December 2020

Keyboard and Mouse Input for Controlling a Remote Machine via A Mobile Device

Erik Jensen

Keyboard and Mouse Input for Controlling a Remote Machine via A Mobile Device

ABSTRACT

Controlling a remote machine via a mobile device can be difficult due to the challenges in capturing touchscreen input from the mobile device in a manner that can support the various functionalities of a full external keyboard and mouse/ trackpad. Directly relaying a character corresponding to the touchscreen text input to the remote machine does not support mobile input techniques such as swiping, gliding, or flicking across the keyboard, handwriting recognition, voice input, etc., and fails for languages that require multiple key presses to generate a single character. This disclosure describes techniques to provide users of mobile devices the full range of keyboard and mouse input functionality needed to control a remote machine accessed using the mobile device. A dual-mode touchscreen keyboard is provided to control a remote machine with a soft keyboard mode to generate key codes corresponding to key presses and a standard mode that functions similar to the usual mobile device keyboard. The techniques also support the full range of mouse input via direct touch or trackpad mode. The keyboard and mouse modes can be combined to relay input events that can include key presses and mouse clicks occurring simultaneously.

KEYWORDS

- Remote desktop
- Remote machine
- Soft keyboard
- Mouse input
- Trackpad gestures
- Keyboard layout
- Key code
- Key location

BACKGROUND

Users sometimes need to use their mobile device such as a smartphone to control a remote machine, such as a desktop or laptop computer. These remote machines are designed to be used with an external keyboard and mouse. However, a touchscreen-based keyboard is typically the only available input mechanism on mobile devices. Controlling a remote machine via a mobile device touchscreen keyboard can be inflexible and inefficient owing to various challenges encountered in entering text in the user's preferred language while simultaneously supporting various functionalities of a full external keyboard, such as keyboard shortcuts.

A basic approach involves directly relaying the character corresponding to the touchscreen text input to the remote machine. For example, if a user presses the 'Y' key on the touchscreen keyboard, the input relayed to the remote machine is the character 'Y.' Such an approach can work reasonably well for languages that use limited character sets, such as the Latin or Cyrillic alphabet. However, since the approach requires explicit individual key presses, it does not support other common mobile device input techniques, such as swiping, gliding, or flicking across the keyboard, handwriting recognition, etc.

Moreover, immediate direct relay of touchscreen key presses fails for languages such as Chinese and Japanese that require multiple key presses to generate a single character. For instance, when the mobile device keyboard is configured for Chinese input using Pinyin, the user is unable to send Chinese characters to the remote machine since the raw Latin characters are sent immediately prior to being composed into a Chinese character using multiple key presses on the Pinyin keyboard. Additionally, direct relay of typed characters does not work with applications on the remote machines (e.g., virtual machine) that receive input based on monitoring low-level events generated by key presses rather than the higher-level text character events.

Further, touchscreen keyboards on mobile devices are typically designed only for text entry. Therefore, touchscreen keyboards often lack various non-text keys, such as function keys (e.g., F1), navigation keys (e.g., PageUp), modifier keys (e.g., CTRL), etc. However, such keys are used often on remote machines that typically operate with full-fledged external keyboards. Lack of these keys on the touchscreen keyboard of a mobile device can prevent the user from using common input mechanisms, such as keyboard shortcuts, that rely on non-text key input. Installing a custom keyboard that includes commonly used non-text keys can help address the issue. However, users might be unwilling or unable to switch to a keyboard other than the one provided by the device operating system (OS). As an alternative, applications on the mobile device designed to interact with a remote machine can include buttons within their user interfaces (UIs) to provide the non-text keys missing from the touchscreen keyboard of the device OS.

Even if users employ a custom keyboard with an extended set of keys or use custom buttons within an application UI, the relay of individual text characters requires adjustments to support multiple simultaneous key presses (e.g., keyboard shortcuts, such as CTRL+C). In such cases, the simultaneous pressing of multiple keys cannot be relayed as individual text characters, thus requiring the relay of key press events that denote keys being pressed at multiple locations across a given keyboard layout. However, if the keyboard layouts of the mobile device and the remote machine do not match, the key press event relayed from the mobile device can result in the wrong key input on the remote machine.

