to a graphics driver to display the virtual input device.

[0133] Example 44 includes the apparatus of any one of examples 41 to 43, including or excluding optional features. In this example, the emulator is to further generate private device data based on the intercepted touch data and send the private device data to a user interface to display feedback.

[0134] Example 45 includes the apparatus of any one of examples 41 to 44, including or excluding optional features. In this example, the apparatus includes a graphics processing unit. The graphics processing unit includes the gesture detector, the redirector, and the emulator.

[0135] Example 46 includes the apparatus of any one of examples 41 to 45, including or excluding optional features. In this example, the apparatus includes a keymap to be used by the emulator module to generate the hardware input data based on the intercepted touch data.

[0136] Example 47 includes the apparatus of any one of examples 41 to 46, including or excluding optional features. In this example, the display further includes private keys to be used to modify or relocate the virtual input device.

[0137] Example 48 includes the apparatus of any one of examples 41 to 47, including or excluding optional features. In this example, the apparatus is a mobile computing device.

[0138] Example 49 includes the apparatus of any one of examples 41 to 48, including or excluding optional features. In this example, the virtual input device

includes a virtual keyboard, a virtual touchpad, a writing window, or any combination thereof.

[0139] Example 50 includes the apparatus of any one of examples 41 to 49, including or excluding optional features. In this example, the virtual input device includes a writing window and the emulator is to recognize characters in the intercepted touch data and generate the hardware input data including keyboard strokes corresponding to the recognized characters.

[0140] Not all components, features, structures, characteristics, etc. described and illustrated herein need be included in a particular aspect or aspects. If the specification states a component, feature, structure, or characteristic “may”, “might”, “can” or “could” be included, for example, that particular component, feature, structure, or characteristic is not required to be included. If the specification or claim refers to “a” or “an” element, that does not mean there is only one of the element. If the specification or claims refer to “an additional” element, that does not preclude there being more than one of the additional element.

[0141] It is to be noted that, although some aspects have been described in reference to particular implementations, other implementations are possible according to some aspects. Additionally, the arrangement and/or order of circuit elements or other features illustrated in the drawings and/or described herein need not be arranged in the particular way illustrated and described. Many other arrangements are possible according to some aspects.

[0142] In each system shown in a figure, the elements in some cases may each have a same reference number or a different reference number to suggest that the elements represented could be different and/or similar. However, an element may be flexible enough to have different implementations and work with some or all of the systems shown or described herein. The various elements shown in the figures may be the same or different. Which one is referred to as a first element and which is called a second element is arbitrary.

[0143] It is to be understood that specifics in the aforementioned examples may be used anywhere in one or more aspects. For instance, all optional features of the computing device described above may also be implemented with respect to either

of the methods or the computer-readable medium described herein. Furthermore, although flow diagrams and/or state diagrams may have been used herein to describe aspects, the techniques are not limited to those diagrams or to corresponding descriptions herein. For example, flow need not move through each illustrated box or state or in exactly the same order as illustrated and described herein.

[0144] The present techniques are not restricted to the particular details listed herein. Indeed, those skilled in the art having the benefit of this disclosure will appreciate that many other variations from the foregoing description and drawings may be made within the scope of the present techniques. Accordingly, it is the following claims including any amendments thereto that define the scope of the present techniques.

Claims

What is claimed is:

1. A system for generating hardware device input, comprising:
a gesture detector to detect an input device trigger from one of two coupled touch-enabled displays;
a redirector to intercept touch data from a triggered touch-enabled display;
an emulator to generate hardware input data based on the intercepted touch data and send the hardware input data to an operating system; and
a user interface to display a virtual input device on the triggered touch-enabled display and receive touch data via the virtual input device.
2. The system of claim 1, wherein the user interface is to receive a start request from the emulator and send a start input device request to a graphics driver, wherein the graphics driver is to detach a display of the triggered touch-enabled display from the operating system in response to receiving the start input device request.
3. The system of claim 1, wherein the user interface is to receive a start request from the emulator and send a request to a graphics driver to display the virtual input device.
4. The system of claim 1, wherein the emulator is to further generate private device data based on the intercepted touch data and send the private device data to the virtual input device to display feedback.
5. The system of claim 1, comprising a graphics processing unit, the graphics processing unit comprising the gesture detector, the redirector, and the emulator.

6. The system of any combination of claims 1-5, comprising a keymap to be used by the emulator module to generate the hardware input data based on the intercepted touch data.

7. The system of any combination of claims 1-5, wherein the user interface is to display private keys to be used to modify or relocate the virtual input device.

8. The system of any combination of claims 1-5, wherein the virtual input device comprises a virtual keyboard, a virtual touchpad, a writing window, or any combination thereof.

9. The system of any combination of claims 1-5, wherein the virtual input device comprises a virtual keyboard and the hardware input data comprises Human Interface Device (HID) packets corresponding to keyboard strokes.

10. The system of any combination of claims 1-5, wherein the virtual input device comprises a writing window and the emulator is to recognize characters in the intercepted touch data and generate the hardware input data comprising keyboard strokes corresponding to the recognized characters.

11. A method for generating hardware device input, comprising:
monitoring a touch-enabled display for an input device trigger;
detecting the input device trigger from the touch-enabled display;
disabling the touch-enabled display at an operating system in response to detecting the input device trigger; and
displaying a virtual input device on the disabled display in response to detecting the input device trigger.

12. The method of claim 11, comprising:
receiving touch data from the virtual input device;

generating hardware input based on the received touch data; and
sending the hardware input to the operating system.

