Text Editors, Vim, and More

Ethan Oswald Massey[1]

[1]Hochschule Bonn-Rhein-Sieg

September 7, 2019

Outline

About Me

- Fifth semester HBRS student currently working on my thesis
- Graduated from University of Wisconsin-Madison with a double major in Computer Engineering & Computer Sciences
- Worked for two years after graduating in the industry doing mostly C and C++ work on embedded and systems level projects
- Avid Vim user of just under 10 years

Text Editor Overview

- Text editing vs text viewing
- Plugins
- Terminal vs GUI
- GUI
  - Atom
  - Lime (open-source Sublime)
  - Gedit
  - Any IDE (VScode, PyCharm, Eclipse, etc)
- CLI
  - Vim
  - Emacs
  - Pico
  - Nano

Vim

What is it? Why use it?

- A CLI text editor
- Unique set of key bindings
- Pros
  - Efficiency
  - Power
  - Ubiquity
- Cons
  - Steep learning curve

Vim

The Basics

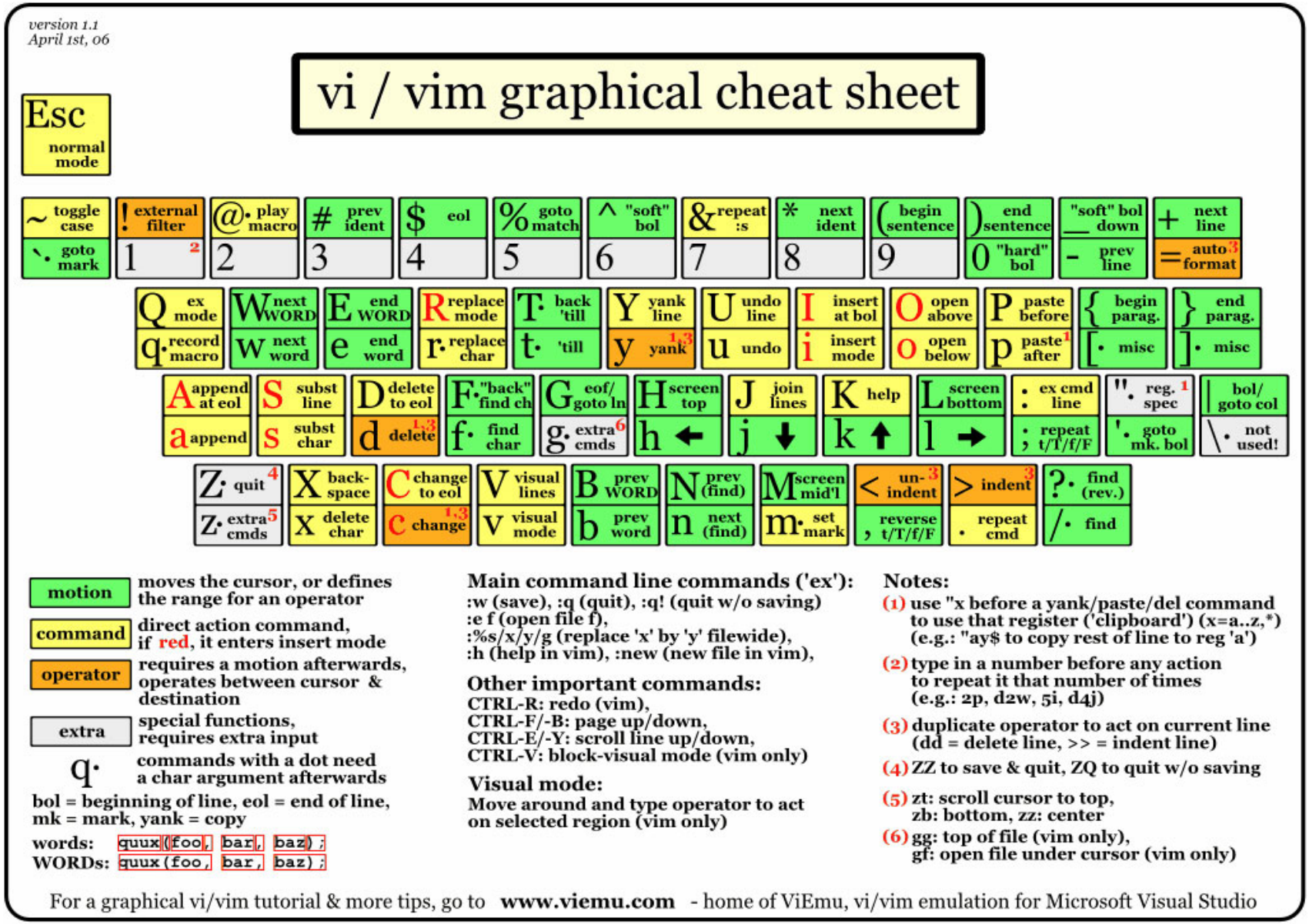
- Two main modes - Normal (commands) & Insert (Typing)
- Move with {h, j, k, l}
- Enter insert mode with “i”
- Exit insert mode with [esc]
- Undo “u”
- Redo ctrl + “r”
- Save “:w”
- Quit “:q”
- Save & Quit ”:wq”

