THE PERFECT KEYBOARD FOR GNU EMACS AND XML: THE SYMBOLICS/APPLE MACIVORY

John Bear
November, 2002

This paper has been in preparation for too long. I finally decided (May 31, 2005) to post it as is, figuring something is better than nothing.

This article is for two groups of people: people who use Emacs and people who don't. My purpose is simple. I want somebody, Apple or Symbolics, or some third party, to bring back the Symbolics MacIvory keyboard. The way I hope to accomplish this is to let people know that it exists, and that it's perfect for Emacs and XML, and merely damn good for everything else. If enough of us make it clear there's a market, I figure someone will start selling them again. We Emacs users win, and so does Symbolics or Apple, or whoever starts selling them. Much of this article is about how the keyboard was designed to work with Emacs, but the rest is about how it out-performs other keyboards even for normal typing tasks.

The MacIvory was an Apple Macintosh computer that had a Symbolics Lisp machine inside, and was sold around 1987. It came with a Symbolics keyboard that had the same layout as earlier Symbolics keyboards, but with keys that were easier to press. It was designed (as were all Symbolics keyboards) to be used with the Zmacs editor, a version of Emacs.

Although the keyboard is not a split keyboard, it was designed with kinesiology in mind and deserves to be considered an ergonomic keyboard. There are so many factors that go into making a good keyboard and none of the so-called ergonomic keyboards on the market that I've tried measure up to the MacIvory.

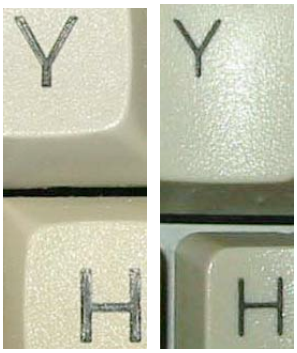
Here are a few of the things the MacIvory keyboard does right that I haven't found to be as good on any other keyboard. They are divided into general points and Emacs-specific points, with the general points first.

General Typing

One of the most commonly hit keys is delete/rubout. Maybe it shouldn't be, but it is. So the rubout key deserves to be close to where the hand is when it's on home row (asdf). It is. There is a well known law called Fitts's Law which says that targets are easier to hit when they are nearby and large. From a design point of view the keys that are used most often should be made easy to hit. The simple way to do this is to make them large and close to the home row. The MacIvory does better at this than any other keyboard that I know of. The rubout key is large (double-wide), and adjacent to the letter A, immediately to its left.

This photo of the MacIvory keyboard shows a Rub out or Delete key that is large and close to the left hand's home position.

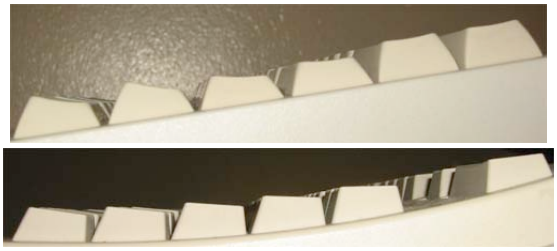
Although there are as many keys per inch, and as many rows per inch on the MacIvory, the MacIvory's keys taper inward more than keys on other keyboards, making them feel farther apart. The tops of the MacIvory keys are 0.50 inches wide by 0.50 inches high. On an Ortek Adesso keyboard (an add-on for Macintosh computers) the key tops are 0.50 inches wide, but 0.57 inches high, and consequently, the rows (on the Adesso) are 0.18 inches apart as opposed to 0.25 inches for the MacIvory.



The left photo is of the MacIvory keyboard. On the right is the Dell.

These photos show that the distance between key tops is larger for the MacIvory. This may lead to fewer typing mistakes on the MacIvory. My conjecture: The larger distance between rows makes it easier to avoid accidentally pressing neighboring keys, leading to fewer typing mistakes, and less repetitive strain.

When a key on the MacIvory keyboard is pressed all the way down, some part of its top is still higher than the top of the keys in the next row down. This is accomplished by canting the key tops a little more towards the home row than on other keyboards, and by having more vertical separation between the rows to begin with. The canting effects a piece-wise linear approximation of concavity.



MacIvory is above, Sun is below.