13. The method of claim 11, comprising detecting an adjustment gesture and adjusting the virtual input device based on the adjustment gesture.

14. The method of claim 11, comprising detecting the absence of an input device trigger and disabling the virtual input device in response to detecting the absence of the input device trigger.

15. The method of claim 11, wherein disabling the touch-enabled display comprises sending a message to the operating system that the touch-enabled display has been detached.

16. The method of any combination of claims 11-15, comprising detecting an exit trigger and disabling the virtual input device in response to detecting the exit trigger.

17. The method of any combination of claims 11-15, comprising:
receiving touch data from the virtual input device;
generating private device data based on the received touch data; and
sending the private device data to a user interface to provide feedback.

18. The method of any combination of claims 11-15, comprising displaying a user-draggable control on the disabled touch-enabled display and moving the virtual input device to a new location based on touch data received at the user-draggable control.

19. The method of any combination of claims 11-15, comprising monitoring the touch-enabled display for additional gestures and disabling the virtual input device in response to detecting an exit gesture.

20. The method of any combination of claims 11-15, comprising disabling the virtual input device in response to detecting a predefined placement of the touch-enabled display.

21. At least one computer readable medium for generating hardware input data having instructions stored therein that, in response to being executed on a computing device, cause the computing device to:

- detect an input device trigger from one of two coupled touch-enabled displays;
- intercept touch data from a triggered touch-enabled display;
- display a virtual input device on the triggered touch-enabled display;
- generate hardware input data based on the intercepted touch data; and
- send the hardware input data to an operating system.

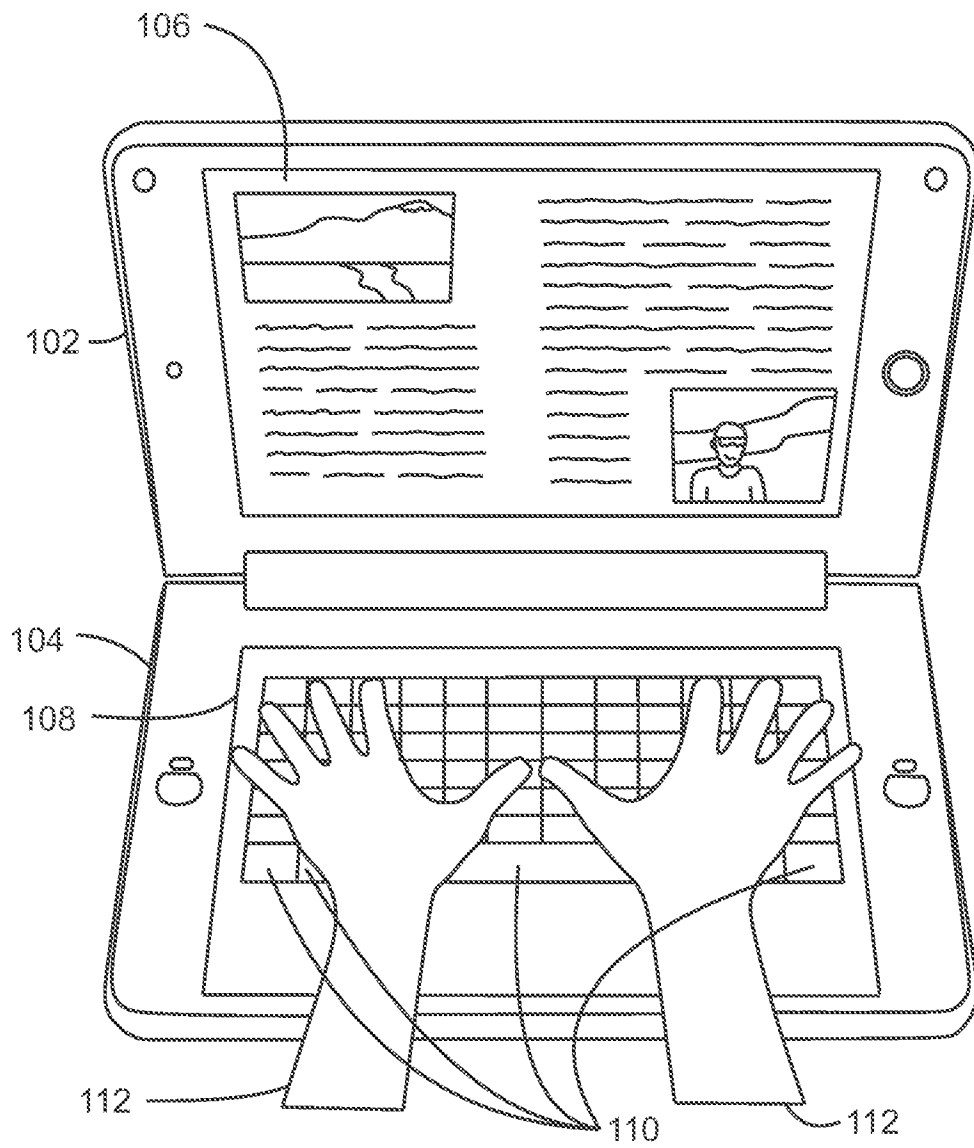
22. The at least one computer readable medium of claim 21, comprising instructions to receive a start request and send a start input device request to a graphics driver, wherein the graphics driver is to detach a display of the triggered touch-enabled display from the operating system in response to receiving the start input device request.

23. The at least one computer readable medium of claim 21, comprising instructions to receive a start request and send a request to a graphics driver to display the virtual input device.