Apart from keyboard limitations, controlling a remote machine from a mobile device can be difficult because of the need to use the device touchscreen to carry out actions that are normally performed using a mouse or trackpad. For instance, the display of a remote desktop is typically larger than that of the mobile device, thus requiring scaling to be turned off in order to enable viewing only a part of the screen of the remote machine. However, focusing on part of the remote screen relies on a mouse or trackpad to operate scrollbars in individual windows or to perform bump scrolling in full-screen mode. However, mobile devices typically provide only touch-based input, thus making it difficult or impossible to perform various mouse or trackpad operations, such as dragging, context clicking, etc.

DESCRIPTION

This disclosure describes techniques to provide users of mobile devices the full range of keyboard and mouse input functionality needed to control a remote machine accessed using the mobile device. To support full keyboard input, the techniques involve augmenting the touchscreen keyboard within a web browser (or other application) used to control a remote machine. Specifically, the augmented keyboard supports two separate input modes: (i) a soft keyboard mode that generates and relays key codes corresponding to key presses, and (ii) the standard mode that functions like the usual mobile device touchscreen keyboard with a corresponding text field.

The layout of the keys in the soft keyboard mode is based on keyboard layouts and corresponding key code values for the remote machine obtained using any suitable mechanism, such as a function specified within an application programming interface (API). Compared to the corresponding external keyboard, the presentation of the layout within the soft keyboard mode is adjusted to accommodate being displayed on a relatively small touchscreen of a mobile device. The adjustments include:

1. If a layout contains a large number of keys (e.g., 15) across the top row, the location of one (or more) of the keys is moved to another row to avoid the top row from being too cramped, especially when the mobile device is in portrait mode. The key chosen to be moved can be one used less frequently, such as the currency symbol for Yen (i.e., '¥') in the Japanese keyboard layout. Such an adjustment enables all layouts presented within the soft keyboard mode to contain a limited number of keys (e.g., 14) in the top row. When a key from the top row is moved, it is presented in a location where keys are typically present in other layouts, but unused in the present layout. For instance, keyboard layouts that contain the Yen key include only one of the two possible Backslash keys present in other layouts, making it possible to move the Yen symbol to the typical location of the second Backslash key.
2. The Space key and/or the left Shift key and/or the right Shift key are sized based on the presence of other keys in the corresponding row of the keyboard layout. For instance, the left Shift key is expanded if the corresponding row of the layout does not contain an IntlBackslash key.
3. The layout is split across two sections toggled using the 'Fn' key. One section includes alphanumeric keys and common modifier keys, such as Shift, Control, etc. The other section includes additional keys, such as function keys, navigation keys, numeric keypad keys, and special keys (e.g., Print Screen, Scroll Lock, Pause, etc.).
4. The layout includes an additional key (such as '⋮') to enable the user to switch between the soft keyboard mode and the standard mode with a text entry field.

Adjustments to key locations and sizes as above are kept to a minimum. As a consequence, the layout is presented such that keys remain close to their standard locations on a full external keyboard. The special keys to toggle the keyboard sections (Fn) and the input mode (⋮) can be removed if the functionality is provided via other suitable mechanisms, such as a toolbar or menu item within an application. If one or both of the toggle keys are removed from the bottom row, their place is filled by expanding the Space key as described above.

