# 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 Moolenar

# VIM is not VI

- VI - ancient 1976 technology
- VIM - state of the art 1991 editor
- 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!

# YOU DON'T LIKE VIM

because you don't get it

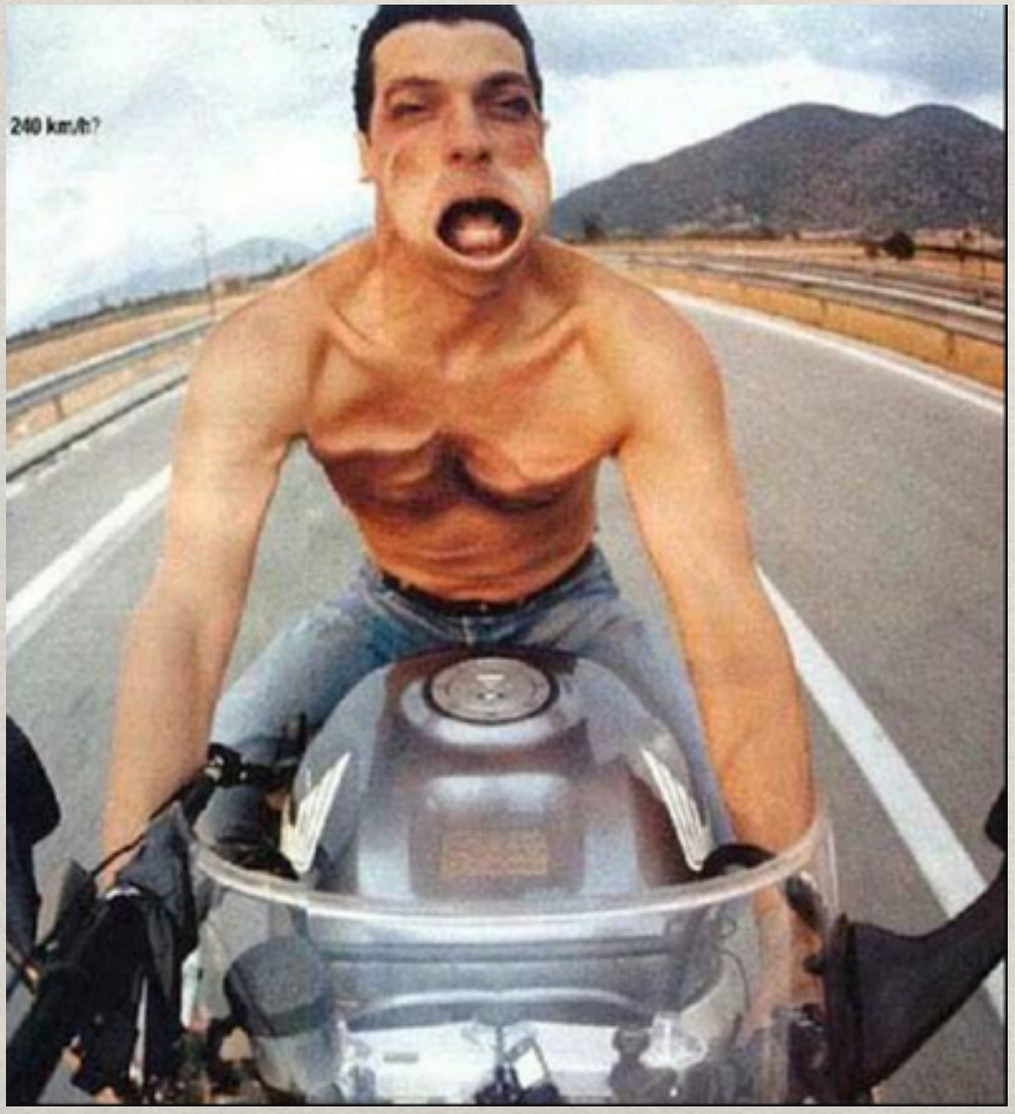
and it's ok!

