

VIM

CRAFTSMAN'S PRECISION TOOL

“IF PEOPLE KNEW HOW HARD I HAD TO WORK TO
GAIN MY MASTERY, IT WOULD NOT SEEM SO
WONDERFUL AT ALL”

MICHELANGELO BUONARROTI

QUICK HISTORY

- ✿ 1971 - ed text editor by Ken Thompson



Applications unlimited . . .

TELETYPE Model 28 Line

Chemical company "wires" distant payroll . . . Tool supplier has instantaneous access to centralized inventory of 15,000 items . . . Railroad keeps track of 50,000 freight cars.

These are just a few examples of the variety of "assignments" Teletype equipment is handling today—helping business to cut costs . . . improve service . . . and cope with the ever-growing paperwork and communication needs generated by rapid growth and decentralization.

Presented here are various machines in the Teletype Model 28 Line. "Model 28" stands for an entirely new concept in record communica-

QUICK HISTORY

- ✿ 1971 - ed text editor by Ken Thompson
- ✿ 1976 - ex (EXtended) by Bill Joy
- ✿ 1976 - vi (visual mode for ex) by Bill Joy
- ✿ 1991 - vim (vi improved) by Bram Moolenaar

VIM IS NOT VI

- ➊ VI - ancient 1976 technology
- ➋ VIM - state of the art 1991 editor
- ➌ VIM introduced more new features, than vi had overall in its lifetime

YOU DON'T LIKE VIM

because you don't get it

YOU DON'T LIKE VIM

because you don't get it

and it's ok !



IT'S OK BRO

YOU DON'T LIKE VIM

because you don't get it

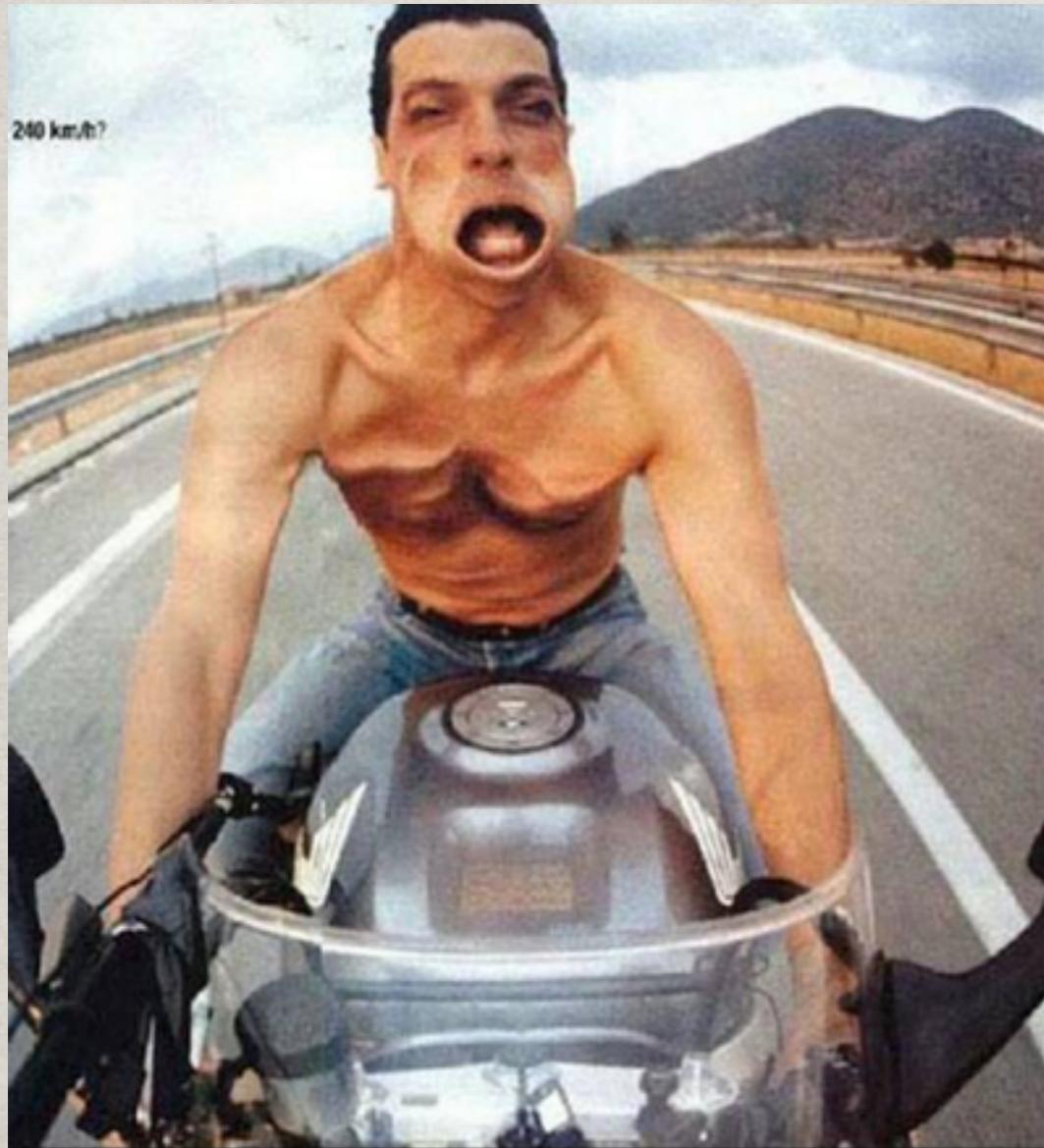
and it's ok!