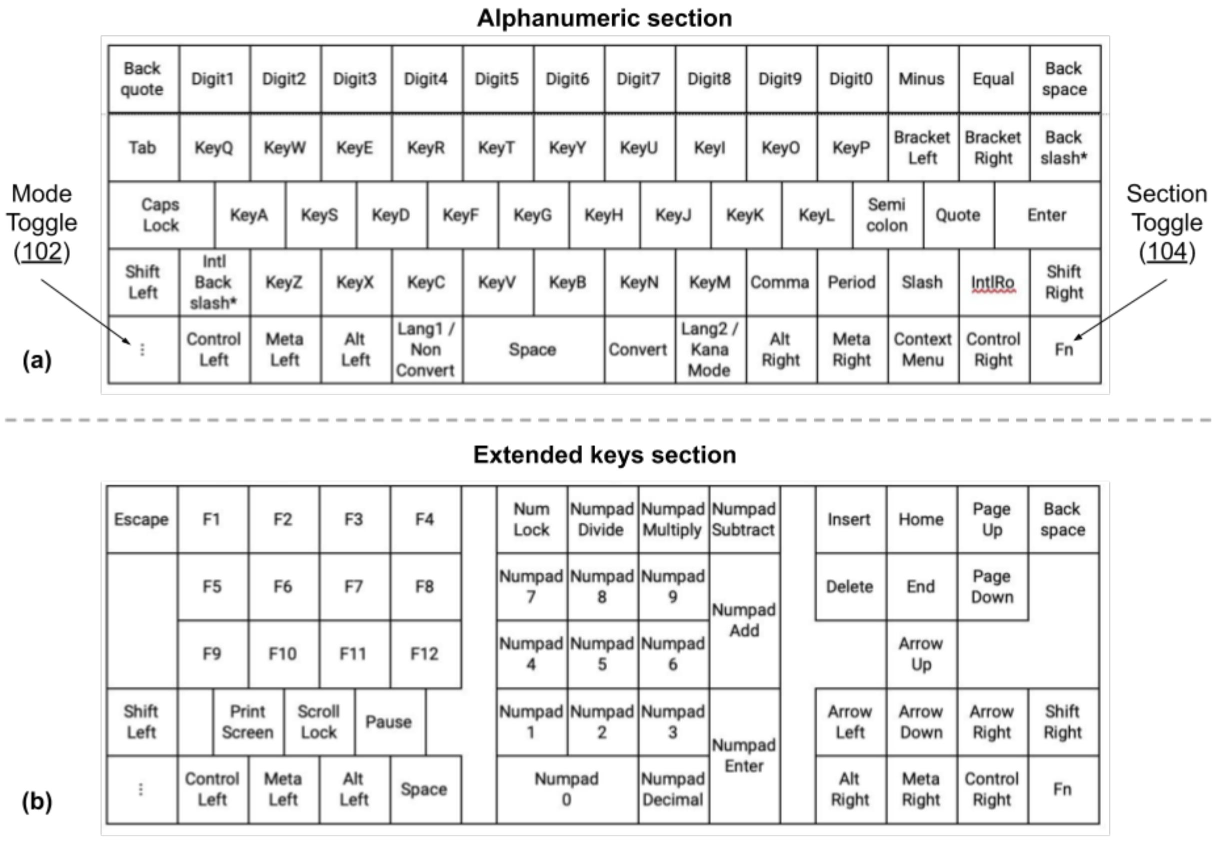

Fig. 1: Layout of all keys for a full-featured soft keyboard organized into two sections

Fig. 1 shows a full-featured keyboard layout generated as described above with Fig. 1(a) (top) showing the section with the alphanumeric keys and main modifier keys, and Fig. 1(b) (bottom) showing the section with the extended set of keys. The section toggle button “Fn” (104) on the right of the bottommost row of keys can be used to switch between the two sections. Fig 1(a) further shows the mode toggle button “:” (102) on the left of the bottommost row of keys. The mode toggle button switches between the key code based soft keyboard mode and the standard mode for multiple character text entry. As described above, the full-featured layout is adjusted for key locations within specific layouts. Fig. 2 shows the alphanumeric section of the keyboard layout adjusted for the set of keys present in a standard US keyboard layout.

