The zen of vim editing

Pranesh Srinivasan

March 5th 2010

Why Vi?

- Modal Editing - Learn to love the 100+ keys on your keyboard
- Laptop Keyboards
- *Muscle Memory*

History

A Long Long time ago

- Bill Joy first wrote Vi (Visual) - as an alternative to ed.
- The editor of the beast—*VI*
- Brought about the concept of Modal Editing.
- Extensibility and Functionality were questioned.

Not so Long Ago

- Several Vi clones came about - nvi, elvis
- Vim - **V**i **IM**proved - Bram Moolenar - best by miles (charityware)
- Most computers have vi sym-linked to vim

Installing Vim

Vim runs on most platforms.

- http://www.vim.org gives binaries for most systems.
- If you run a debian based system, you could do `sudo aptitude install vim-full gvim`
- Personal preference - gvim

The principles behind the Vim editor

- **Not** mean to be an IDE.
- The GNU Philosophy - **do one thing well**
- Lets you do the basics right in a quick way.
- Otherwise stays out of your way.
- Keystrokes **Matter**
- Big Learning Curve, but once crossed, gives great power and flexibility.

The zen of the Vi Editor

```
"To me, vi is Zen.
 To use vi is to practice zen.
 Every command is a koan.
 Profound to the user,
 unintelligible to the uninitiated.
 You discover truth every time you use it."
```

The concept of the Vi Editor

- 2 Basic Modes (actually 6)
- Two distinct phases - writing, and editing.
- Reuse Keys

Movement

- Get used to `h, j, k, l`. It really helps.
- Moving screenfuls forward and backward.
- Moving your cursor to the highs and lows of the screen.

The magic key: **Escape**

The art of insertion

- The really simple one: `i, a`
- The basic ones: `I, A`
- Above and Below: `O, o`
- Replacing Character(s) : `r, R`

Undo and Redo

- Undo: `u`
- Redo: `Control + r`
- *UberCool* Feature: Eariler (`:earlier`)

Copying and Pasting

- Copy, Cut (Delete), Paste - `y`, `d`, `p`
- Copying, and deleting lines - `yy`, `dd`
- Paste before, and after - `P`, `p`

Moving across the file

- Start of File `1G`
- End of File `0G`
- Line number `:number`. Example `:234`

Visual Mode

- Normal Visual Mode (`v`)
- Line Visual Mode (`V`)
- Block Visual Mode (`Control + V`)

Searching and Replacing

- Searching is done with a `/`
- Find next - `n`, and find previous - `N`.
- Replace with `:%s/this/that/`

Buffers

What are buffers?

- opening multiple files
- buffer next, buffer previous -> `:bn`, `:bp`
- jumping between buffers -> `C-^`
- listing buffers `:buffers`

Saving and Exiting

- Exit - `:q`
- Save and Exit `:wq`
- Exit, and throw away changes - `:q!`

Combo Time

What do you think these do?

- `xp`
- `ddp`
- `A;<Esc>`

What's to Come?

The next few sections should hopefully have something new for everyone. Indeed it is these sections that really make up Vim's power.

Whetting appetite - the g command

- `gq` - indent
- `gd` - goto local declaration
- `gD` - goto first Declaration in the file
- `gf` - goto the file (`gF` works too)
- `gj`, `gk` - move up and down with warped text
- `ga` - give me the ascii value

Oh, you Completeth me

- Simple History Based Completion - `C-n`, `C-p`
- Complete filenames - `C-x C-f`
- *UberCool* Feature - Omni Complete `C-x C-o`

Treat me like Text

- Match parantheses - `%`
- Words - `w`, `e`, `b`
- Lines - `0`, `$` (Remember Regexps ?)
- Sentences - `(`, `)`
- Paragraphs - `{`, `}`
- Code Blocks (till the nearest { or }) - `[[`, `]]`

I can jump, accurately

- The `f` command. (`F`)
- The `t` command. (`t`)

Examples:

- `f.` - goto the nearest dot
- `t)a,`- go till the nearest ')', and then append and place a comma. (Adding an extra argument to the function)

Text Objects - Who said you had to mug?

Very powerful

Choose the action. Choose how the text object is to be interpreted. Choose the text object. And it makes perfect sense as a whole too!

- Delete `d` or Change `c`
- Inside `i` or the whole thing `a` or from here till end (nothing)
- Text Object

Examples :

- `das` - delete the sentence we are in
- `ci)` - change the parts within the paranthesis
- `daB` - delete an inner Block of code {}

More Examples

More Examples:

- `s` - sentence
- `w` - word
- `W` - WORD
- `t` - tag ('dat' removes from <xml-style-tag> till </xml-style-tag>
- `p` - paragraph
- `B` - block of code ('{' or '}' works only for C-style blocks)
- `(` or `)` - parenthesis (...)
- `"` - the string " ... "

Note: Uppercases vs. Lowercase `s`, `w`, etc..

Making projects

- `:set makeprg=make`
- Errors can be seen and viewed in the error buffers
  - `:cn` - next error
  - `:cN` - previous error

Splitting Windows

Splitting

- Vertical - `C-w v`
- Horizontal - `C-w s`

Moving between them is done by `C-w x` where `x` is one of `h, j, k, l`. Close with a simple `:q`.

Registers

Registers store information in Vim.

- `:registers` to view the registers (A few special registers. Also saved across sessions)
- Copy into or paste from registers by prefixing the command with the register name.
- Registers are addressed in the form `"x`.
- Captial letter tells vim to append when copying. (Example : Function declarations)

The + register, and the OS buffer.

Take me Home, Country Roads - Marks

- Like Registers, they have names as letters.
- Different namespace from Registers
- Mark a position with `mx` where x is the register name.
- Goto mark `x` with `'x`.
- `:marks` shows all the marks.

Capital Letters indicate global marks (from any file or buffer), whereas small letters are local to the file.
Marks can also be used as text objects, and motions.

Folds

- An amazing programmer tool.
- fold with `zf<motion>`
- open with `zo`
- can be nested
- open all levels of folds with `zR`
- delete a fold with `zE`

Different types of fold modes available `foldmethod`. `indent`, `syntax`, `diff`, `marker` are interesting.

Loading and Restoring Views

- `:mkview` - makes the current view
- `:loadview` - loads the current view

What they save is stored in `sessionoptions`

Interaction with the outside world

- Running shell commands `!` and throwing their output `r!`
- `tree` example
- `sort` example
- latex themes example

Miscellaneous Stuff

- The `J` and `K` commands. (The Table Example)
- Spell Check
- Incrementing and Decrementing Numbers
- Non printable characters -> `:list` and `:listchars`
- mapping <*Esc*> to capslock
- `vim -`

At a Later Date

- A more comprehensive Plugins survey
- Configuring your vimrc
- Macros -> Macros Macros
- ex, the editor in vim, you never knew
- The magic and nomagic of regexps
- Writing your own functions - python, perl and tcl are now supported by vim.
- Color themes, syntax files.
- File hooks, and snippets

Live Vim

Once the muscle memory is formed, exploit it everywhere

- vimperator
- vifm

Credits

Made with

- Pandoc
- LaTeX-beamer
- Vim

Thanks

Thanks for your time and patience.