- * it's different from newbie-friendly visual editors
- * it requires some (major) investment from the user, aka “the learning curve”

TO EACH HIS OWN

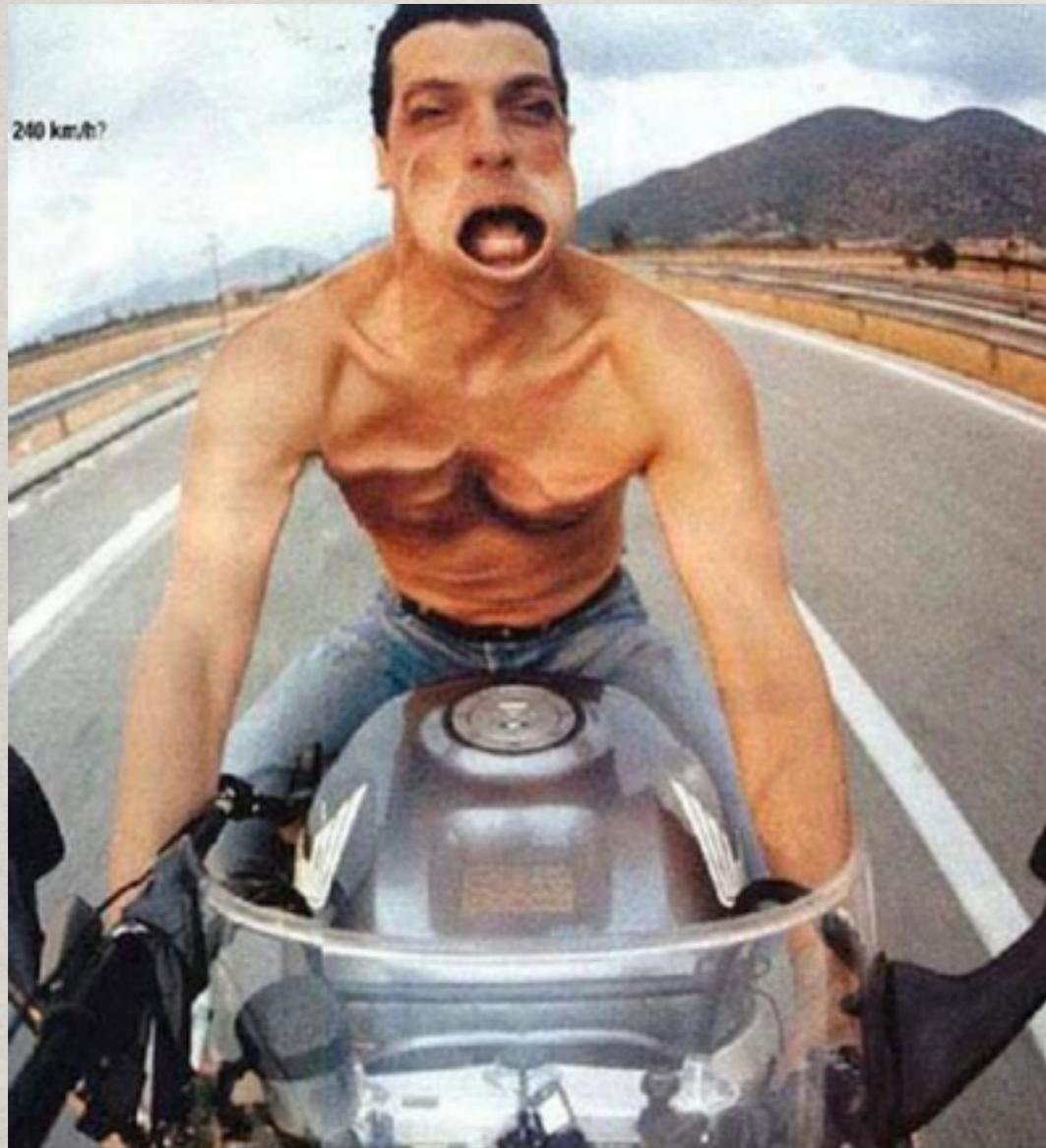
TO EACH HIS OWN

some like it straight



TO EACH HIS OWN

some like it straight



some like it curvy



THE MODAL EDITOR

version 1.1
April 1st, 06

vi / vim graphical cheat sheet

ESC normal mode															
~ toggle case	! external filter	@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :s	* next ident	(begin sentence) end sentence	"soft" bol down	+ next line			
\. goto mark	1 2 3 4 5 6 7 8 9 0 "hard" bol										- prev line	= auto-format			
Q ex mode	W next WORD	E end WORD	R replace mode	T back 'till	Y yank line	U undo line	I insert at bol	O open above	P paste before	{ begin parag.	}	end parag.			
q record macro	W next word	e end word	r replace char	t 'till	y yank	u undo	i insert mode	o open below	p paste after	[misc] misc			
A append at eol	S subst line	D delete to eol	F "back" find ch	G eof/ goto ln	H screen top	J join lines	K help	L screen bottom	: ex cmd	": reg. 1 spec		bol/ goto col			
a append	s subst char	d delete 1,3	f find char	g extra 6 emds	h ←	j ↓	k ↑	l →	line	t/T/f/F	' goto mk. bol	\ not used!			
Z quit 4	X back-space	C change to eol	V visual lines	B prev WORD	N prev (find)	M screen mid'l	< un- 3 indent	> indent 3	? find (rev.)						
Z extra 5 cmd	X delete char	c change 1,3	v visual mode	b prev word	n next (find)	m set mark	,	, reverse t/T/f/F	/ 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:

- 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')
- type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)
- duplicate operator to act on current line (dd = delete line, >> = indent line)
- ZZ to save & quit, ZQ to quit w/o saving
- zt: scroll cursor to top, zb: bottom, zz: center
- 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

THE MODAL EDITOR

Every BUTTON performs a different
function in a different mode

THE MODAL EDITOR

Every BUTTON performs a different
function in a different mode

do you think it's weird and uncommon?
think again

THE MODAL WORLD

What would you rather use?

This
(could you even tell
what every button
is for?)



THE MODAL WORLD

What would you rather use?

This
(could you even tell
what every button
is for?)