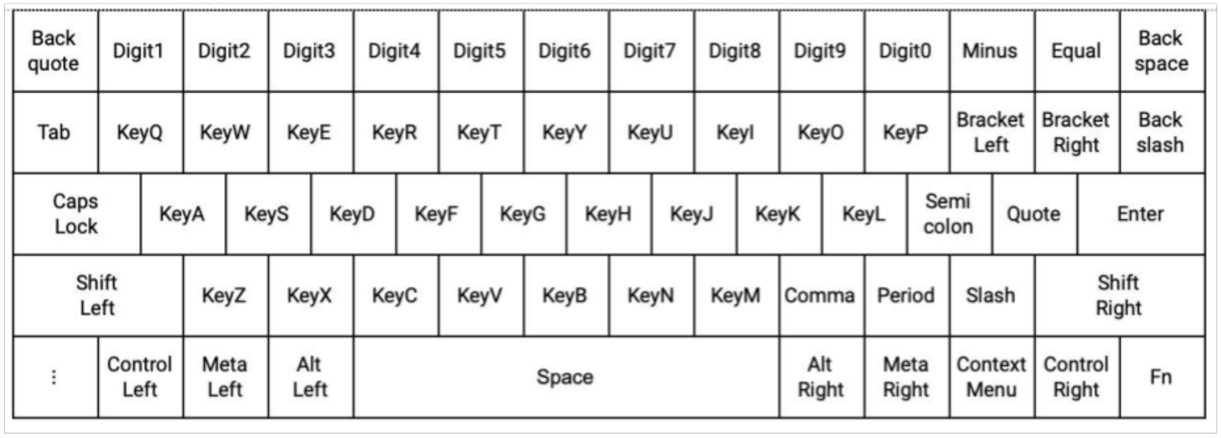

Fig. 2: Keyboard layout for the alphanumeric section of the standard US keyboard

Since key presses within the soft keyboard mode generate key codes instead of key values (specific characters corresponding to a key), the key mappings in the keyboard layout presented on the mobile device screen need to match those of the remote machine. However, maintaining the mappings for all possible languages and keyboard layouts can be inefficient and resource intensive. To avoid such a cumbersome mechanism, the soft keyboard mode can include an additional API protocol message that enables the client application to seek the entire keyboard layout of the remote machine. Alternatively, or in addition, the host can use the protocol message to inform the client in case the host keyboard layout changes during a session.

When the user presses a modifier key, such as Shift, within the soft keyboard mode, the keyboard layout presented on the screen is altered according to the modified state. For example, pressing either of the Shift keys on a standard US QWERTY layout results in the ‘4’ key changing to the ‘$’ key. To support such an operation, the host can send an updated keymap whenever a modifier key is pressed. Alternatively, the client can retrieve all possible keymaps for the various modifier key press combinations up front at the start of the session. Whether to transfer all keymaps up front or on an as-needed basis can be dependent on the platform or the application.

The operation of the modifier keys is independent of the location within the layout presented for the soft keyboard mode. When the user presses a modifier key in the soft keyboard mode, it can operate in a *sticky* manner, remaining pressed until the user presses it again or presses a non-modifier key. While the key remains in the sticky pressed state, the styling of the key can be changed to provide a visual indicator that it remains pressed. Sticky operation permits pressing keys in sequence to generate the code equivalent to simultaneous presses of multiple keys (e.g., Shift + F11, CTRL + Shift + V, etc.).

Operations that require modifier keys to be pressed for multiple successive inputs (e.g., ALT codes) can be supported by locking the pressed state of a modifier key via double tapping. When locked as such, the modifier key remains in the pressed state until it is tapped again to release the lock. Operations that require pressing a modifier key by itself can be achieved by triple tapping the modifier with the first tap to press, second tap to lock, and third tap to release the lock and generate the corresponding key press code.

On devices that support multi-touch capabilities, the user can alternatively hold down one or more modifier keys while pressing the modified key with a different finger. The soft keyboard mode can further support multi-touch operation for simultaneous presses of two or more non-modifier keys as well.

While the soft keyboard mode enables the user to access the full range of keyboard input capabilities available on the remote machine, it does not support standard input techniques, such as swipe, glide, flick, etc., typical of mobile device keyboards. Therefore, the soft keyboard mode can be cumbersome and inconvenient when interaction with the remote machine requires the user to enter large amounts of standard text. In such cases, the user can switch to the standard mode that permits using the familiar mobile device keyboard layout to compose text within a standard text field. The text field can be shown at a suitable on-screen location, such as directly above the keyboard or at the bottom of the screen. In the standard mode, the user can enter text in the text field using any available input technique such as voice dictation, handwriting recognition, gesture-based input (e.g., swipe, glide, flick), Input Method Editor (IME) interfaces such as Pinyin, etc.