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

# TO EACH HIS OWN

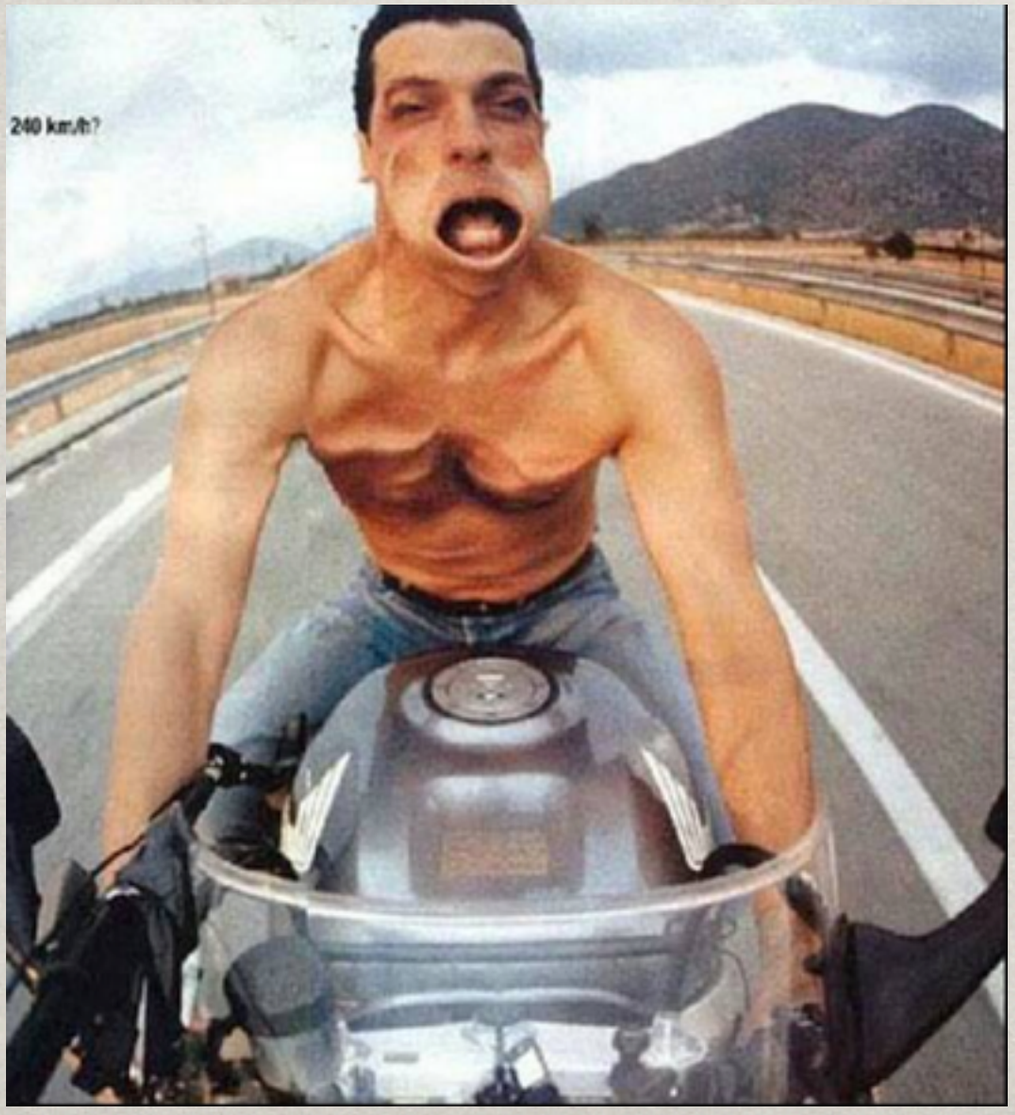
# TO EACH HIS OWN

some like it straight



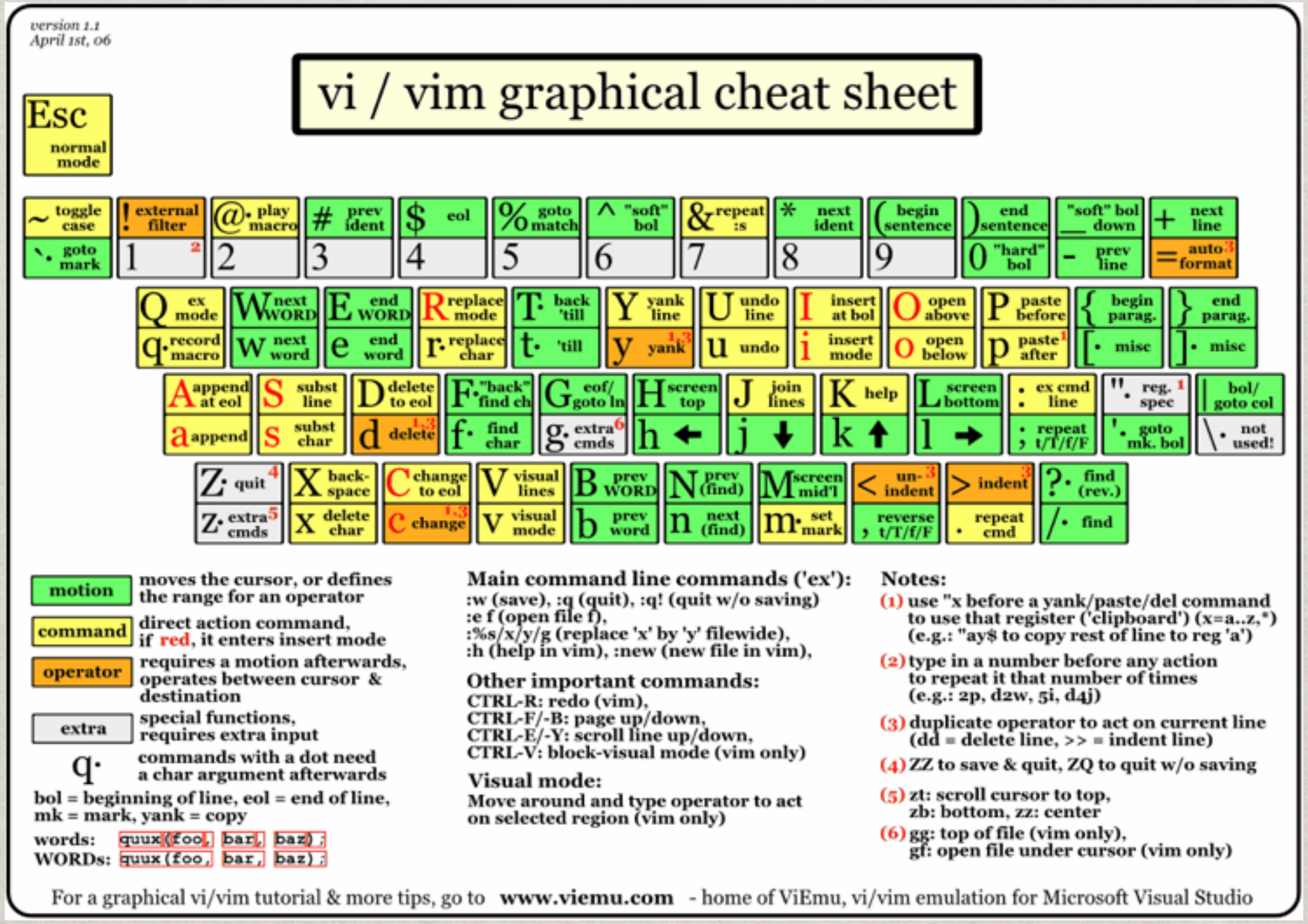
