AS322578

Gaming Peripherals for Designers —Improving Your Interface

Michael Freiert

Learning Objectives

- Discover how macro commands can be used to speed up workflow, and identify hardware to apply them
- Learn about gaming peripherals, and learn how to select the best one(s) for your practice
- Learn how to find a mechanical keyboard appropriate for your needs in a CAD workflow
- Discover solutions for using these tools in a corporate environment

Description

In a well-set-up workflow, it gets harder to find incremental improvements we can use to shave seconds off each task. What if we could shave seconds off many of tasks with one tool? Computer gamers and mechanical-keyboard enthusiasts have a few tools that the CAD community has not yet widely embraced. Gaming peripherals are designed to send multiple keystrokes or commands to a computer in one key press. This class will look at how input devices such as game pads, gaming mice, and customized keyboards can improve a design professional's workflow. We will look at several types of individually customizable input tools, and discuss how we can implement tools that were not designed for a corporate environment in that environment. We will examine where these tools have shortcomings, and how to dial in their utility versus the potential for them to be a distraction. The speaker has been using the tools we will discuss for more than a decade in Revit software and AutoCAD software—find out why.

Speaker

Michael Freiert has many hobbies, has had a few professions, and is a strong believer in cross discipline problem solving. He has worked in ACAD since R12, Revit since 6.1, and has learned a bunch of things from working with Legos, wood, scenic design, special effects, security, live sound, machining, sewing, cooking, explosives and gardening - all influencing his design philosophy, choice of tools, and preference for using the most apt, if sometimes unconventional tools for a particular task. He has been working primarily in BIM Management roles for the last 15 years with a healthy stint as a BIM mercenary.

Introduction -

Fair warning, I am going to gloss over some of the more nuanced technical elements to look at the big picture and talk about things in ways targeted to introduce concepts. This talk is addressed to folks who may want to make use of these tools. This is not targeted towards the serious keyboard geek. It is aimed at introducing useful concepts in a practical way. As such, I may opt for some slightly 'non-standard' terms.

Simultaneously a strength and weakness due to a hard learning curve, traditional Computer Aided Design software (such as AutoCAD) utilises typed commands and shortened aliases for those commands. In modern Graphical User Interface "contextual" or "ribbon" command based design software those commands may be hidden behind multiple clicks, but may be accessible with keyboard shortcuts. Instead of mousing and clicking through three or four contextual menus, we can push one or two buttons and trigger that command without mousing in between the control area up to the ribbon and back. Even with two button keyboard shortcuts for commands, that can often mean one hand bouncing around a keyboard or a hand coming off the mouse to type a letter on the right side of the keyboard.

We want to take the simplicity of the little finger and index finger hitting <Ctrl><c> and bring that level of ease of use to as many as possible of our most used commands, across each design software we use.

What is a Macro and why do you care?

In short, by pressing one button, or a combination of keys as a shortcut, a user can trigger a specific action or complex series of actions on a computer. Without trying to find an extensive granular and legalistic definition, for our purposes this can be something as simple as pressing <Ctrl> <C> instead of mousing to the "Edit" drop down and then in the drop down clicking on "Copy". It can also be something as complex as pressing one or two keys to trigger a custom LISP or API routine that will execute multiple complex actions on a whole series of files in nested folders.

In most design software we can use macros to trigger commands that require mousing away from the work area, or are buried in a drop downs or contextual menus. This saves mousing time, and can make keyboard shortcuts even faster.

How do I make this happen?

There are a few different ways we will look at to implement macros, each have different complications. Corporate technology environments all have very different security protocols. Some offices let each individual work as an administrator on their physical work station, other offices do not allow any third party software to be installed or unapproved pheripherials used on an individual machine. If you can bring your own device you have options. If you can install (or get approval to install) third party software you have even more options. If you are locked down by your IT department, you are going to have to sell them on adding approved software or hardware.

Macros can be managed either with hardware or software solutions. Hardware solutions are programmable devices that send the full set of instructions with one key press, software solutions listen for a key press and then translate that at the computer to the desired input.

Software

Software options are either stand alone software that will take any input and generate the desired output, or device manufacturer specific software that includes the ability to add macros.