or this

THE MODAL MIND

You're a craftsman - think first, edit later

THE MODAL MIND

You're a craftsman - think first, edit later

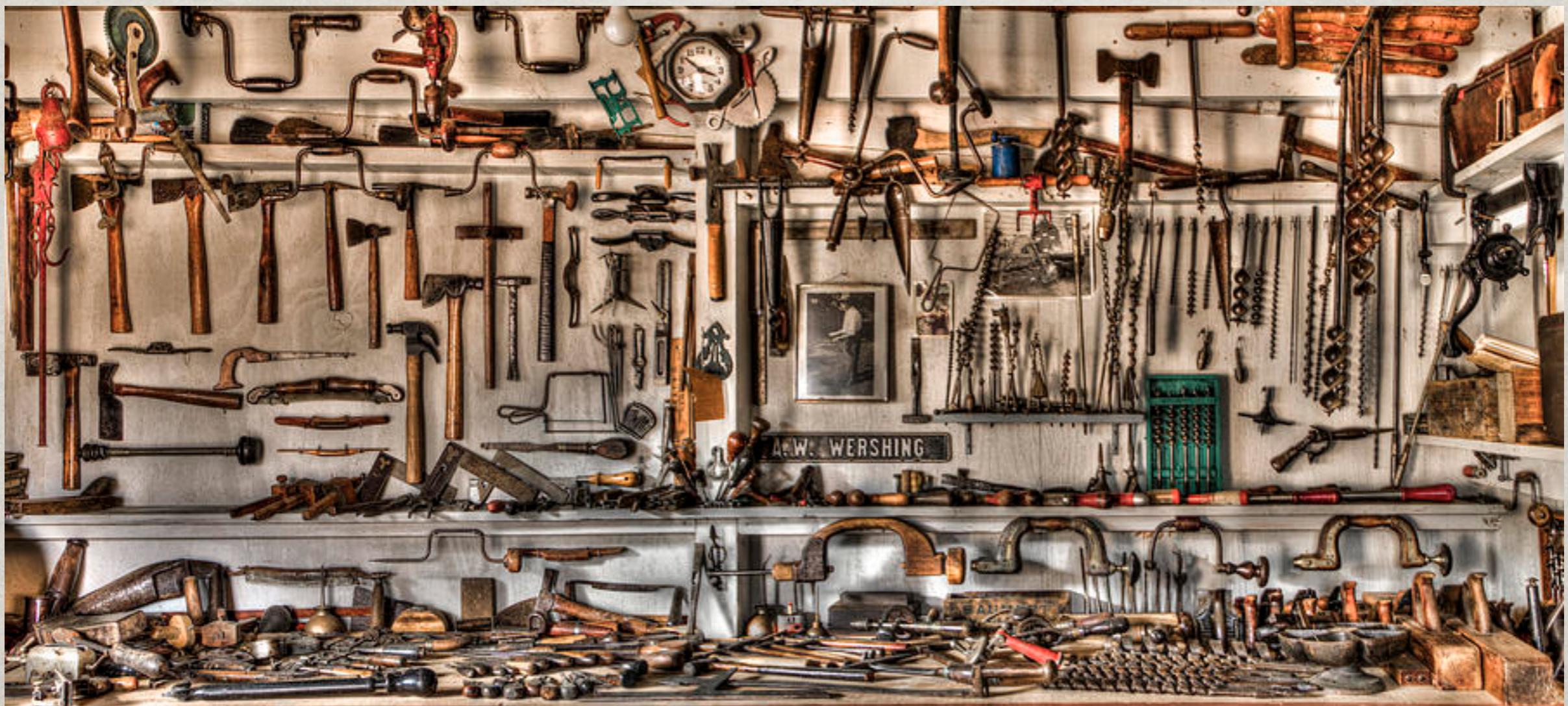
think about what you're doing



THE MODAL MIND

You're a craftsman - think first, edit later

think about how you're doing it



THE MODAL MIND

You're a craftsman - think first, edit later

and

ONLY THEN

do it

MOVING AROUND

IDEAL: follow the eye

MOVING AROUND

IDEAL: follow the eye

Mouse is actually close enough

MOVING AROUND

IDEAL: follow the eye

Mouse is actually close enough

Especially with enough target practice



MOVING SLOWLY

But mouse targeting is slow

MOVING SLOWLY

But mouse targeting is slow

FITT'S LAW

The time required to rapidly move to a target area
is a function of the distance to and the size of the
target

MOVING SLOWLY

But mouse targeting is slow
and tiresome and boring