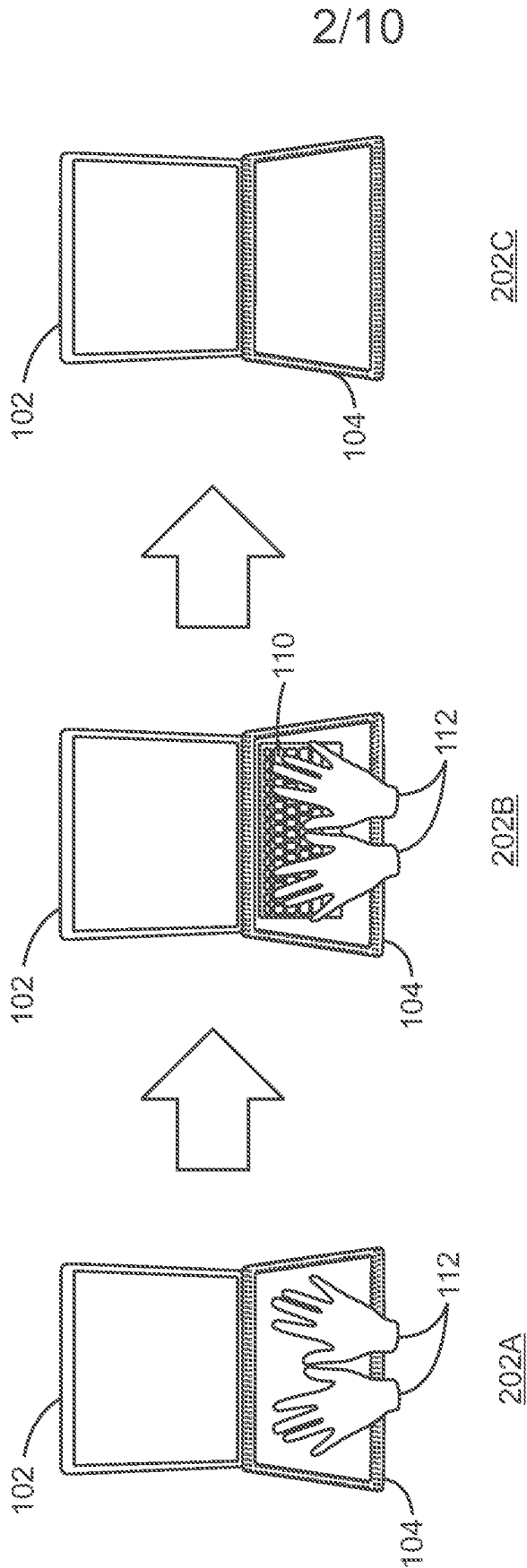
24. The at least one computer readable medium of any combination of claims 21-23, comprising instructions to generate private device data based on the intercepted touch data and send the private device data to the virtual input device to display feedback.

25. The at least one computer readable medium of any combination of claims 21-23, comprising instructions to disable the virtual input device in response to detecting an exit trigger.