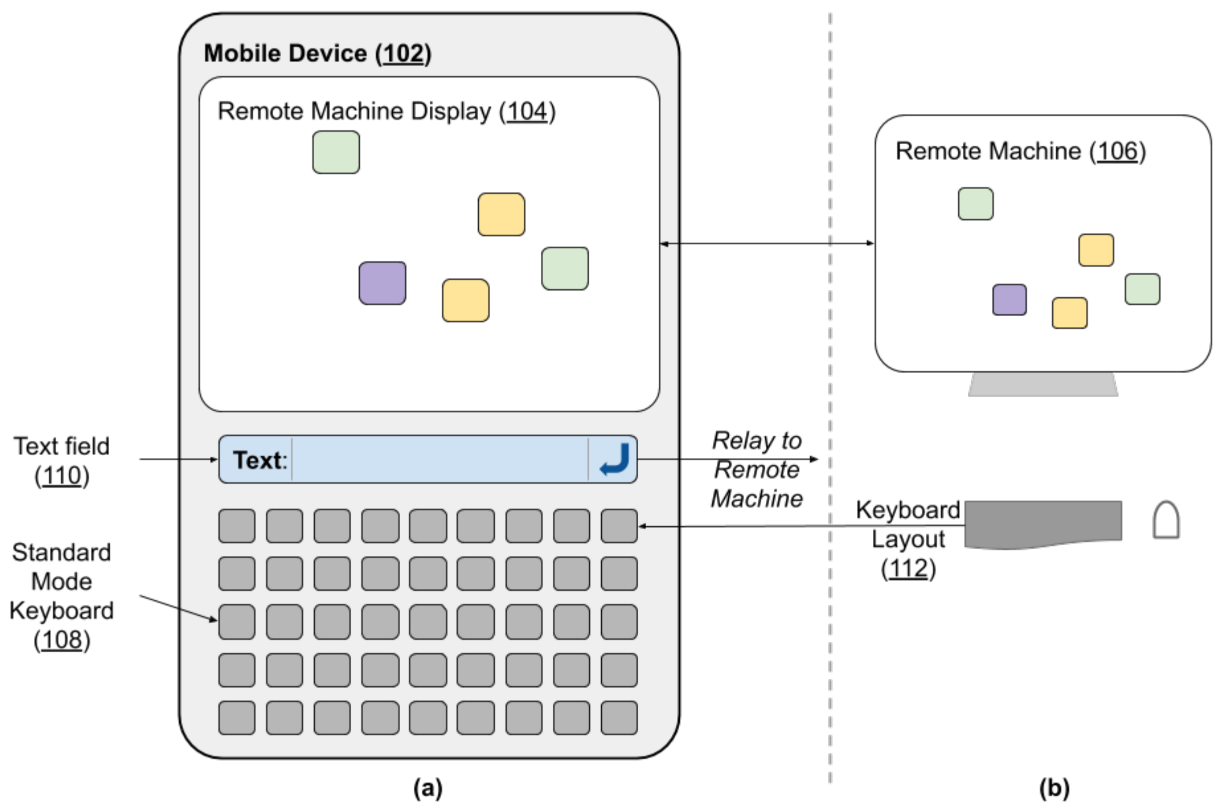

Fig. 3: Capturing text input in standard keyboard mode prior to remote relay

Fig. 3 shows an operational implementation of the standard mode described above. A user connects to a remote machine (106) via a mobile device (102) such that the remote machine display is mirrored (104) on the device. The standard mode keyboard (108) layout is generated and presented. In this mode, the displayed keyboard is the keyboard/ layout/ input method that has been configured on the mobile device for regular text input, and is not based on the remote machine's keyboard layout. In another mode, the keyboard layout on the mobile device can be based on the keyboard layout (112) of the remote machine. A text field (110) is used to capture the text entered via the keyboard using any suitable interaction technique.

Once done composing the text in the standard mode, the user can relay it to the remote machine all at once as Unicode text events corresponding to the text characters. If the user chooses to relay the text when the text field is empty, the action can be considered equivalent to pressing the Return (⏎) key and results in relaying the newline character to the remote machine.

In case the user is using an external keyboard connected to the mobile device, input from the external keyboard can be handled according to the active keyboard mode connected to the remote machine. In the soft keyboard mode, input from the external keyboard can send key codes based solely on key location, which are interpreted by the remote machine according to its configured keyboard layout or IME. On the other hand, in the standard mode, input from the external keyboard can be shown as text within the standard text field shown in that mode, in this case using the configured keyboard layout or IME of the local device.

The described techniques can support additional uses, such as a helper using a mobile device to provide remote assistance to a user on another machine. In such cases, the touchscreen keyboard shown on the helper's device in the soft keyboard mode can facilitate seeing the layout of the user's keyboard while permitting the helper to use the standard mode to type text based input via the local keyboard layout.

In addition to the keyboard, the touch input on the screen of the mobile device can be mapped to carry out mouse or trackpad based operations on the remote machine. In direct touch mode, in cases where the remote machine operating system supports it, equivalent touch events are generated on the remote machine, rather than converting them to mouse events. The handling of touch gestures (pinch, swipe, et cetera) in such a setup occurs on the remote side. The described techniques for translating taps to clicks/mouse actions only apply when it is not possible to inject touch events (due to lack of support by the remote machine OS.

In the usual direct touch mode of the device, the mouse cursor of the remote machine need not be shown since it cannot be positioned without clicking. Instead, tapping the screen can be translated directly to a click on the corresponding part of the screen of the remote machine by generating a cursor move followed by a click. The tap-translated-to-click is indicated by an appropriate feedback mechanism, such as a gray-circle touch animation. Similarly, right and middle clicks can be generated based on tapping with two and three fingers, respectively. However, such an operation requires users to ensure that the first finger to touch the device screen is located at the point at which the right or middle click is desired. Alternatively, the click location can be set at a screen location in the middle of the user's fingers touching the device screen.