Stand alone macro software can include mouse actions. I have found these most useful where I need to execute a particular repeated operation in a specific software where the individual commands do not include the ability to program keyboard inputs for those commands.

Typically, you use the software to record a complex task once, and then trigger it for each manually selected instance of that task to be accomplished, or tell the computer to run that routine a specific number of times. I have not found these to be as useful as other options, but some of them can be used to approximate a programmable keyboard on a non-programmable one. e.g. a macro could be triggered by <ctrl><alt><a> on any keyboard that “pastes aligned to level” in Revit. They are particularly useful for Mac users who want to utilize a non-standard keyboard that is not supported by MacOS.

Hardware - Human Interface Devices (HID)

HID are the devices that humans use to talk to computers. They are a collection of interconnected switches, and an on board microprocessor which sends a particular signal to the computer when individual switches are pressed. The computer then does various things with that signal. HID include keyboards, mice, gaming pads, joysticks, VR/AR controllers and the likes. Broadly speaking, they are devices programmed to send a particular internationally standardized HID input code to a computer when we do something to them, such as press a button or spin a wheel. e.g. some USB “game controllers” send common HID “keyboard” signals such as the letter “a” to the computer where others send more peculiar HID signals such as the submarine control “dive plane up”. (yes, “dive plane up” is a standardized message, just like “Home” or “a”) The “media” buttons on many keyboards tell the keyboard to send HID controls for things such as volume up/down and “play”

Broadly speaking HIDs usually result in the computer interpreting several different “signals” from the same keypress with some coordinating keypresses. When you press <g> the signal “g” is sent to the computer. When you press <shift><g>, both “shift” and “g” are sent to the computer and the computer knows that said combination should be read as “G”.

Keyboard “Modifiers” are signals that are sent coincidentally to input signals that notify the computer to treat them differently. <Ctrl><c> is read as “copy” whereas <Shift><c> is read “C”. <Alt> is the other common Modifier. Behavior changed by Modifiers are handled by the computer.

What we normally think of as the basic keys with the “shift variations” is a standard “keymap” with Modifiers. Multiple keymaps can exist on one device (but often do not). We can switch between different keymaps by changing “Layers.” A Layer is a different “keymap” or meaning for each individual button to be pressed. The most common use of a Layer is the <fn> ‘function’ key that most laptops and many USB keyboards include. “<fn> should not be confused with the “F#” Function keys.) Layers are sometimes difficult to differentiate from Modifiers but true Layers change the HID code being sent to the computer with each keypress.

On a laptop, when you press <fn><f5> you may be sending “multi window” instead of “f5” or you may be sending “???” and “f5” that your computer is interpreting as “multi window”. The difference is if your ‘keyboard’ is sending a common HID signal with a custom modifier (on your laptop this is probably not something you can easily change) or if the computer has a customized Human Interface Device that looks like a “standard” keyboard and changes the signal it sends before it leaves the “keyboard” by switching to a new keymap.

Some HIDs use Layers to internally alter the signals they send, and some are merely oddball shaped keyboards that use software on the host machine to redirect their sent keyboard signals into macros or variously programmed keystrokes or HID commands.

HIDs

Some terms and notes on HIDs relative to using them for more interesting control options -- There are lots of other types of HIDs but we will look at the following:

Mice

Beyond the normal left and right click, there are mice that include more buttons and controls. Common additional inputs include Scroll wheels, Paddles, Thumb Buttons and mousing resolution switching buttons (DPI).

Keyboards

In addition to basic keyboard functions, some keyboards have additional features such as dedicated macro keys, additional dedicated keys (such as media) or fewer overall keys:

- Mechanical keyboards use a mechanical switch rather than the standard rubber dome for improved accuracy and to change the feel of the keypress.
- Programmable keyboards include the ability to change what individual keys send to the computer
- “60%” keyboards include “text typing” keys only - Ctrl to Ctrl and Ctrl to Number Row,
- “80%” or TenKeyLess (TKL) keyboards do not have a number pad, but do include F# row, arows and the “Insert/Delete” cluster.
- “Full Size” keyboards include a number pad and F row keys, and are typically 104 keys

- Variant keyboard layouts include “ortholinear” (instead of staggered layout) and a variety of “ergo” keyboards.