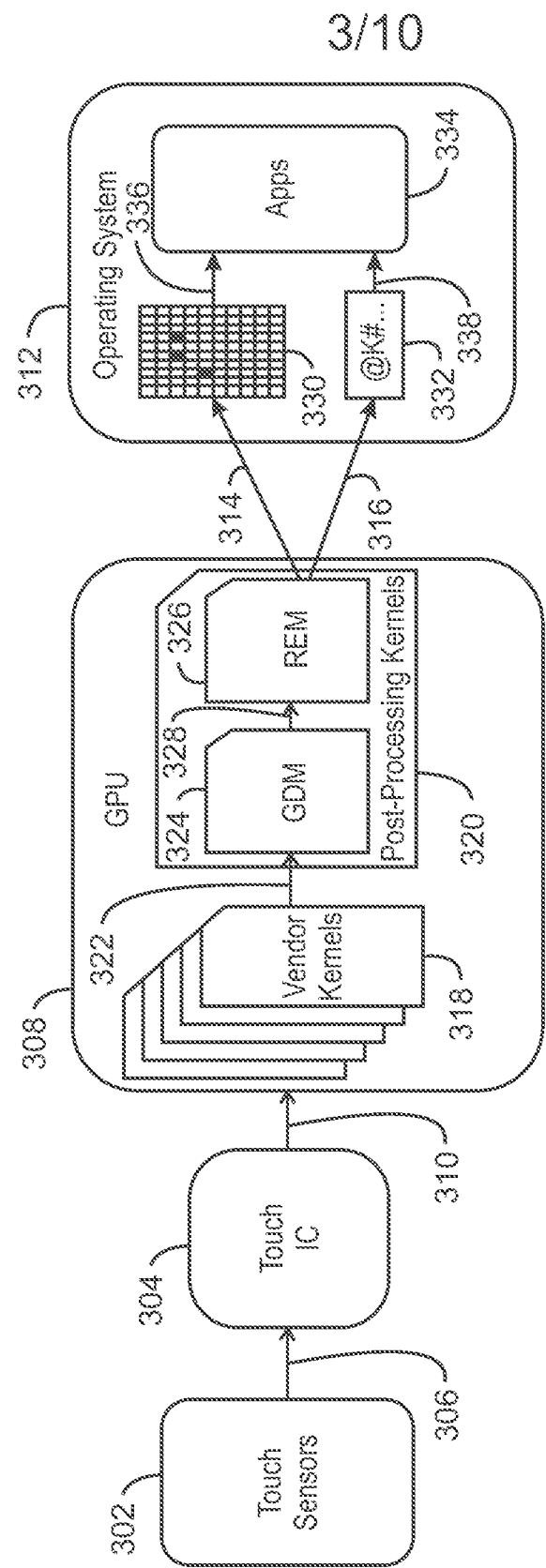
1/10



100
FIG. 1

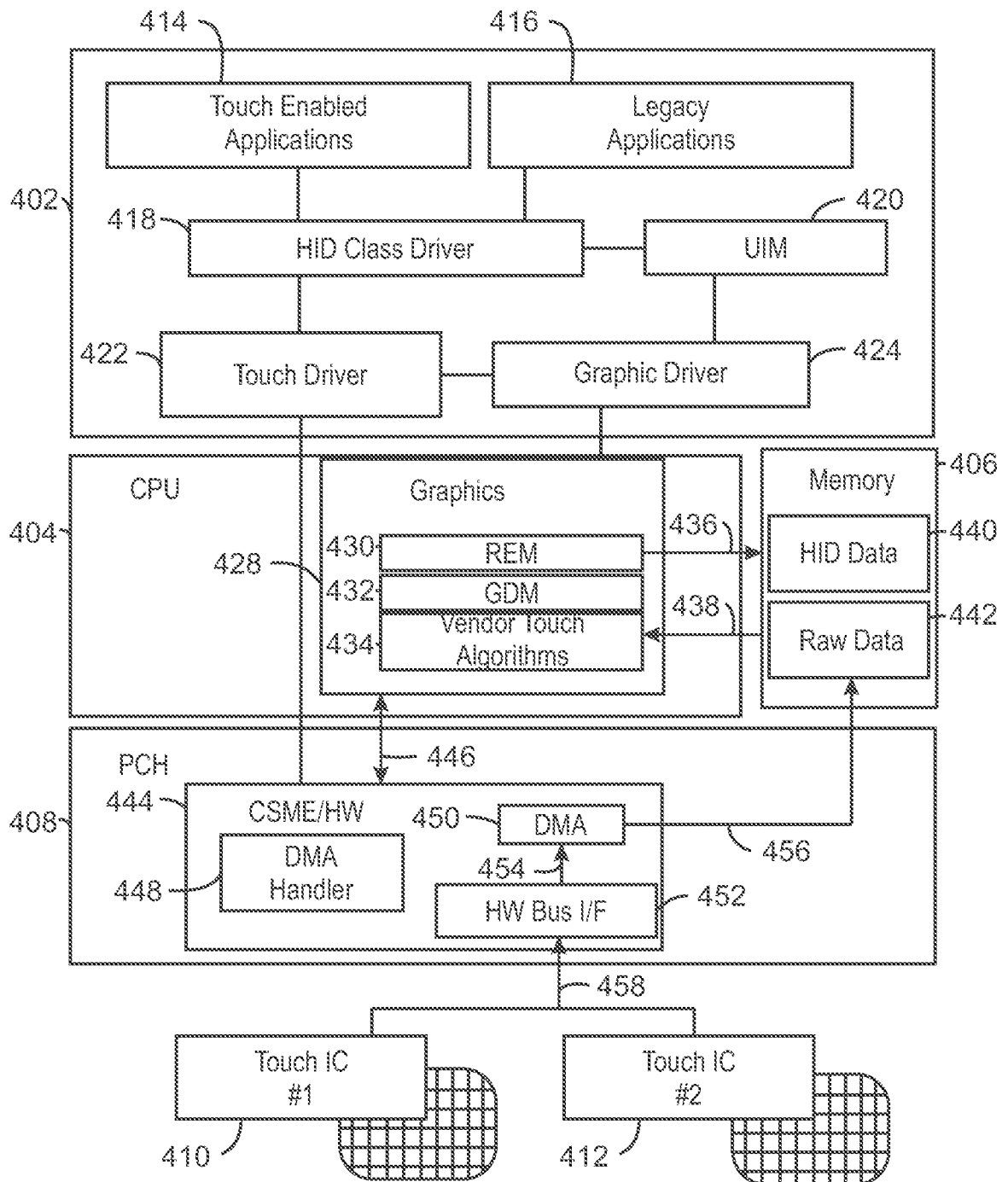