# TO EACH HIS OWN

some like it straight




some like it curvy

# THE MODAL EDITOR



# 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

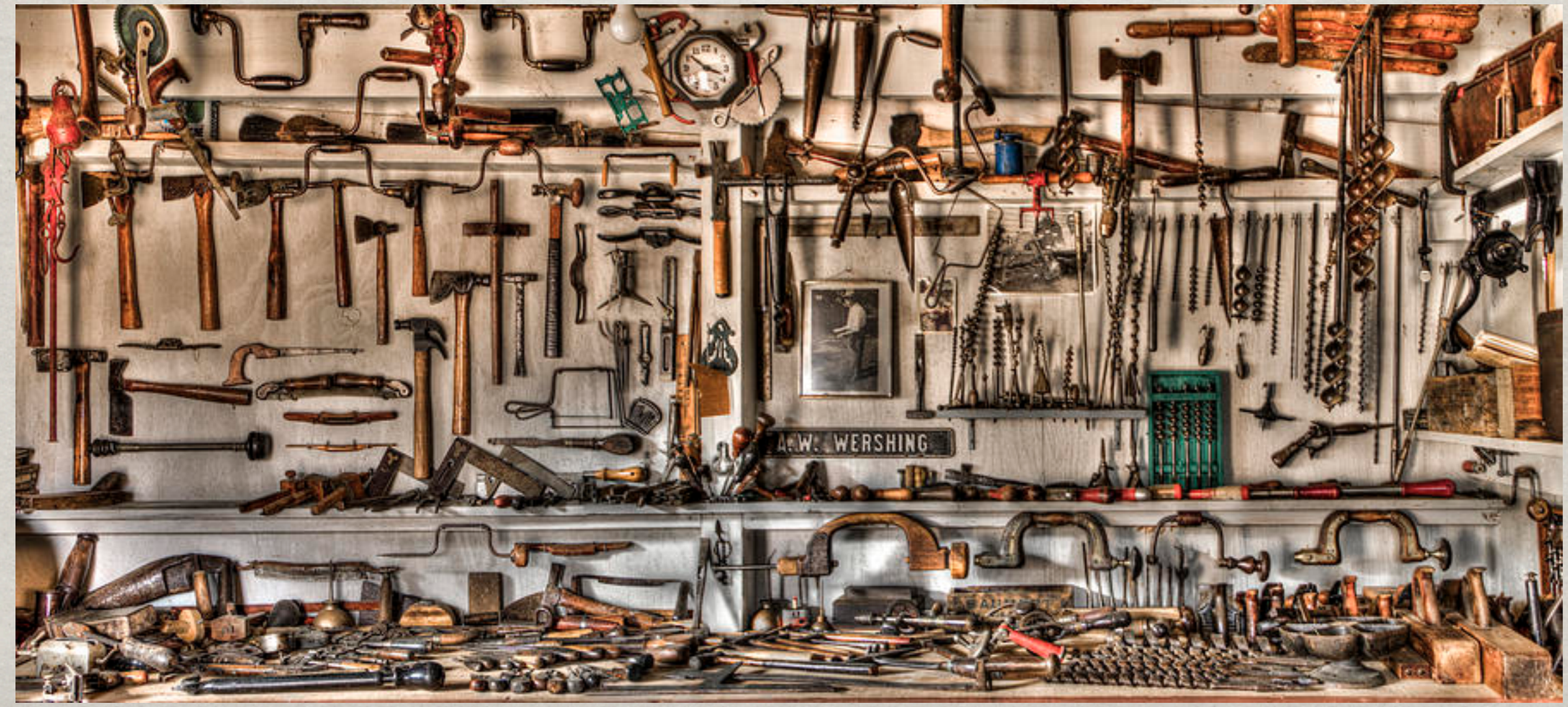