Gamepads

These are a cluster of 15 to 30 keys that behave as a portion of a keyboard. They may have scroll wheels, thumb buttons and a variety of other accessories. Most of them rely on computer software to remap their keypresses to produce various effects and are similar to macropads, but typically more “designed”. They can be ortholinear, a partial staggered layout or completely unique.

Macropads

These are 1-30 button “keyboards” that send individual key presses or macros. They are typically hardware programmable, and usually a simple grid array of keys. Some are software configured, with a custom driver intercepting their standard HID commands for the computer.

Gaming Devices

These are a broad classification of HIDs that usually include some sort of customizability to send specific macros or commands for use in computer games. They also usually include a lot of marketing hype and LED backlighting. They are, however the market segment where we can find most left hand controllers such as Gamepads and customizable or extra button Mice. As customizable or “performance” devices these often use mechanical switches instead of rubber dome switches.

Keyboards

Just as many cars do not come with a tachometer unless you get the “rally” package, mechanical keyboards are a more “performance” option for typing. People who type a lot professionally have developed a variety of variations on the typical keyboard. This includes extra keys, fewer keys, gently or dramatically varied keyboard layouts, and variations in the behavior of the switches themselves.

The performance option we are most interested in are macros and layers which are found in programmable keyboards, which are typically mechanical keyboards. There are a LOT of other options. One particularly interesting option is left hand only partial keyboards to pair with right hand mouse input, but again these are typically mechanical keyboards.

Keyboard switch types:

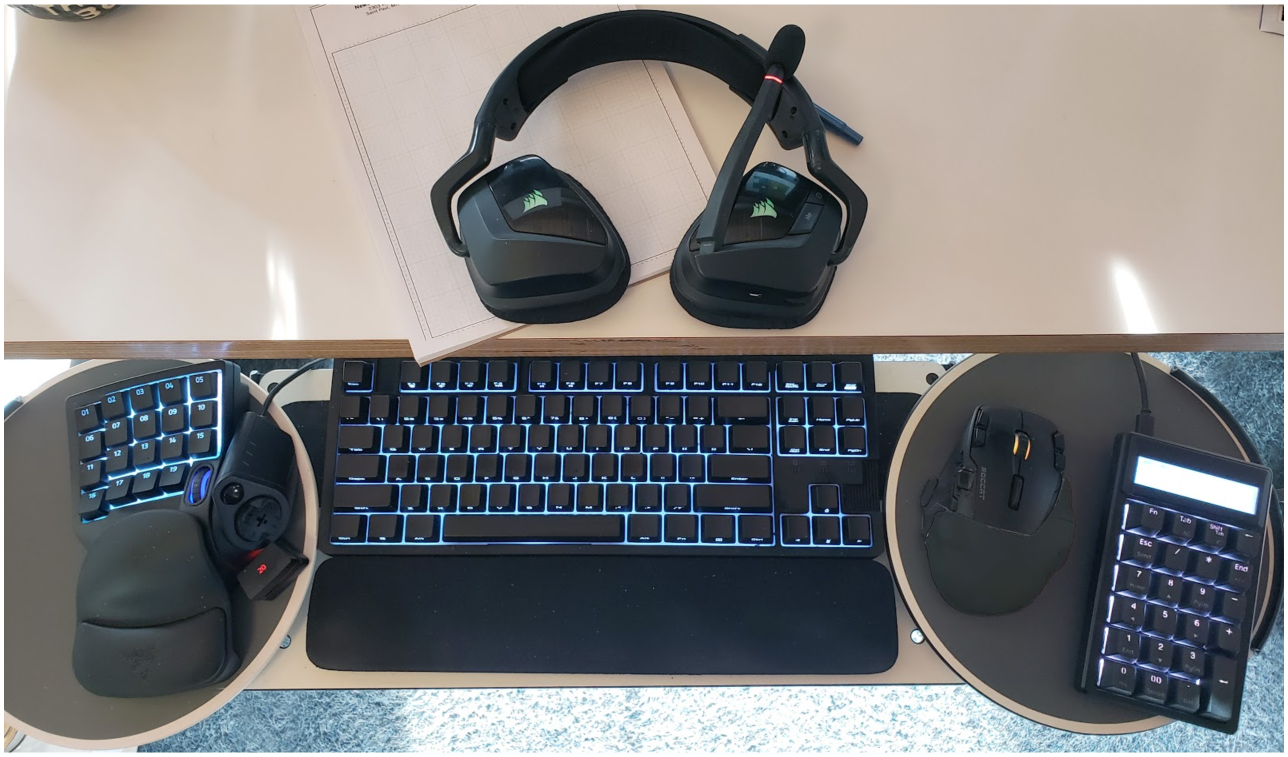
There are a variety of types of switches used in keyboards, which affect the feel of each keypress, the durability, and the moisture and dust resistance. Briefly, the most common are:

- Rubber dome: most modern keyboards - a rubber dome with an electrical contact on it is squished into another contact. “Feels mushy” If you’ve taken apart a cheap calculator, that’s probably this switch.

- Scissor switch: a rubber dome switch with little ‘x’ shaped stabilizers - more stable and shallow than other types, commonly used on laptops
- Buckling spring: most famously the iconic “IBM model M” keyboard. A spring is compressed and gets very soft when it bends sideways. Widely considered the “best” keyboard by many enthusiasts. Rare in modern keyboards, but reproductions are available.
- Cherry MX switch: the common “mechanical” switch. A tiny coil spring holds up a stem which presses a contact closed when it is in the down position. There are several manufacturers: Cherry, Gateron, Otemu, Kalih, Greetech. There are also custom runs by those manufacturers for other brands: Zealio, Hako, Halo. There are three general types, and many many varieties. The “color” of the switch refers to the color fo the plastic on the stem that the keycap mounts to, and is used to differentiate when a variety of switches are used on one device.
    - Linear switches - “Red” and “Black” These switches are smooth throughout the full press
    - Tactile switches - “Brown” These switches have a gentle detent where they are actuated which gives feedback before bottoming out
    - Clicky switches - “Blue” These switches are similar to a tactile switch but include an audible “click” sound at their actuation point for additional feedback. - Very common in gaming devices. Less ideal for open office environments.
- Romer G: more or less a donut shaped Cherry type mechanism
- Infra optical interrupt: include an IR emitter, IR sensor and a shutter which blocks the IR signal to trigger a keypress, there is no metal contact so they have reduced wear, and can include multiple inputs per key as a sort of slider.
- Topre: are similar to a rubber dome switch, but sense electrostatic capacitance rather than physical contact pad, and include a metal spring for a more reliable feel.
- Hall effect: A magnet next to a micro chip activates the circuit. Similar to an optical interrupt these have no physical contact and are *very* reliable -- in the billions of cycles compared to “only” 50 million cycles in a typical Cherry style switch.

What I use

It has taken me awhile to get here, but this is my current desk layout. Left to right, I have an ergonomic left hand Gamepad that uses control software. I have a "tenkeyless" keyboard[1] to type on that includes internally programmable layers. I have a gaming mouse that uses control software, and I have an independent numberpad that also functions as a desktop calculator.   I also have a wireless headset that I use for web hosted meetings.



For GUI type design work, I mostly use the gamepad and mouse, centered around the keyboard. When typing I pull both hands into the keyboard. When I need ten-key input, depending on which hand I want to use, a layer on the gamepad acts as a number pad, or I use the number pad to the right of my mouse.

On the Gamepad I have configured common commands, non-letter characters (±, Ø) , common short text bursts (VIF, AFF) and a number pad. On the keyboard, in addition to the typical QWERTY layout, I also have a layer for non-letter characters that get included in typed information, and a few more complex typed strings or commands. On my mouse I have very commonly used keystrokes mapped to extra mouse buttons.

[1] I normally use a KBD75v2, but was modifying it when this picture was taken. Pictured is my backup tenkeyless (same keycaps).

This setup has evolved over the years, but the process was just a few minutes here and there as I found commands I wanted to include or relocate. Initially, I added a Nostromo n52 Gamepad to my full sized keyboard and programmed it with my most common commands. I determined what those common commands were with a LISP routine in ACAD, and have adjusted as software needs changed, and as I've found other input options.