Vim

Find & Replace

- Search with “/” (case sensitive)
- Use “*” or ”#” to search forward/backward current word
- “n” for the next instance
- “N” for the previous instance
- Find and replace using sed
- e.g. “:%s/replace_me/replaced!/g”

Vim


version 1.1
April 1st, 06
vi / vim graphical cheat sheet
Esc
normal mode
~ toggle case
` goto mark
! external filter
1
@ play macro
2
prev ident
3
$ eol
4
% goto match
5
^ "soft" bol
6
& repeat :s
7
* next ident
8
(begin sentence
9
) end sentence
0 "hard" bol
__ "soft" bol down
- prev line
+ next line
= auto format
Q ex mode
q record macro
W next WORD
w next word
E end WORD
e end word
R replace mode
r replace char
T back 'till
t 'till
Y yank line
y yank
U undo line
u undo
I insert at bol
i insert mode
O open above
o open below
P paste before
p paste after
{ begin parag.
[misc
} end parag.
] misc
A append at eol
a append
S subst line
s subst char
D delete to eol
d delete
F "back" find ch
f find char
G eof/ goto ln
g extra cmds
H screen top
h
J join lines
j
K help
k
L screen bottom
l
: ex cmd line
; repeat t/T/f/F
" reg. spec
' goto mk. bol
| bol/ goto col
\ not used!
Z quit
Z extra cmds
X back-space
x delete char
C change to eol
c change
V visual lines
v visual mode
B prev WORD
b prev word
N prev (find)
n next (find)
M screen mid'l
m set mark
< un-indent
, reverse t/T/f/F
> indent
. repeat cmd
? find (rev.)
/ find
motion
moves the cursor, or defines the range for an operator
command
direct action command, if red, it enters insert mode
operator
requires a motion afterwards, operates between cursor & destination
extra
special functions, requires extra input
q·
commands with a dot need a char argument afterwards
bol = beginning of line, eol = end of line,
mk = mark, yank = copy
words: quux(foo, bar, baz);
WORDs: quux(foo, bar, baz);
Main command line commands ('ex'):
:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:%s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :new (new file in vim),
Other important commands:
CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)
Visual mode:
Move around and type operator to act on selected region (vim only)
Notes:
(1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay$ to copy rest of line to reg 'a')
(2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)
(3) duplicate operator to act on current line (dd = delete line, >> = indent line)
(4) ZZ to save & quit, ZQ to quit w/o saving
(5) zt: scroll cursor to top, zb: bottom, zz: center
(6) gg: top of file (vim only), gf: open file under cursor (vim only)
For a graphical vi/vim tutorial & more tips, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

Vim

A Method to the Madness

- Vim is a language
- Verbs - e.g. “d” (delete), “c” (change), “y” (yank)
- Modifiers - e.g. NUM, “i” (inner), “t” (to), “f” (from)
- Nouns - e.g. “w” (word), “)” (sentence), “}” (paragraph)
- The goal is not to memorize random key combinations but to think “I would like to change all the parameters in this function call” which translates to “ci)“

Vim

Not convinced?

Demo

Vim

Anti-Patterns & Things to Avoid

- Never use the arrow keys (don't move in insert mode)
- Try to avoid using the mouse
- Avoid hitting the same key more than a few times (use numbers)
- If you are doing an action repeatedly, ask yourself if there's a better way

Vim

Tips and Goals

- “.” repeates the last command/action
- “ci” + w, (, ¡, ” is extremly useful
- “gg=G” will reformat your code
- Learn to use visual[Line & Block] mode
- The global buffer can be accessed with “”+p” “”*p”
- Alternatively you can paste with ctrl + shift + “v”
- **Have patience**
- **Learn a new key every few days**

Closing Thoughts

General Recommendations for a Development Environment

- Take time to customize and explore settings
- Play around, break things
- Ask others what they use/how they do things
- Avoid elitism but don't always do what's easiest

References & Resources

 Vim Cheat Sheets

 What is Vim and Why use Vim?

 Learning Vim in 2014: Vim as Language

 Learn vim For the Last Time: A Tutorial and Primer

 Vim anti-patterns

 Vim Adventures

 Online Vim Tutor

 Online Vim Tutor