You're a craftsman - think first, edit later

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)

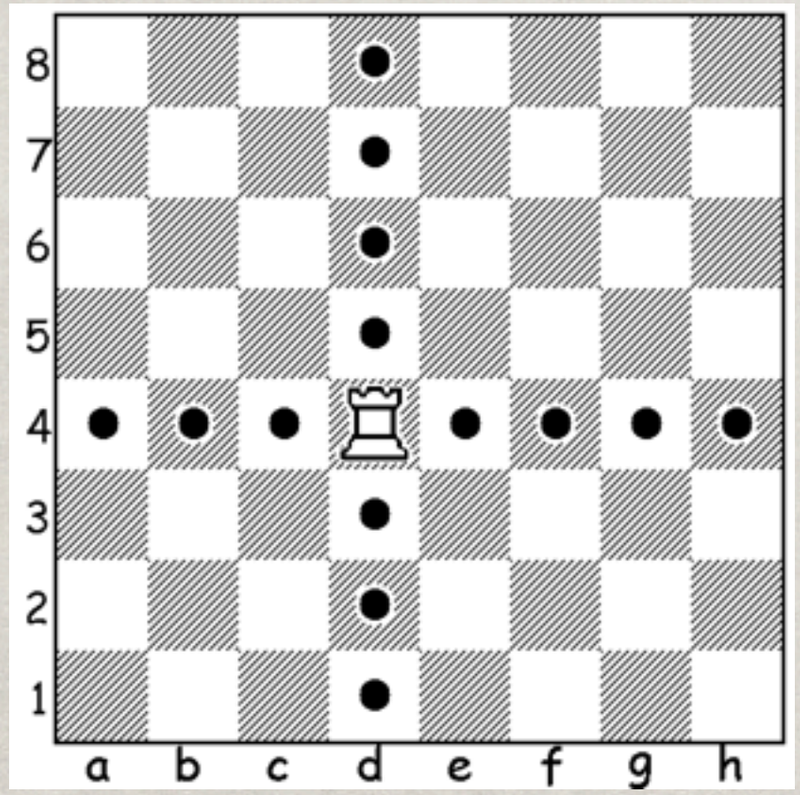
Hunting for smaller targets (specific chars/lines):