200
FIG. 2

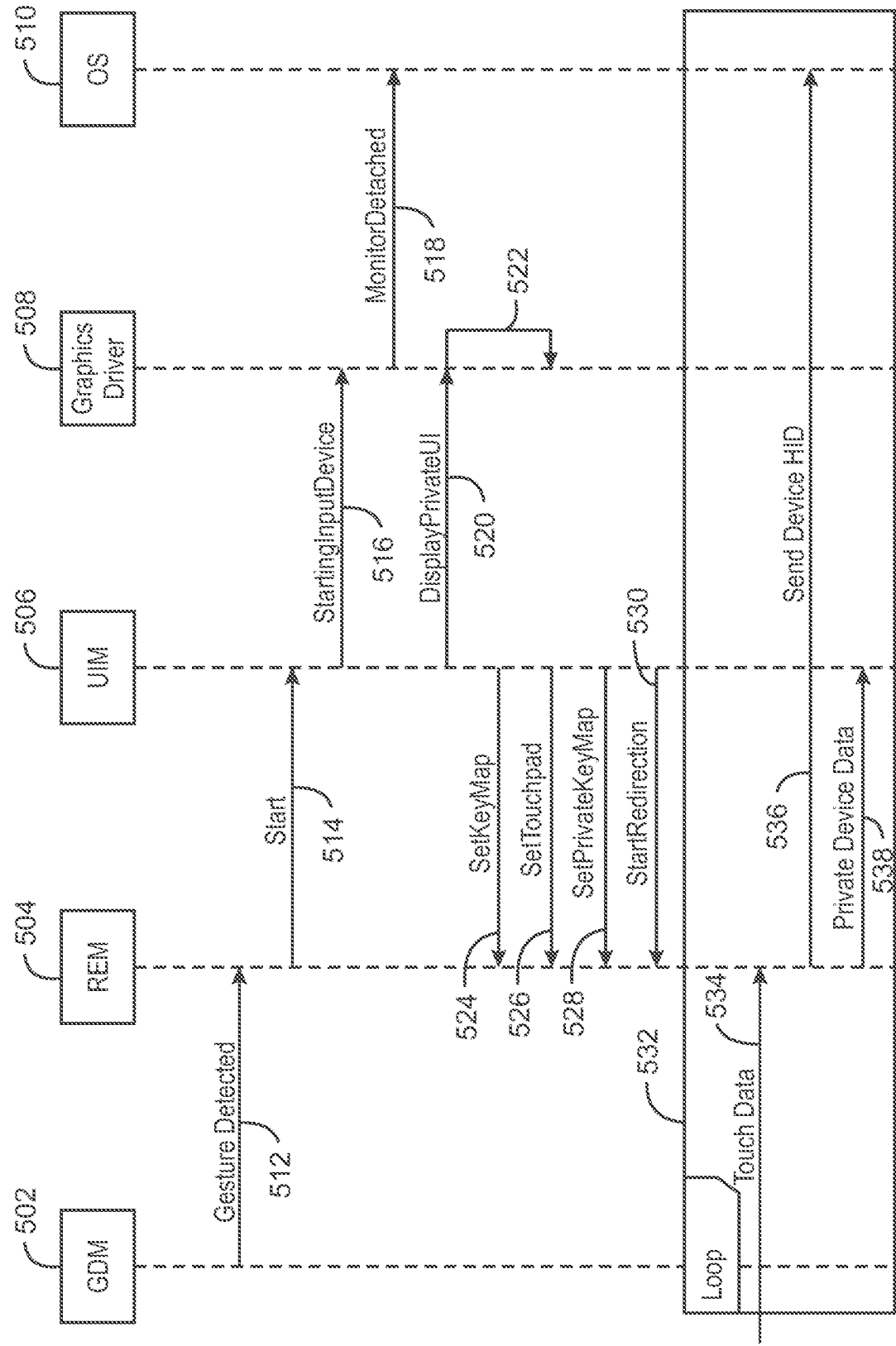


300
FIG. 3

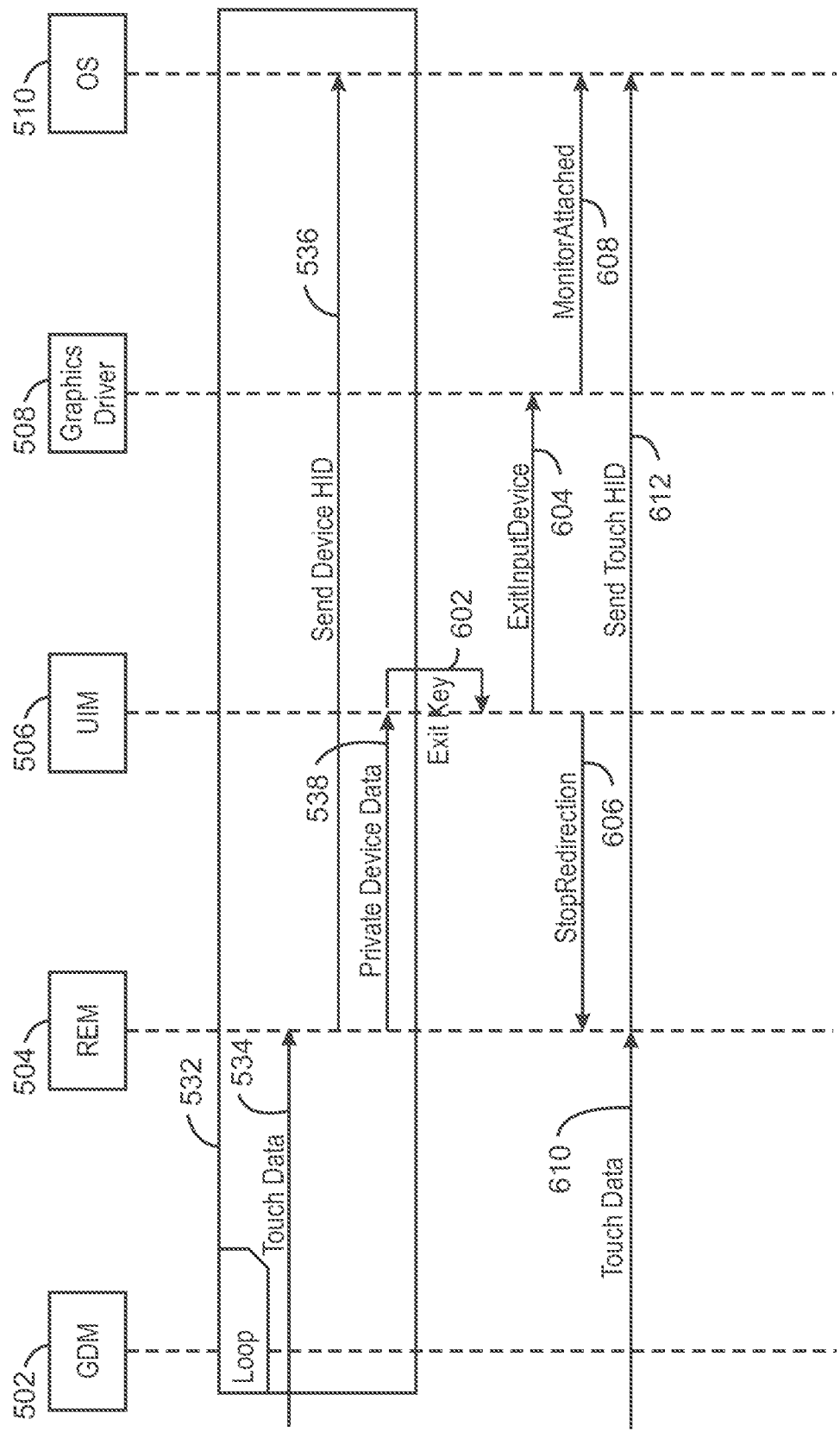