```
while (!onTarget) do
    moveTowardsTarget
end
```

so many wasted brain cycles

MOVING SLOWLY

But mouse targeting is slow
and tiresome and boring
and breaks the flow

MOVING ROUGHLY

Keyboard is usually ok (even in Notepad)

imaging doing that on that teletype with unlimited applications...

MOVING ROUGHLY

Keyboard is usually ok (even in Notepad)
but only if you need to hit BIG targets
(top/bottom of the file, start/end of the line)

imaging doing that on that teletype with unlimited applications...

MOVING ROUGHLY

Keyboard is usually ok (even in Notepad)
but only if you need to hit BIG targets
(top/bottom of the file, start/end of the line)

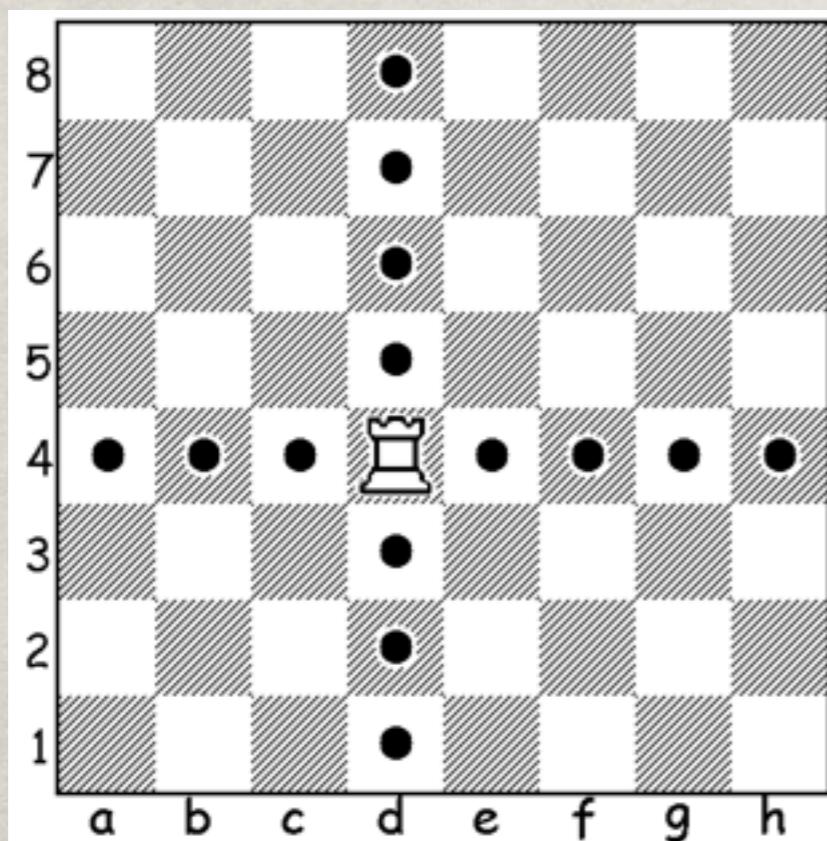
Hunting for smaller targets (specific chars/lines):

```
while (!onTarget) do
    moveTowardsTarget
end
```

imaging doing that on that teletype with unlimited applications...