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

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

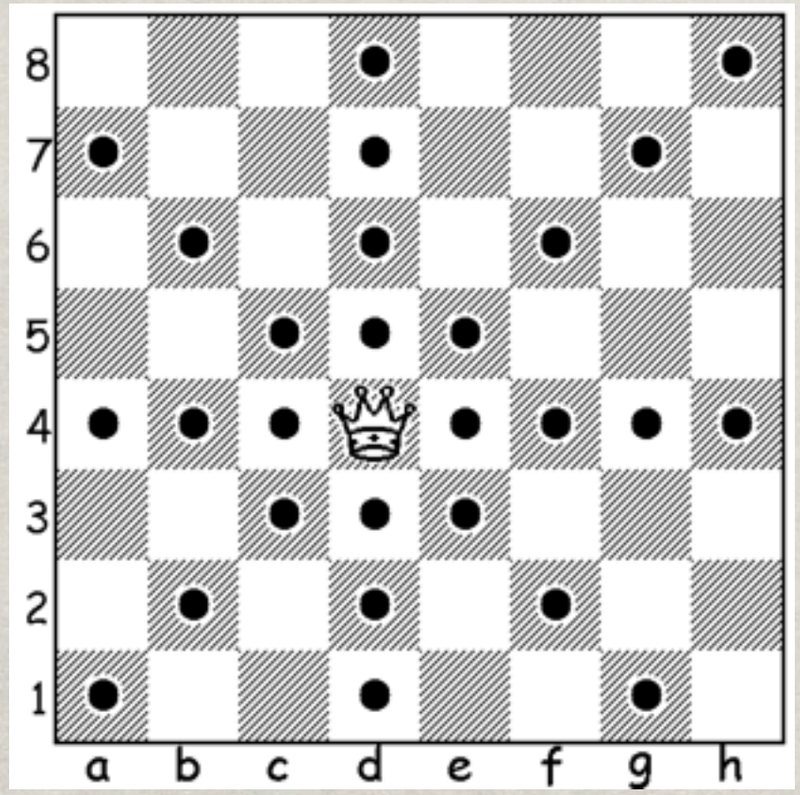
# MOVING PATTERNS

average
text editor



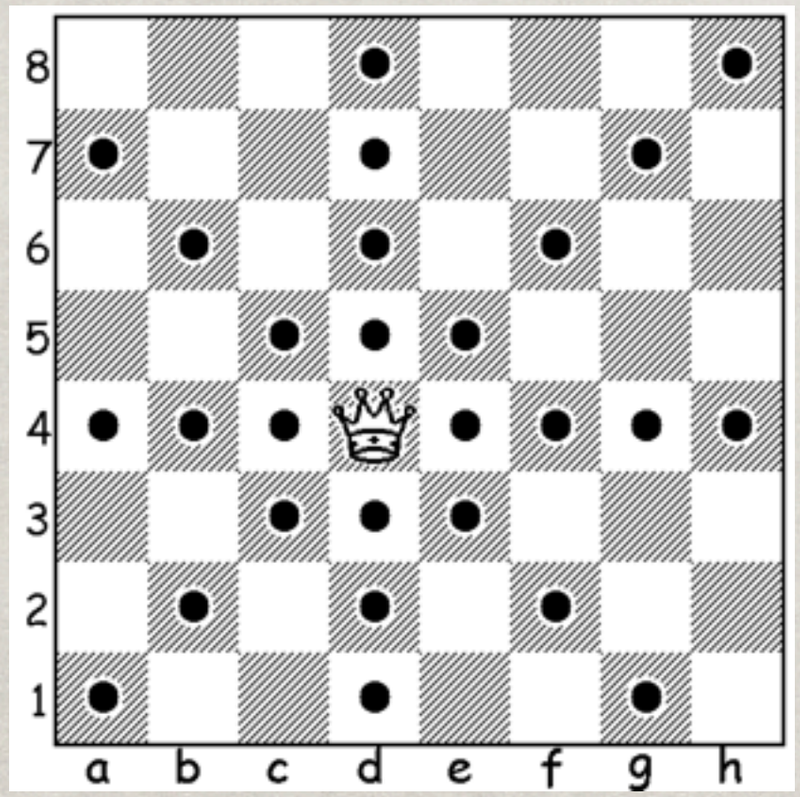
# MOVING PATTERNS

Above
average
text editor

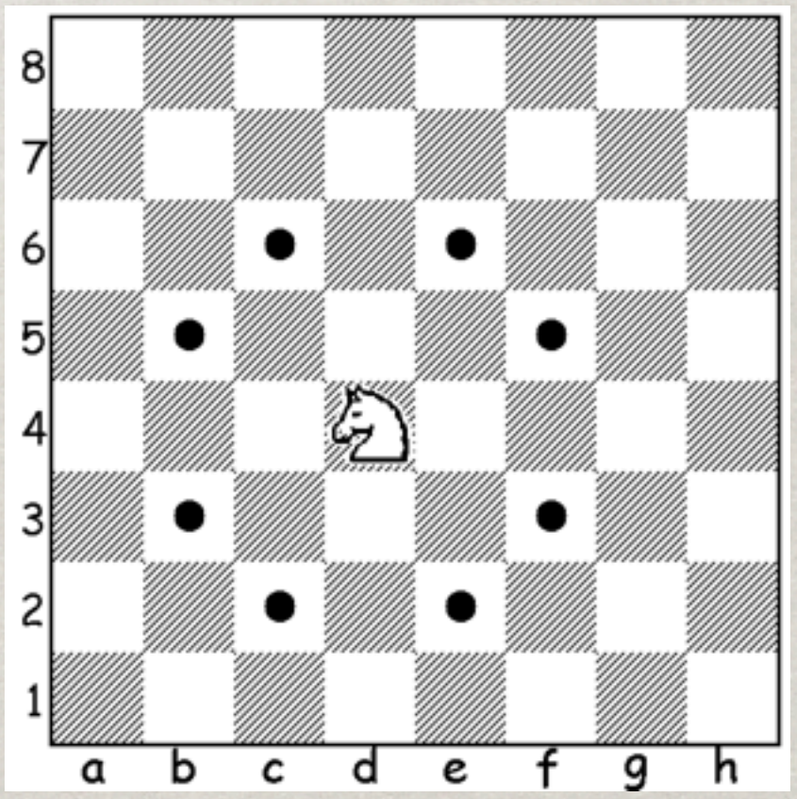


# MOVING PATTERNS

Above
average
text editor

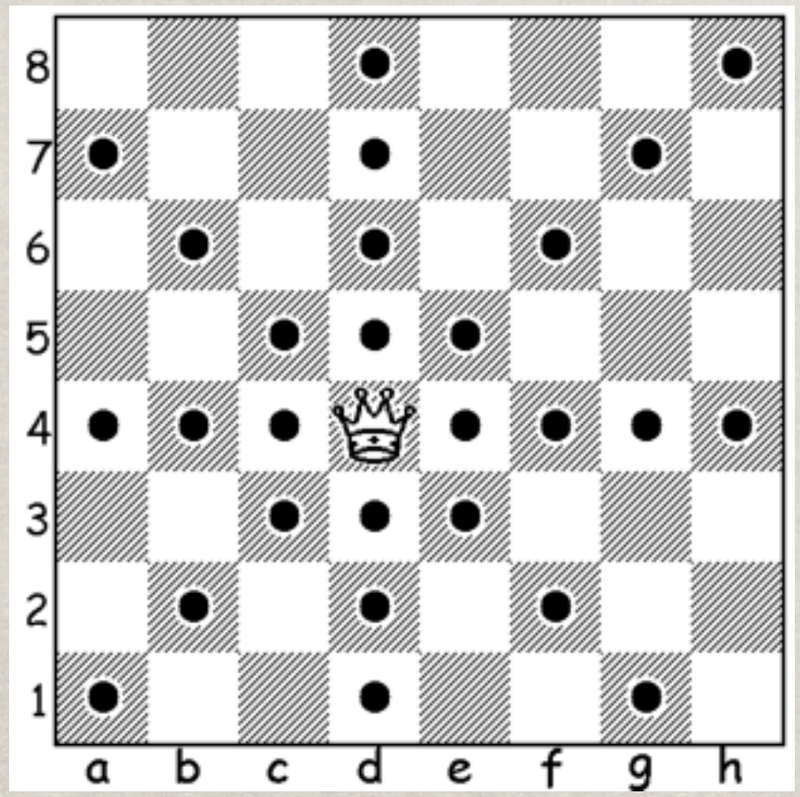



VIM