4/10



400
FIG. 4



500
FIG. 5



600

FIG. 6

7/10

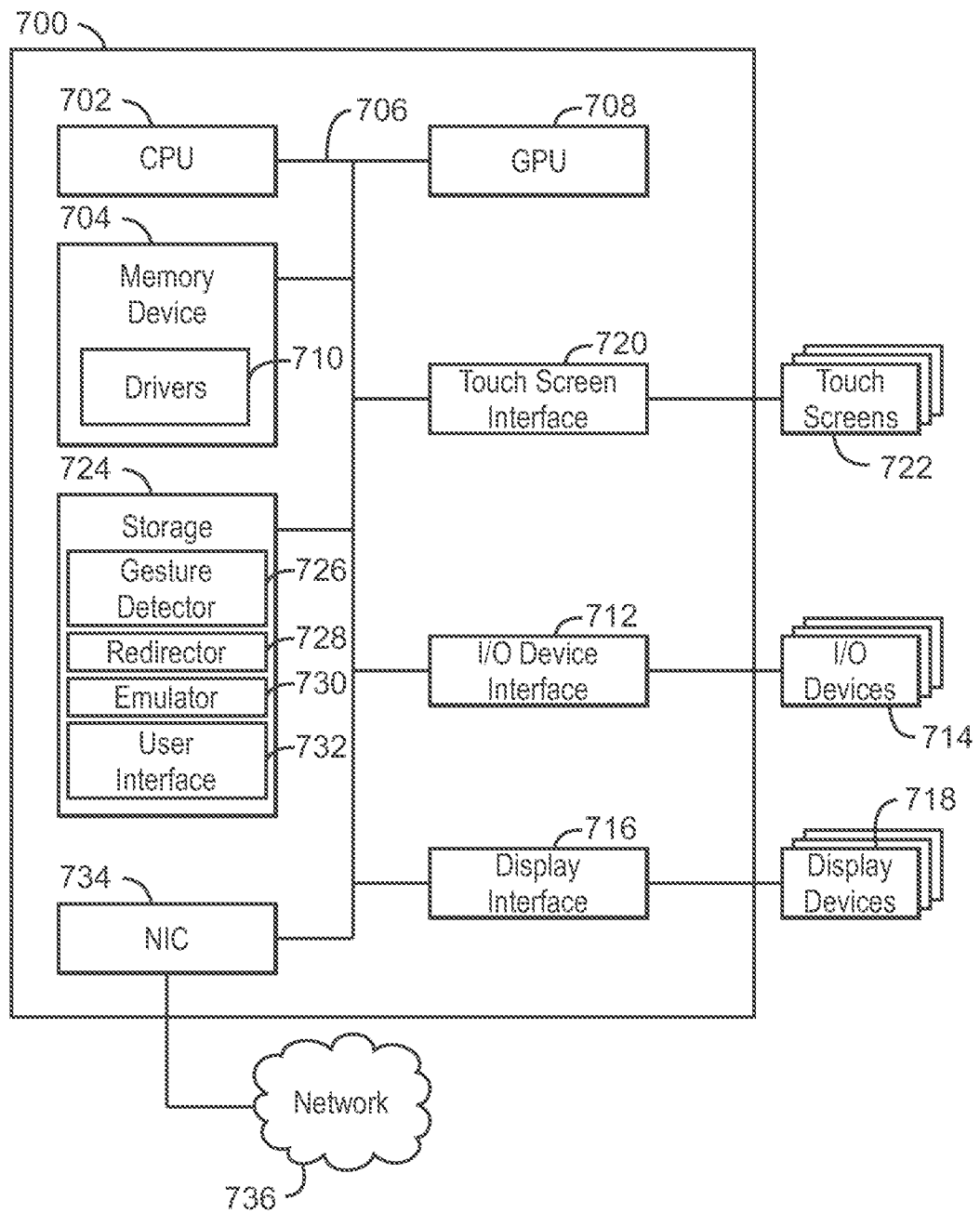