The photo of the edge of the MacIvory keyboard shows that the tops of the keys are sloped, and that the slopes are different for the different rows. It also shows that the tops of the keys are concave front to back as well as side to side. In contrast, the tops of the middle three rows of keys of the Sun keyboard are flat and level; the bottom two rows of the Sun are sloped only slightly and are straight across, not curved. The photos also show that the MacIvory achieves the effect of having rows that are farther apart. The Sun image has been reversed horizontally for easier comparison with the MacIvory.

Table of row heights for two keyboards

Key Row	Adesso	Symbolics/MacIvory
1234	0.17	0.23
QWER	0.13	0.18
ASDF	0.12	0.15
ZXCV	0.09	0.14/0.18

Height of a row of keys above the next lower row. Measurements were taken from both the high and low sides of a key and averaged. The final entry has two numbers because the space bar on the Symbolics is lower than the other keys in its row. The second number is the height above the space bar. Measurements are in inches, made with a dial caliper calibrated in hundredths. For the Adesso keyboard, measurements were made with the keyboard's feet deployed for maximal similarity with the MacIvory.

This uses the converse of Fitts's Law. If it's easier to hit a key when it's nearby, it's also easier to hit it accidentally. Conversely, if a key is a little farther away, it's easier to avoid hitting it accidentally. This aspect of the MacIvory keyboard lets the fingers that are not pressing keys relax a little more. They don't have to work as hard to keep from accidentally hitting other keys. This leads to fewer typing errors and less repetitive strain. Again, this is a conjecture.

We used a small level on several different keyboards, using it to push down keys in the different rows, and seeing whether the key in the next row down was pushed down. On the MacIvory, each row, even when depressed, was higher than the next row down, except for the home row.

The tops of the home row keys (asdf) went down slighly farther than the tops of the next row down. On two different Dell keyboards (Quitekey and another whose name I don't know), pressing down on a key brought its top lower than the next row down for each of the rows. On the Sun (type 5C class B), Silicon Graphics, and Dell Quietkey keyboards, the tops of the keys also went farther down than the next row down. In fact, pressing a key in home row with the level on these keyboards depressed not only the key in the next row, but also the space bar, a row away.

Close-up of F,J for Macivory

The careful observer will notice that the F and J keys on the MacIvory have a deeper depression in the top than the others. Most keyboards have some way of letting the fingers distinguish the F and J keys without looking. Often it's a raised bump. On the MacIvory it's a deeper depression. The key action on the MacIvory keyboard is better than on any other keyboard I've tried. When you press a key on the MacIvory keyboard, it goes down smoothly. This makes it easier to avoid typing mistakes. It matters. Ask anyone who has used a MacIvory keyboard, or the Symbolics keyboard on which it is based, and they will tell you the keys feel good, good enough to matter. So far we've compared the MacIvory with a few other keyboards with respect to generic typing tasks. Next we'll consider some Emacs-specific aspects.

Emacs-specific Ergonomics Ergonomics is about making it easy, safe, and efficient to accomplish your task. In ANY computer-related task there are two types of subtasks: data entry, and controlling the computer. Any discussion of keyboards and ergonomics should address both. By "data entry" we mean typing a word, like ergonomics, and having it show up in a text editor. By "controlling the computer" we mean:

- scrolling windows and frames resizing windows,
- frames, pictures, icons, text, etc.
- selecting objects
- sending email
- debugging - compile, load, test, edit, repeat
- browsing and navigating the file system hierarchy
- connecting to another computer
- loading files
- invoking a process or application

- editing
- etc.

One extremely important subclass of commands for controlling the computer is the class of actions that constitute editing a document.

- copying
- cutting
- pasting
- searching
- saving
- reformatting
- changing fonts
- changing indentation
- inserting links or pictures
- scrolling
- moving the cursor
- etc.

These tasks are accomplished by executing commands. The software developer who wrote the software is the one who decided which keyboard sequences would execute which commands. Sometimes these decisions are in accordance with interface guidelines set out by one of the powers that be, such as Apple or Microsoft.

So far as I know, the MacIvory and Symbolics keyboards (and their kin) are the only ones that were designed to make controlling the computer with Emacs easy. The ergonomics keyboards I know about aim to make typing easy, but don't try to make it easy to hit esoteric control-meta key combinations.

The MacIvory has two sets of control and meta keys. These keys are part of almost every Emacs command. Their use is pervasive. As befits Fitts's Law, the MacIvory's control keys, at least, are large and close to where they should be. They flank the space bar. Moreover, the two most common Emacs command prefixes are Control-x and Meta-x, so it matters that there be at least one set of the control and meta keys near the X key. On the MacIvory there is. The meta keys should be larger, but at least they are there where they should be, next to control, on both sides of the keyboard.