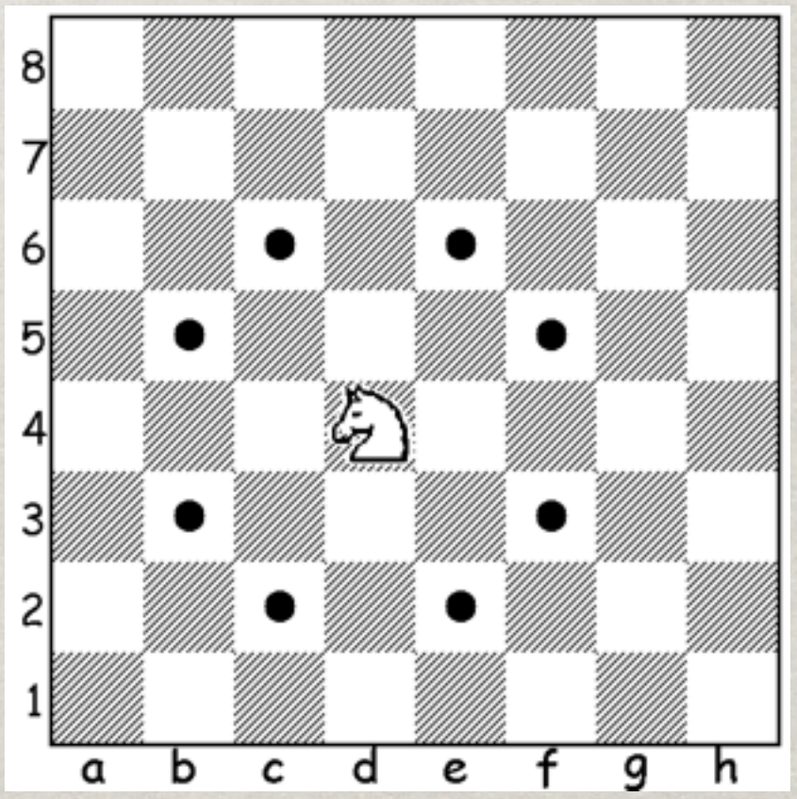


# MOVING PATTERNS

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.

f l

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.

f o

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.

f o ; ;

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 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

t t

The quick brown fox jumps over the lazy dog.

# DRONE STRIKE

The quick brown fox jumps over the lazy dog.

destroy

t t

The quick brown fox jumps over the lazy dog.

d t .

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”`<br>`i”` | Braces: | `a}`<br>`i}` |
| Round brackets: | `a)`<br>`i)` | Tags: | `at`<br>`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

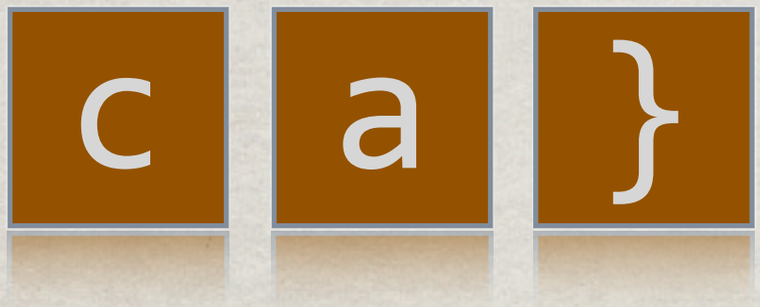
same for Ruby blocks (with a plugin)

`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