

appointment. Many computer operating systems use sounds to alert the user of a variety of errors and events, but portable computers are sometimes used in public places where the sound must be turned off. The backlight can serve as a replacement alert in this situation. This feature is especially useful when alert dialogs are moved onto the touch screen from the main screen as disclosed in relation to **FIG. 10A**. Alert dialogs obstruct the view of the application data or interaction that may have raised the alert; by moving the alert to the touch screen and calling the user's attention to it by flashing the backlight, the present invention can improve the effectiveness of alert dialogs throughout the operating system.

[0114] If display **204** is a color display, then the system can flash backlight **206**, or color display **204** itself, in different colors to signal different types of alerts to the user. In this alternative, the use of color is analogous to the use of different sounds for audible alerts, and the touch screen may implement a mapping from standard sounds supplied by the operating system to standard color alerts.

[0115] Conversely, in some systems backlight **206** may be omitted to save space, power, or cost. On such systems, an alternate attention mechanism may be provided to alert the user when the touch screen is activated or changed to a new image with different active buttons. Suitable attention mechanisms include audible alerts, an icon or special cursor shape on the main display of the computer, an LED mounted near the touch screen, or a tactile feedback mechanism integrated with the touch screen.

[0116] The touch screen of the present invention must provide a mechanism for application software running on touch screen controller **216** or host computer **214** to create icons such as those shown on the iconic screen of **FIG. 4** and to create auxiliary and pop-up images such as those shown in **FIGS. 8-15**. Various mechanisms are possible to accomplish this.

[0117] If the software that manages an icon or pop-up screen resides in touch screen controller **216**, then the software has direct access to touch sensor **202** and display **204** via controllers **208** and **210**. The software on controller **216** can interpose its own images into the sequence of images it receives from host **214** for display. The software on controller **216** can also intercept finger touch information from sensor **202** before sending this information to host **214**. By these means, icons and pop-up screens can be implemented by software entirely in controller **216** with no participation by host **214**. Depending on the nature of interface **218**, controller **216** may also be able to send keystroke information to host **214** to allow its icons and pop-up screens to control host **214** by simulated keystrokes.

[0118] In an illustrative embodiment, many icons, auxiliary screens, and pop-up screens are implemented by various software applications running on host **214**. To coordinate access to the touch screen by these various applications, host **214** includes driver software that serves as a conduit between software applications and touch screen controller **216**.

[0119] **FIG. 16** illustrates an exemplary software architecture for the touch screen of the present invention. Touch screen architecture **1600** consists of hardware layer **1602**, driver layer **1604**, and application layer **1606**. Those skilled

in the art will recognize that many other software architectures are equally able to implement the user interface enhancements disclosed herein.

[0120] Hardware layer **1602** includes touch screen module **1610**, which in turn includes touch screen controller **216** of **FIG. 2**. Touch screen module **1610** connects to peripheral controller **1612**, which is included in host computer **214** of **FIG. 2**. Peripheral controller **1612** would be a USB host controller subsystem in the case that the USB protocol is used. Peripheral controller **1612** is operated by hardware driver **1614**. Hardware driver **1614** is supplied by the operating system of the computer and is not particular to the present invention.

[0121] Driver layer **1604** includes touch screen driver **1620**, which communicates with hardware driver **1614** to operate the touch screen hardware. Touch screen driver **1620** communicates with pointing device driver **1622**. Pointing device driver **1622** is supplied by the operating system and is responsible for operating mice and other pointing devices. When the touch sensor is operating as a conventional pointing device, touch screen driver **1620** converts sequences of finger positions reported by touch screen **1610** into motion signals similar to those produced by a mouse. Touch screen driver **1620** also examines the finger presence indication from touch screen **1610** to recognize finger tapping gestures. U.S. Pat. No. 5,543,591 discloses methods for computing tapping gestures on a touch pad sensor. These motion and gesture signals are conveyed to pointing device driver **1622** to cause cursor motion and clicking compatible with a mouse or conventional touch pad.

[0122] Touch screen driver **1620** also operates application programming interface (API) layer **1624**. Software applications running on the computer, represented in **FIG. 16** by software applications **1640**, **1642**, and **1644** in application layer **1606**, can use API **1624** to obtain special access to the touch screen. API **1624** exports a variety of touch pad and touch screen commands to the applications in application layer **1606**. These commands include requests for information about finger and "mouse" button activities on the touch sensor, as well as requests to override the cursor motion normally conveyed to pointing device driver **1622** with different cursor motion generated by the application based on finger movements. The API commands also include requests to display or update an icon on the iconic screen image, or to display or update a full-screen auxiliary or pop-up image.

[0123] Touch screen driver **1620** is responsible for deciding among conflicting API requests. For example, touch screen driver **1620** may consult pointing device driver **1622** or other operating system components to determine at all times which application, application window, or dialog has the input focus. If applications **1640** and **1642** each post a request to display an auxiliary screen image, it may be advantageous to have driver **1620** send the auxiliary image of application **1640** to touch screen **1610** only when application **1640** has the input focus. Similarly, driver **1620** sends the auxiliary image of application **1642** to the touch screen only when application **1642** has the input focus. If application **1644** has not posted an auxiliary image, then when application **1644** has the input focus, driver **1620** may display a default iconic screen image similar to that of **FIG. 4**.