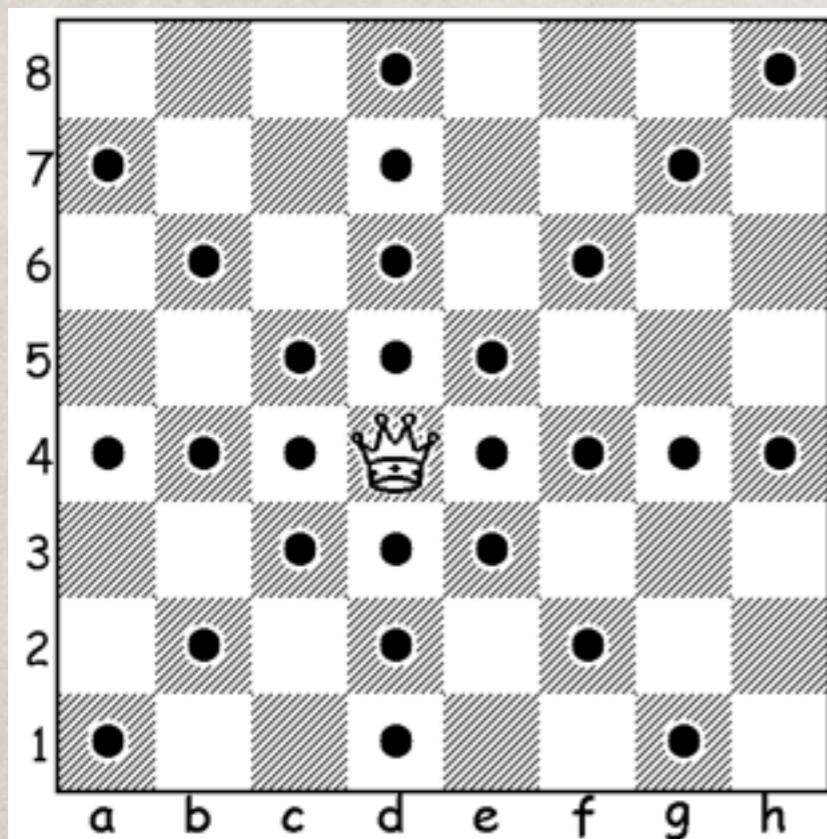
MOVING PATTERNS

average
text editor



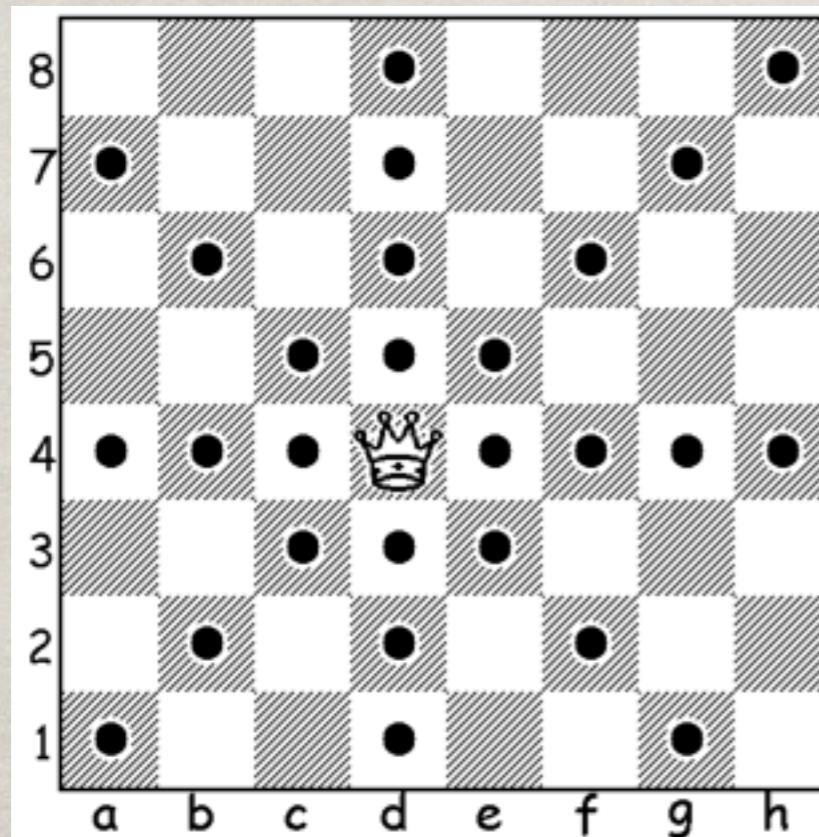
MOVING PATTERNS

Above
average
text editor

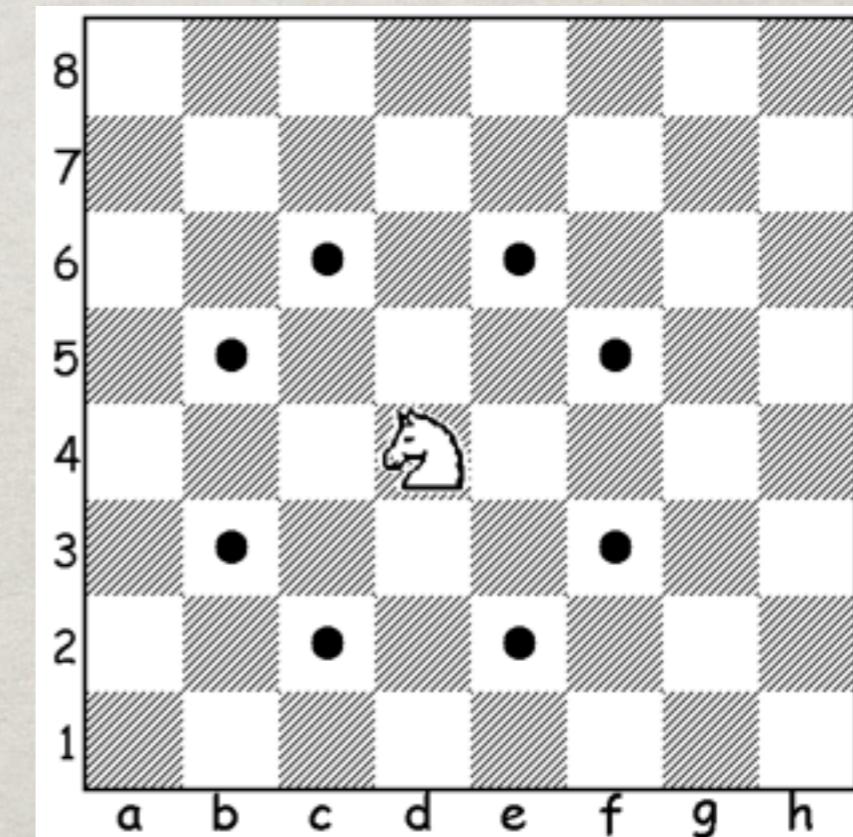


MOVING PATTERNS

Above
average
text editor

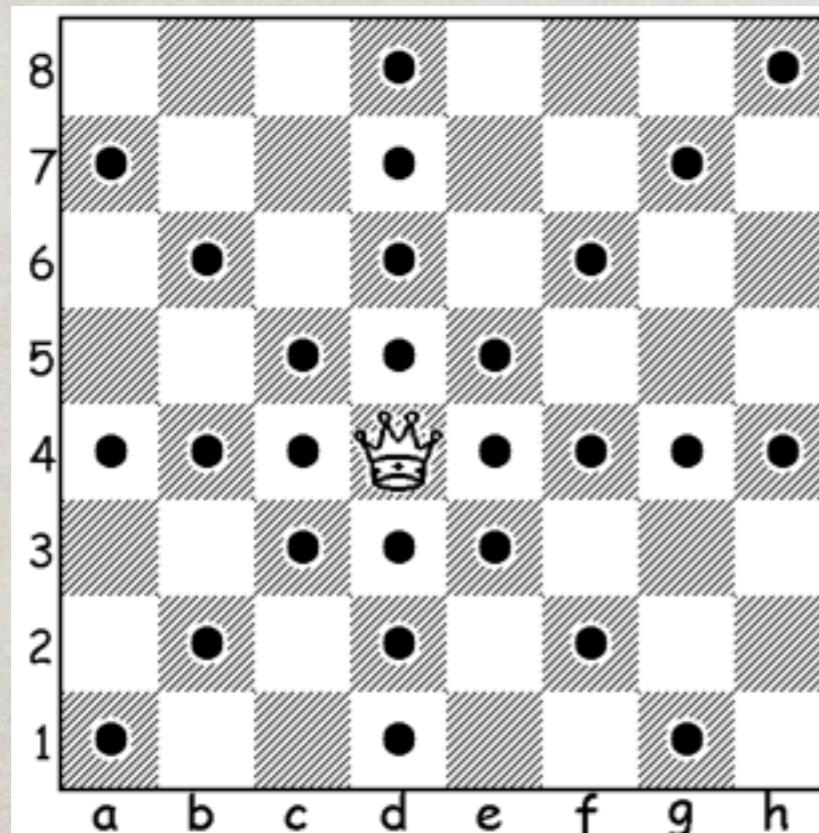


VIM

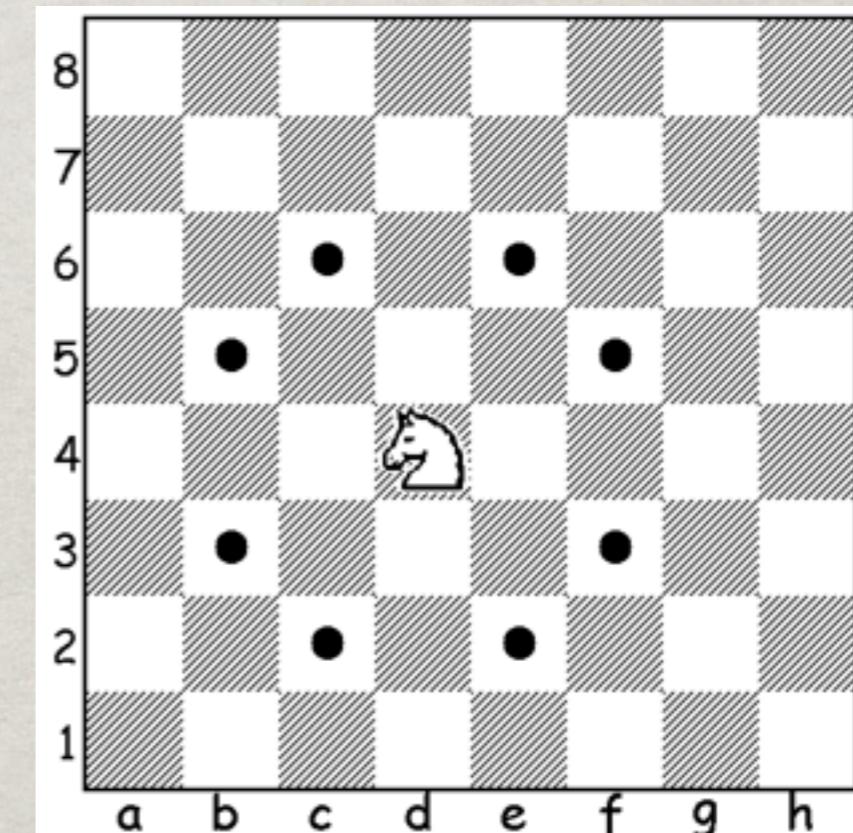