To start from scratch, look at the common commands you use or tasks you repeat and assign them to a keyboard shortcut (triggered by a macro) or a Macro depending on how you are going to trigger them. If possible, assign the same keyboard shortcut to similar behavior in all of your software. e.g. I have Revit's Trim command keyboard shortcut mapped the same as my AutoCAD Fillet command (radius 0) shortcut as I usually use their behavior in similar ways. Some Gaming Devices that use a software side macro will detect which program you have active, and can automatically apply a different keymap to your device.

To learn to use a new input device (or new keymap) I havefound it easiest to first assign the most used commands to the easiest to access points, and then keep adding commands that are used often enough to automate as you realize you do not have a proverbial "easy" button for them.

Our firm allows local Admin privileges on workstations, which means our users can install "non standard" software such as gaming HID manager software. I have device manager software running for my mouse, game pad and headset. This is *not* viable under some office IT protocols.

If I was not able to install these device managers, but could bring my own devices, I would be using a left handed programmable keyboard (such as ErgoDox) that includes all of those GamePad macros internally. I would also look at modifying my programmed keyboard to include behavior similar to my Gamepad setup. For AutoCAD I used a Logitech m520 mouse with two thumb buttons programmed to <m> and <Enter>. That mouse's standard drivers let you specify one keystroke for those thumb buttons, and it is a boon for single key shortcuts. Now, my thumb buttons are <Enter> and <Space> to repeat commands in Revit and flip objects, although I could set them to anything.

Gaming Device Manager Software

These software packages are installed on a local machine and listen for one of their associated HIDs to send an HID signal to the computer. They are designed so that competitive gamers do not get frustrated with system performance or unreliability, although cheaper versions of them can be less reliable. These are all varying degrees of tedious to set up, but run quite smoothly once set up, and if you can find all of the behavior you want in one brand, can be very coordinated with multiple devices. When the software hears a particular "trigger", rather than 'typing "a"' the software signals the OS to do whatever the user programmed the "a" button to do. That could be <left arrow> or <t><r> <pause> or to type a lengthy text string replete with <enter> and <tab> commands.

Programmable Keyboards and Macropads

The advantage to a programmable keyboard over a ‘gaming’ keyboard is that this key-mapping or macro ability is internal to the keyboard. While all keyboards have a microcontroller on board, these devices specifically allow the user to customize the signals sent for each keypress. This is accomplished by reprogramming the firmware with a file generated by a website or programming software. They usually include multiple Layers, meaning you can include “keys” that are missing from your physical keyboard. e.g. A 60% keyboard that is “missing” a number pad and Function rows, can double up the number row as a Function row, and include a number pad or odd characters as well as macros to keyboard shortcuts.

Most of my experience with programmable keyboards is with kit keyboards that are limited production runs or assembled by the user. Several of these can be be purchased pre-assembled, but not always in the exact configuration that you may want. The advantage to this is endless possibilities of combinations of custom keycaps, switches, lighting, springs, cases and other options. Off the shelf configurations can have keycaps swapped out, and some can be easily user modified. Just as there are different switches, there are different keycap shapes which can impact the feel of the keyboard.

Setting up Macros

This varies a bit by manufacturer or programming software, but in all cases you first define the macro, and then assign it to a key. This can be by “recording” typed keystrokes, or by “defining” the keystrokes. With some software it is possible to Record the command string and then modify the definition.

Exact terms vary, but in defining a macro, the options are to “Type” a key, “Press” (and then) “Release”, “Wait”, and “Set Interval”. Type is just a single keypress and release. Press is depress and hold a key until it is Released. Wait will pause for a chosen value of milliseconds, and “set interval” will allow you to define the default interval between actions. This can be helpful if you want to enter things a bit slower to give software time to respond to prior inputs or you’re seeing dropped keystrokes. Using design software, I’ve found no issues with single commands with a 10 millisecond delay between keystrokes, or noticed the few hundredths of a second delay in execution. (In competitive gaming this is apparently much more important.)

Examples

n.b. There are a LOT of ways to accomplish these inputs, these are illustrative.