The CONTROL and META keys are on each side of the MacIvory's space bar.

Document structure and navigation

For a specific instance of the use of control and meta keys let's consider navigation, especially within a file. Emacs has commands for going forward or backward by one s-expression, or going to the beginning or end of a function definition. These commands were designed for navigating and editing

lisp code, but they are handy in any document.

The commands I'm thinking of are:

- control-meta-f (move cursor forward 1 s-expression)
- control-meta-b (move cursor backward 1 s-expression)
- control-meta-a (move cursor to beginning of function definition)
- control-meta-e (move cursor to end of function definition)

I've included two short snippets of more or less equivalent code, lisp and javascript, so that if people are looking at this page in Emacs and aren't used to these commands, they can try them out. Just about any coherent chunk is an s-expression. A parenthesized or bracketed list is an s-expression. A word is an s-expression. A double-quoted string is one too. If you have a string within a bracketed expression, if the cursor is on the initial bracket and you type control-meta-f (or escape control-f) the cursor will skip to the matching bracket. If you put the cursor on the initial double quote inside the brackets, and type the command, the cursor will skip to the matching double quote. If you can't get the control and meta keys to work together, using escape first is a workaround, that will let you what the cursor does, but it won't have as good a feel as using both keys at the same time, as was intended.

You can try out the commands here if you're looking at this in Emacs.

```
(defun foo (y)
  (cond ((or (eql y 1)(eql y 4)) (print "y is 1 or 4"))
        ((equal y "Tuesday") (print "Y is a day"))
        (t (print "y isn't 1 or 4 or Tuesday"))))
```

```
function foo(y)
   switch(y)
   {case 1,4:
    alert("y is 1 or 4");
    break;
    case "Tuesday":
    alert("Y is a day");
    break;
    default:
    alert("y isn't 1 or 4 or Tuesday");}
```

Although very few people code in lisp these days, there are still reasons for wanting to be able to use these commands. I'll give two. First, every document, whether it's a computer program, a web page, a project plan, or a report, has structured elements. It is very handy to jump to the next segment whether it's a block of java code, an HTML anchor, a subgoal of a project, or just the next chapter or subsection or paragraph of a report. Emacs control-meta navigation commands were designed for this.

The second reason is XML. XML is one of the waves of the future. Microsoft Word has ways of dumping files with XML markups in them. Apple OS X stores system properties in XML. Web services are going in the direction of XML. Web pages are gravitating from HTML to XHTML, and are heading toward pure XML. It will be important for a long time to come.
XML is essentially a bracketing language. (Really of course it's a meta-language.) The expressions in an XML document are conceptually identical to the s-expressions of lisp, which Emacs and the MacIvory keyboard were designed for. The Emacs control-meta commands for traversing s-expressions

are exactly what XML developers need, but only if there is a keyboard available that makes it easy to use these commands. Again, the MacIvory keyboard does make it easy. If the MacIvory keyboard were available today (November, 2002), people would use it and find it considerably better than what they have. But some would also write XML versions of the commands so you could go forward or back by one XML expression, or go to the beginning or end of an xml expression of a certain type.

[Postscript note: This has already been done, and may have been done when I was writing this initially in 2002. JB]

Conclusion

Some of what I've written is about how the MacIvory keyboard was designed for Emacs. But many of the points I've made are about how the keyboard is just plain good, as a general purpose keyboard. People who have never heard of Emacs, if they could use this keyboard, would appreciate it.

The MacIvory keyboard embodies a principled solution in hardware to the problem of how to enable a person to use Emacs editing and control software effectively. There is a market for this keyboard now, and the market will grow as XML, on the one hand, and Gnu/Linux on the other, gain popularity. The keyboard will not be cheap, but people buy ergonomic keyboards for hundreds of dollars these days that aren't nearly as good as the MacIvory. There are, of course, much cheaper keyboards on the market today that aren't as good. There is room in the marketplace for both.

Acknowledgements

I want to thank the people who helped with this document. Susanne Riehemann took all the photos and made many good observations. Mabry Tyson, who loaned me the MacIvory keyboard, also reminded me of some of the history and made helpful comments. Thanks to both of you.