# KeyNav.
## By Oscar Blanco.

Since I was a little kid and saw Tony Stark working with hand gestures and voice commands, a desire in me to develop tools interact with my computer in a similar way, was born. So I decided to discard the use of a mouse and use only my keyboard to begin with… and that's how KeyNav (stands for keyboard navigation) was conceived.

Here is a brief description of how it works for those who don't know much about coding.

Essentially, when KeyNav is activated, you can think that your normal keyboard behavior is overwritten, e.g. you can't press alt+f4 to close programs, or alt+tab, crazy, I know but please, bear with me for a little. In that state, you can only use the keys that are assigned to a "mouse function" which are a fixed set of shortcuts that I will show later, later on, -I may add a way to customize this with no need of modifying the code, but for now you can do it modifying the configFile.ini, if you have some experience coding-.

So, to be able to use your mouse again like you always have, you have to press the cobination assigned to the "alternateMode" in the .ini file, or the one I will mention later. So you can alternate the state of the program each time.

**Click functionalities:**

These allow you to use j, k, l to simulate click actions

f: for single left click
i: for middle click
o: for right click

**Scroll functionalities:**

These will allow you to use the h, j, k, l keys to scroll through the application you are using:

j: scroll down
k: scroll up
h: scroll left
l: scroll right

**Move cursor by zones:**

Used for occasions where you want to move your cursor to a determined part of the screen without much detail, for example when you want to scroll a different tab than the one your cursor is currently on. When you just open the program, the screen is divided in 9, then you select a section, then that section is divided in 9, and so on…

q: top-left
w: top-middle
e: top-right
a: middle-left
s: middle-middle
d: middle-right
z: bottom-left
x: bottom-middle
c: bottom-right

When you select a zone, you'll notice that it kind of gets stuck, that is because you have selected a small zone, and in those moments you need to reset the cursor, to its original zone, and for that you use:

r: reset cursor

**Clicks press without release:**

In those ocations when you need to press and hold a click, you must use this tool.

shift+f: press/release left click
shift+i: press/release middle click

shift+o: press/release right click

Note that, when you press and then release a click, it generates a click, so be careful with that, I didn’t find a way around it sorry.

And if you think you pressed a click and don’t remember which, you can release all the clicks with:

shift+r: resetClicks

**<u>Move cursor in certain directions:</u>**

This is used when you want to move your cursor with more detail.

Shift + h = move mouse left
Shift + j = move mouse down
Shift + k = move mouse up
Shift + l= move mouse right

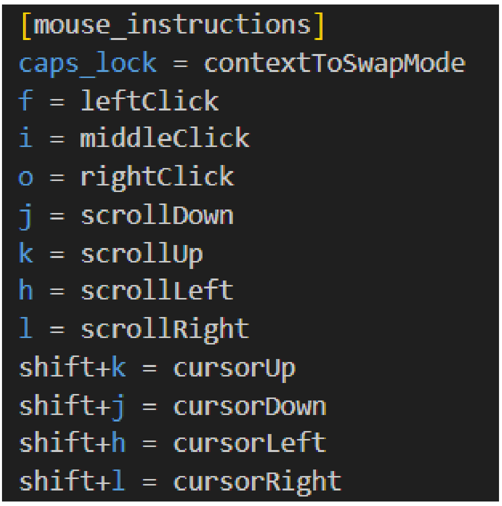
I know this may be weird for some people, but this is based on the hotkeys of a software I use for coding, again, I will add a way to edit the commands for each functionality, or you can modify the .ini file.

## <u>CONFIGURATION:</u>

The software is designed in a way that you can configure what combination of keys executes each action in every mode. The default configuration is based on a Spanish keyboard distribution, so if you have another keyboard distribution, you can configure the shortcuts for every action, although, the write mode, if you have a significantly different distribution,  might not be so useful for you.

To modify the shortcut of an action:

- open the .ini (configuration) file of the mode where the action is, you’ll see something like the image:
- what you write on the left is going to be the combination of keys that execute the action on the right. To combine different keys in the same shortcut, just write them separated by a “+” symbol, with no spaces in the middle as you see on the image.
- IMPORTANT:
  - don’t use two control keys like {shift, control, alt} in the same shortcut,
  - if any shortcut uses a control key, make sure it’s followed only by ONE key, else ways, it will recognize it as two shortcuts
- at the end of the file, there’s a list of how every key should be called when creating a shortcut.



```
[mouse_instructions]
caps_lock = contextToSwapMode
f = leftClick
i = middleClick
o = rightClick
j = scrollDown
k = scrollUp
h = scrollLeft
l = scrollRight
shift+k = cursorUp
shift+j = cursorDown
shift+h = cursorLeft
shift+l = cursorRight
```

## <u>KEYS NAMING CONVENTION FOR SHORTCUTS:</u>

Note: the naming for the altGr hasn’t been added because windows represents it as ctr+altgr, and I haven’t worked on a way to detect that specific case, to convert it into just alt_gr.

You can use ctrl+alt+(key here) and then if you press either that or AltGr+(the same key), it will activate the mapped function

Also the Fn key doesn’t have a naming because it is not detected by the software.

- for all the letters, their own character is their name. E.g. a = function

- for numbers, same as letters. E.g. 1 = function
- for each function key, their name is the key’s name but in low caps. E.g. f1 = function
- the naming of the tab key is: tab
- the naming of the backspace key is: backspace
- the naming of the block mayusc key is: block_mayusc
- the naming of the enter key is: enter
- the naming of the escape key is: esc
- the naming of the shift key is: shift
- the naming of the control key is: ctrl
- the naming of the win key is: cmd
- the naming of the alt key is: alt
- the naming of the contextual menu is: menu

Continues...

Special characters:
Certain special symbols are their own naming:

- |
- ‘
- ¿
- ´
- +
- {
- }
- ,
- .
- -
- <

The rest of the symbols like ~, can’t be used as shortcuts directly, but you can use the combination of keys that result in that symbol e.g. ctrl+alt++ = ~.