MOVING PATTERNS

Above
average
text editor



VIM



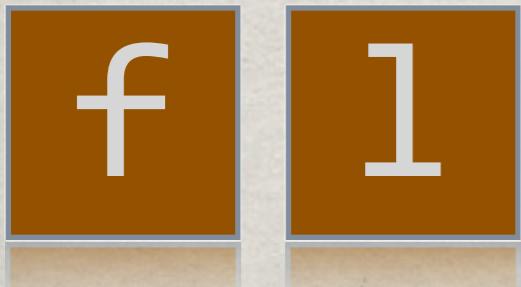
Knight in a single move hits a target
that takes a queen 2 moves to hit

TARGET HITTING 101

The quick brown fox jumps over the lazy dog.

TARGET HITTING 101

The quick brown fox jumps over the lazy dog.



The quick brown fox jumps over the lazy dog.

TARGET HITTING 102

The quick brown fox jumps over the lazy dog.

TARGET HITTING 102

The quick brown fox jumps over the lazy dog.



The quick brown fox jumps over the lazy dog.

TARGET HITTING 102

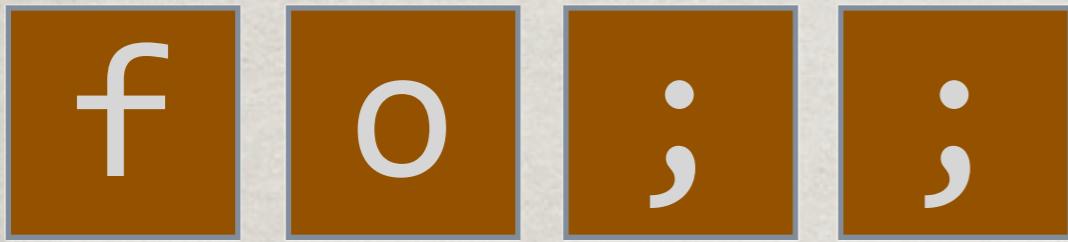
The quick brown fox jumps over the lazy dog.

f o ;

The quick brown fox jumps over the lazy dog.