<u>Revit Trim command</u>

Set Interval 10 milliseconds
- Type t
- Type r

<u>Paste aligned as one keystroke</u>

- Press <shift>
- Press <ctrl>
- Wait 5 miliseconds
- Type <z>
- Wait 5 miliseconds
- Release <shift>
- Release <ctrl>

On a gaming device (keyboard, mouse or gamepad) this is usually accomplished by opening the macro editor in the device software, and then saving the profile. The device manager software often has the ability to detect which software is active, and can switch profiles automatically. This can mean you do not need to match keyboard shortcuts between software, but also means you may need to reprogram some or all commands for software with divergent shortcuts. It can also mean that you can have entirely different behavior from the device for different software.

More common on gaming keyboards, there is occasionally the option to add in a macro that's manually typed. Usually a "record" key is pressed at the same time as the "target" key and then the macro is typed in. On some devices this is replayed in real time, on others it is played back at a fixed delay between key presses.

Programmable keyboards use a few different firmwares to run, which are very similar. Using the correct compiler on your computer, you program your keymap and macros and then use the software to flash this to the keyboard.   Compilers vary from web pages that are very easy to use (but may not have all features available), to local software that can have a nice GUI or very arcane plain text editing.   The compiler generates a specific file which is saved and then flashed to the keyboard with secondary software. Some programmable keyboards include an all in one software to manage this process.

Resources:

Some of these I've found useful, some others have been recommended. Features, EULA and licensing may change, please take their mention here with a rather large block of salt.

Information

A good switches primer: https://www.pcgamer.com/best-mechanical-switches-for-gaming/
Partial switch type comparison list:
https://mechanicalkeyboards.com/switches/index.php?mkref=b686xub
List of gaming peripheral manufacturers:
https://en.wikipedia.org/wiki/List_of_computer_peripheral_companies
QMK firmware (the firmware for many programmable key boards): https://docs.qmk.fm/#/
Macro Software:
http://www.pitrinec.com/products/macro-toolworks
https://www.autohotkey.com/
Mechanical Keyboard Forums:
https://geekhack.org/index.php
https://deskthority.net/ (also great Wiki)
https://www.reddit.com/r/MechanicalKeyboards/
ErgoDox: https://www.ergodox.io/

Devices

My favorite switches: Gateron Silent Brown, Kailh Pro Purple
Switches I also like: Cherry MX Brown, Gateron Brown, Kailh Speed Copper.

Gamepads:
Belkin/Razer Nostromo Speedpad n52 :
https://www2.razer.com/au-en/gaming-keyboards-keypads/razer-nostromo
Razer Tartarus V2: https://www.razer.com/gaming-keyboards-keypads/razer-tartarus-v2
Razer Tartarus Pro: https://www.razer.com/gaming-keyboards-keypads/razer-tartarus-pro
Ergo keyboards:
Ergodone: https://kbdfans.cn/collections/ergodone
Ergodox Infinity:https://kono.store/products/infinity-ergodox-keyboard

Mice:
Roccat Leadr: https://en.roccat.org/Mice/Leadr
Logitech m510: https://www.logitech.com/en-us/product/wireless-mouse-m510 ‘
X-Mouse Button Control: (macro/HID control software for any mouse)
https://www.highrez.co.uk/downloads/XMouseButtonControl.htm

Numberpad:
Ducky Pocket: http://www.duckychannel.com.tw/page-en/Ducky-Pocket/

Keyboards - programmable:
Melody 96/YDMK 96:
https://github.com/qmk/qmk_firmware/tree/master/keyboards/melody96
KBD75v2: https://kbdfans.cn/collections/diy-kit/products/kbd75v2-custom-keyboard-diy-kit

Keyboard non-programmable:
Insist G55 Pro: http://www.insist-gaming.com/product/show.php?itemid=6&l=en#slide6
GMMK compact* : https://www.pcgamingrace.com/products/the-glorious-gmmk-compact
*I have not tried the GMMK, but it’s a well reviewed and a relatively affordable keyboard that can hotswap different switches to see if you like the feel of particular switches before committing to soldering 60-110 of them onto a PCB.