FIG. 7

8/10

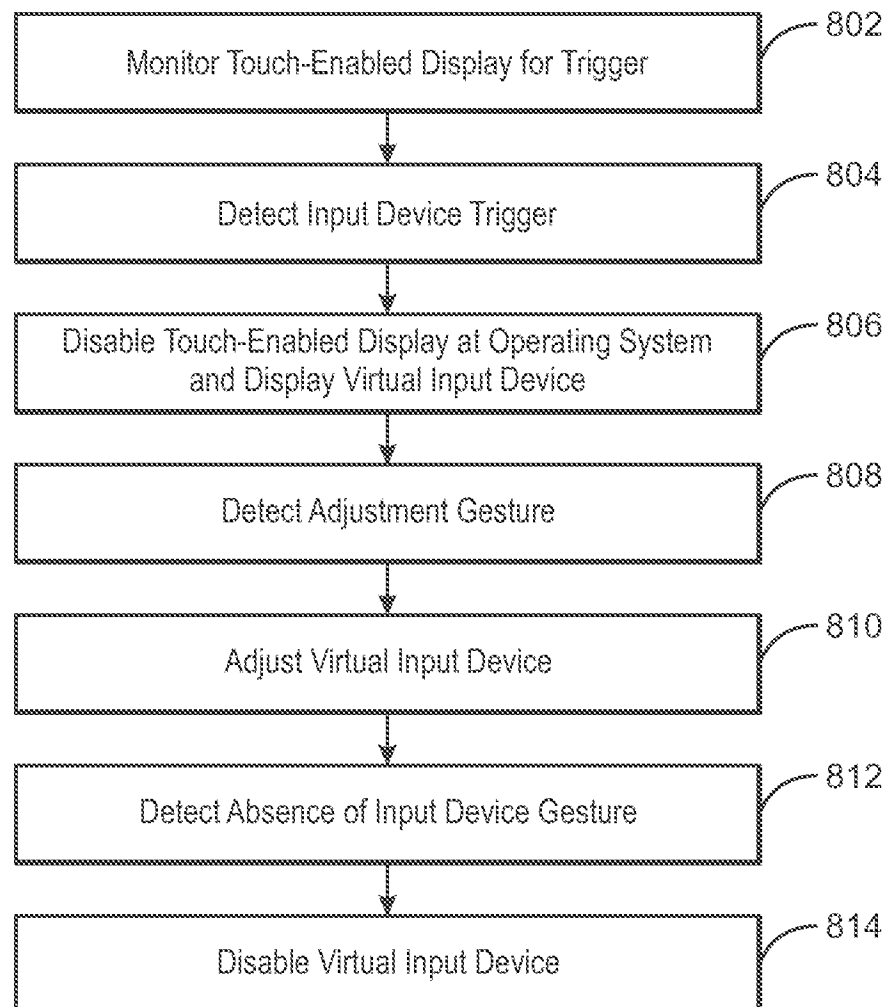
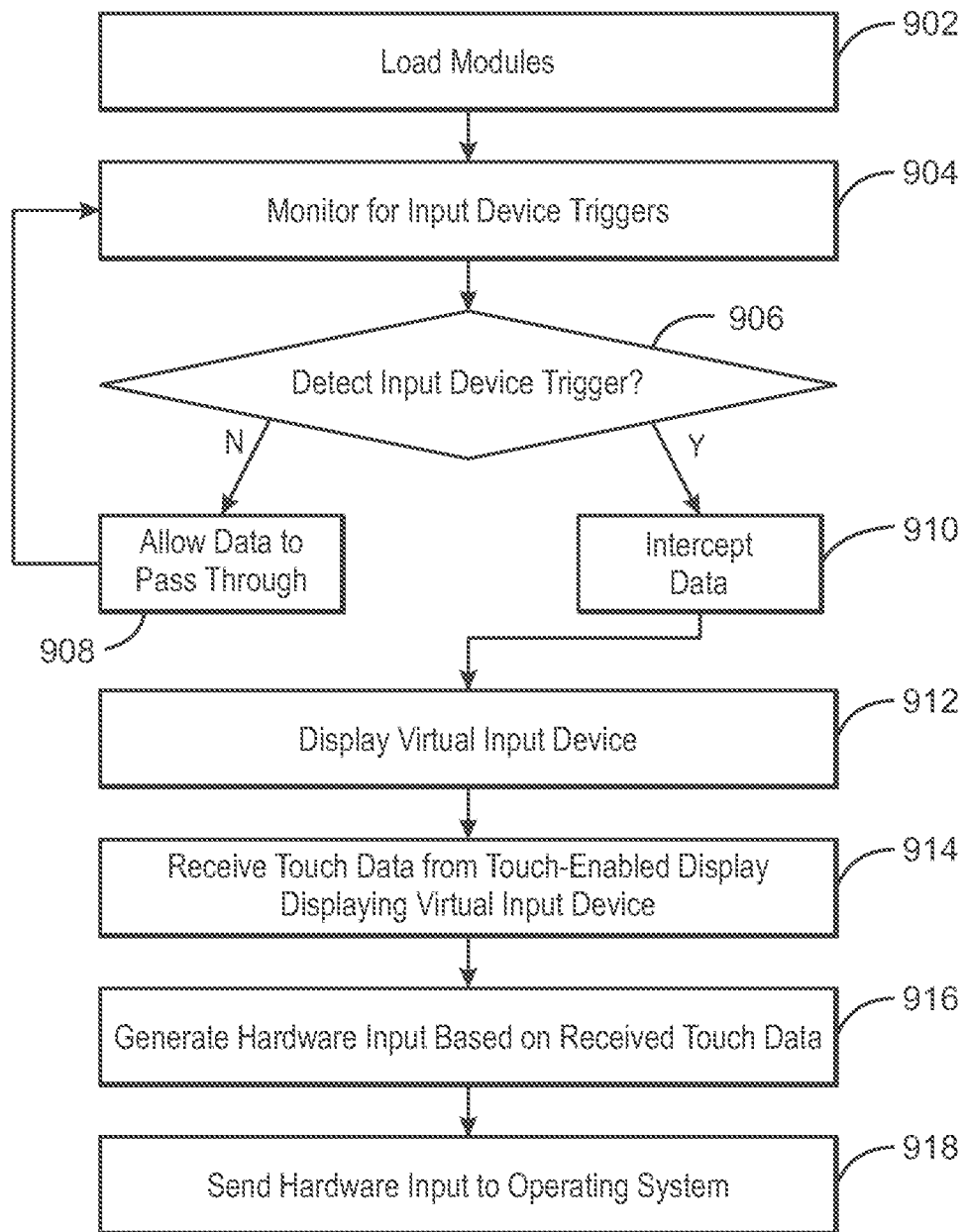
800