TARGET HITTING 102

The quick brown fox jumps over the lazy dog.



The quick brown fox jumps over the lazy dog.

TARGET HITTING 102

The quick brown fox jumps over the lazy dog.



The quick brown fox jumps over the lazy dog.

TARGET SELECTION

- ✿ Better targets:
- ✿ punctuation
- ✿ special characters (@, &, \$)
- ✿ uppercase letters
- ✿ etc

TARGET SEARCH

- ✿ Learn to love search - /
- ✿ Pro-tip: enable find-as-you-type
- ✿ Learn current word lookups (*, #)

DRONE STRIKE

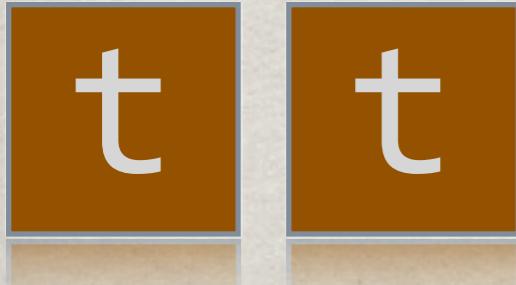
The quick brown fox jumps over the lazy dog.

destroy

DRONE STRIKE

The quick brown fox jumps over the lazy dog.

destroy

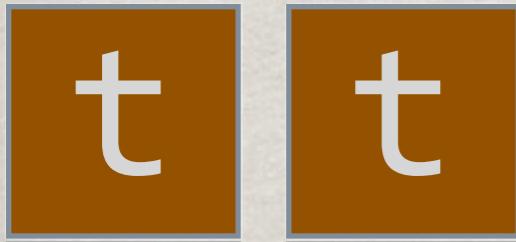


The quick brown fox jumps over the lazy dog.

DRONE STRIKE

The quick brown fox jumps over the lazy dog.

destroy



The quick brown fox jumps over the lazy dog.



The quick brown fox jumps over.

TEXT OBJECTS

Program code is full of blocks

Easy matching:

- **a?** - match the whole block
- **i?** - match insides of the block

Quotes: a"
 i"

Braces: a}
 i}

Round a)
brackets: i)

Tags: at
 it

TEXT OBJECTS 101

```
params = {  
    users: [ "Mal Reynolds", "Inara Serra" ]  
}
```

TEXT OBJECTS 101

```
params = {  
  users: [ "Mal Reynolds", "Inara Serra" ]  
}
```



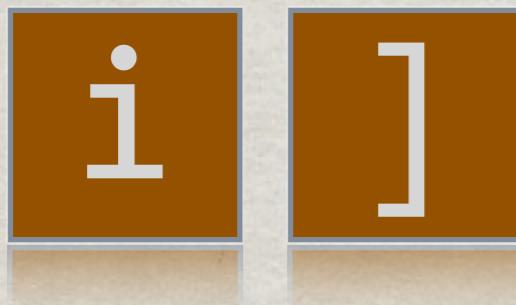
```
params = {  
  users: [ "Mal Reynolds", "Inara Serra" ]  
}
```

TEXT OBJECTS 101

```
params = {  
  users: [ "Mal Reynolds", "Inara Serra" ]  
}
```

TEXT OBJECTS 101

```
params = {  
  users: [ "Mal Reynolds", "Inara Serra" ]  
}
```



```
params = {  
  users: [ "Mal Reynolds", "Inara Serra" ]  
}
```

TEXT OBJECTS 101

```
params = {  
    users: ["Mal Reynolds", "Inara Serra"]  
}
```

TEXT OBJECTS 101

```
params = {  
  users: [ "Mal Reynolds", "Inara Serra" ]  
}
```



```
params = {  
  users: [ "Mal Reynolds", "Inara Serra" ]  
}
```

TEXT OBJECTS 101

same for Ruby blocks (with a plugin)

`ar` - select the whole Ruby block

`ir` - select inside Ruby block

TEXT OBJECTS 101

```
params = {  
  users: [ "Mal Reynolds", "Inara Serra" ]  
}
```



```
params = [
```

CHANGES

- ✿ Undo change - u
- ✿ Repeat change - .
- ✿ Shines when coupled with modal mode
- ✿ Think of your changes as steps, or brushstrokes

PRO TIPS

- ✿ Remap your ESC
- ✿ Disable your cursor keys
- ✿ Don't use others' configs in full
- ✿ Train yourself gradually
- ✿ But either dive in, or stay away

PERSONAL FAVORITES

- ✿ VIM running in pure text console mode
- ✿ Ctrl-Z to fall back into the shell
- ✿ Use external commands to process the data
- ✿ Coupled with tmux integration
- ✿ Quickly launch separate tasks (e.g. rspec) in tmux window

LOVE YOUR EDITOR