The direct touch mode can support other common mouse or trackpad operations by mapping them appropriately to suitable touch mechanisms. For instance, initiating the dragging functionality can be achieved with press-and-hold or tap-and-a-half. In the former case, the user presses and holds the touchscreen at the location of the item to be dragged until receiving feedback, such as gray-circle touch animation, that the drag operation has started. The user can then move their finger to the new location where the item is to be dragged and lift their finger to drop. Alternatively, the user can indicate the time to drag by a standard screen tap and then quickly placing their finger back on the screen to start the drag operation and subsequently dragging their finger across the screen to drag the item to the new location. The screen can pan automatically when the user's finger reaches the edge of the screen. Further, either of the dragging approaches can be used with two and three fingers to perform right- and middle-button drags, respectively.

Similarly, scrolling can be performed with two fingers with the scroll beginning at the point of contact of the first finger to touch the screen; panning the display can be mapped to swiping with a single finger; and pinching and spreading can adjust the zoom level in relation to the focus point located between the fingers touching the device screen.

Alternatively, or in addition, mouse operations can be supported by a trackpad mode in which the user can treat the part of the touchscreen of the mobile device showing the remote machine akin to a trackpad. The screen displays a mouse that is moved based on the relative direction and distance of the user's single-finger swipes on the screen of the mobile device. Swiping with two fingers can generate scroll events instead of moving the displayed mouse. Tapping anywhere with one, two, or three fingers can result in the corresponding click (left click, right click, and middle click, respectively) at the location of the mouse on the screen without moving the mouse. Dragging in the trackpad mode can work similar to the corresponding dragging operations in the direct touch mode as described above. However, unlike the direct touch mode, dragging is performed on the item over which the on-screen mouse is positioned. Once dragging is initiated, the user can use one finger to move the on-screen mouse, thus dragging the item underneath.

The trackpad mode can support standard pinch gestures to control zoom level. When zoomed in, the viewport can stay centered on the on-screen mouse cursor. Moving the mouse can pan the display to keep the mouse centered. However, the mouse cursor is moved to the edge of the screen upon reaching the edge of the display of the remote machine to avoid creating a large blank area resulting from moving the edge of the remote display to the middle of the display of the mobile device.

The direct touch mode can match the typical interaction mechanisms of mobile devices while the trackpad mode can provide more precise mouse positioning to click smaller-sized targets, especially when at lower zoom levels. As such, the direct touch mode can be intuitive for those who regularly use mobile devices, and the trackpad mode can provide the full range of mouse operations for experienced users of physical trackpads.

The keyboard and mouse modes described above can be combined to relay input events consisting of key presses and mouse clicks occurring simultaneously. For example, tapping the CTRL modifier key and then clicking in the trackpad mode can be utilized to relay a CTRL+click to the remote machine. Further, locking a modifier key can modify keyboard presses in the first described keyboard mode (and not in the mode when composing text in the text field), and mouse clicks in either mouse mode.

The described techniques can be implemented in a mobile web browser or other application that facilitates control of a remote machine (remote desktop). Implementation of the techniques can enhance the user experience (UX) of controlling a remote machine via small-screen touchscreen devices, such as smartphones, tablets, etc.

CONCLUSION

This disclosure describes techniques to provide users of mobile devices the full range of keyboard and mouse input functionality needed to control a remote machine accessed using the mobile device. A dual-mode touchscreen keyboard is provided to control a remote machine with a soft keyboard mode to generate key codes corresponding to key presses and a standard mode that functions similar to the usual mobile device keyboard. The techniques also support the full range of mouse input via direct touch or trackpad mode. The keyboard and mouse modes can be combined to relay input events that can include key presses and mouse clicks occurring simultaneously. The described techniques can be implemented in a mobile web browser or other application that facilitates control of a remote machine (remote desktop).

REFERENCES

1. "UI Events KeyboardEvent code Values" W3C Candidate Recommendation, 01 June 2017, available online at https://www.w3.org/TR/uievents-code/
2. "Click, drag, or scroll with the touchpad" available online at https://help.gnome.org/users/gnome-help/stable/mouse-touchpad-click.html.en#:~:text=To%20double%2Dclick%2C%20tap%20twice,with%20two%20fingers%20at%20once
3. "KeyEvent" Android Developers Documentation available online at https://developer.android.com/reference/android/view/KeyEvent