FIG. 8

9/10



900

FIG. 9

10/10

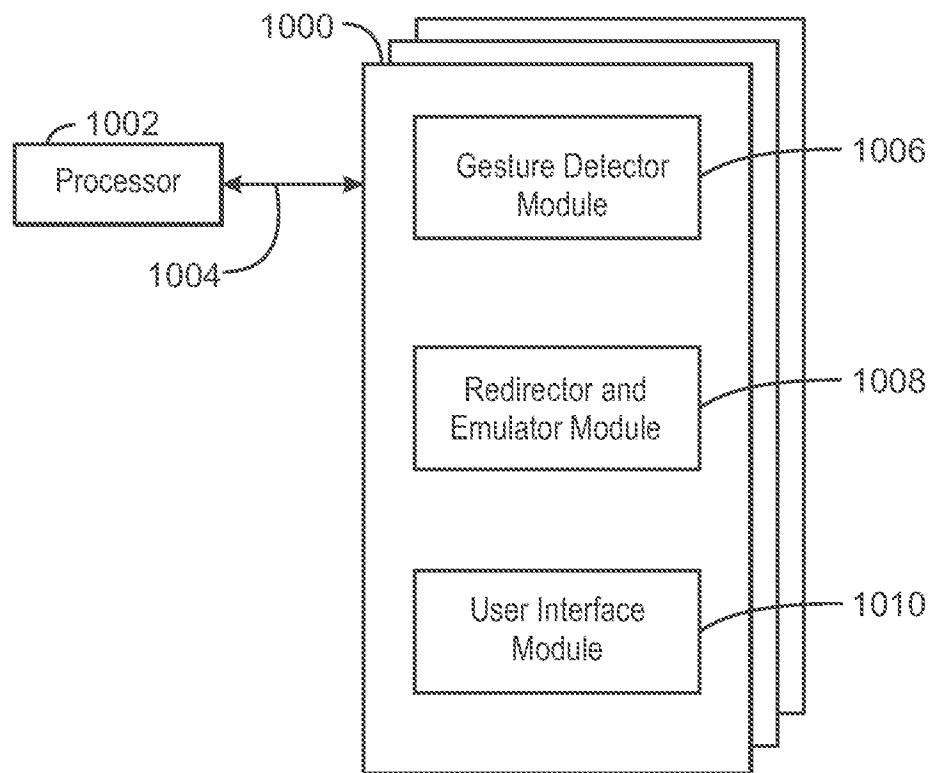


FIG. 10

A. CLASSIFICATION OF SUBJECT MATTER**G06F 3/0488(2013.01)i, G06F 3/0486(2013.01)i, G06F 3/14(2006.01)i**

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)

G06F 3/0488; G06F 3/02; G06F 3/048; G06F 3/041; G06F 3/0486; G06F 3/14

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: virtual keyboard, second touch display, gesture, detect, trigger, disable

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2011-0285631 A1 (AKIRA IMAMURA et al.) 24 November 2011 See paragraphs [0027]-[0030], [0039]-[0042], [0061]; claims 1, 9-10; and figures 1-2, 4, 9.	1-25
Y	US 2013-0127729 A1 (TIMOTHY J. MOSBY et al.) 23 May 2013 See paragraphs [0009]-[0018], [0024], [0040]-[0042]; claims 1, 9; and figures 1, 4.	1-25
A	US 2013-0257732 A1 (ROBERT DUFFIELD) 03 October 2013 See paragraphs [0021], [0063]-[0066]; and figures 1, 17.	1-25
A	US 2012-0260207 A1 (ANTON TRESKUNOV et al.) 11 October 2012 See paragraphs [0027]-[0028], [0047]; and figures 1, 2A-2C, 5.	1-25
A	US 2012-0050187 A1 (YANG CHENG CHEN) 01 March 2012 See paragraphs [0029], [0031]; and figures 3, 5.	1-25



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search

12 May 2017 (12.05.2017)

Date of mailing of the international search report

12 May 2017 (12.05.2017)

Name and mailing address of the ISA/KR

International Application Division

Korean Intellectual Property Office

189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea

Facsimile No. +82-42-481-8578

Authorized officer

LEE, Dong Yun

Telephone No. +82-42-481-8734



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2017/015033

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2011-0285631 A1	24/11/2011	JP 2011-248411 A	08/12/2011
US 2013-0127729 A1	23/05/2013	US 2009-0237361 A1 US 8358277 B2 US 8619036 B2	24/09/2009 22/01/2013 31/12/2013
US 2013-0257732 A1	03/10/2013	None	
US 2012-0260207 A1	11/10/2012	KR 10-2012-0114139 A US 9430145 B2	16/10/2012 30/08/2016
US 2012-0050187 A1	01/03/2012	TW 201209646 A	01/03/2012