

# vim

“Who needs a mouse?”

Nicolas Dorrmann

04. Nov. 2020

# History and Design Philosophy

- ▶ Developed by Bram Moolenaar, initially released in 1991.
- ▶ Successor to `vi` (**v**isual **i**nterface for line editor `ex`).
- ▶ Editing works entirely over keyboard, most commands are easily reachable over home row.
- ▶ Plugins (written in VimScript) *can* provide functionality similar to an IDE.

# But why?

- ▶ It's everywhere (if you want it) — a lot of software has the option of using vim keybindings.

`qutebrowser` webbrowser

`vifm` filemanager

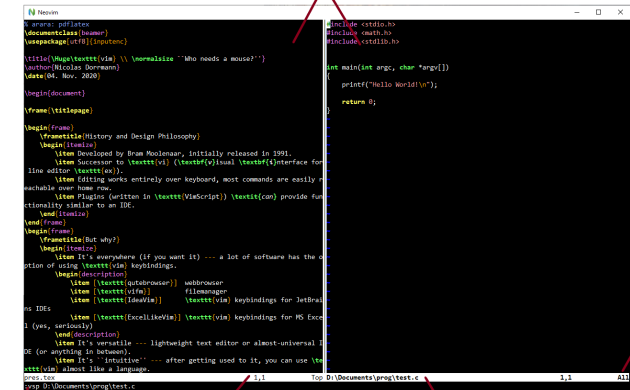
`IdeaVim` vim keybindings for JetBrains IDEs

`ExcelLikeVim` vim keybindings for MS Excel (yes, seriously)

- ▶ It's versatile — lightweight text editor or almost-universal IDE (or anything in between).
- ▶ It's “intuitive” — after getting used to it, you can use vim almost like a language.

# Basics - The Interface

## Buffers



The screenshot shows the Neovim editor interface with two buffers open. The left buffer contains a document with a title, author, date, and a list of features. The right buffer contains a C program. Annotations with red arrows point to various parts of the interface:

- Buffer**: Points to the top of the right buffer.
- Buffer Position**: Points to the line number 11 in the right buffer.
- Cursor Position**: Points to the cursor at line 1, column 1 in the left buffer.
- Buffer Filename**: Points to the filename 'Top D:\Documents\prog\test.c' in the bottom right.
- Command Line**: Points to the command line at the bottom left showing 'prog D:\Documents\prog\test.c'.

```
1: setruler
2: documentclass[beamer]
3: \usepackage[utf8]{inputenc}
4:
5: \title{Huge\texttt{vim} \& \normalize "Who needs a mouse?"}
6: \author{Nicolas Dornemann}
7: \date{04. Nov. 2020}
8:
9: \begin{document}
10: \frame{\titlepage}
11:
12: \begin{frame}
13:   \frame{title[History and Design Philosophy]}
14:   \begin{itemize}
15:     \item Developed by Bram Moolenaar, initially released in 1991.
16:     \item Successor to \texttt{vi} (\texttt{w} \texttt{visual} \texttt{s} \texttt{interface} for
17:       line editor \texttt{ex}).
18:     \item Editing works entirely over keyboard, most commands are easily r
19:       eachable over home row.
20:     \item Plugins (written in \texttt{VimScript}) \texttt{can} provide fun
21:       ctionality similar to an IDE.
22:   \end{itemize}
23: \end{frame}
24:
25: \begin{frame}
26:   \frame{title[But why?]}
27:   \begin{itemize}
28:     \item It's everywhere (if you want it) --- a lot of software has the o
29:       ption of using \texttt{vim} keybindings.
30:     \begin{description}
31:       \item [\texttt{q} \texttt{webbrowser}] webbrowser
32:       \item [\texttt{f} \texttt{filemanager}] filemanager
33:       \item [\texttt{I} \texttt{IdeaVim}] \texttt{vim} keybindings for JetBrains
34:     \end{description}
35:     \item [\texttt{E} \texttt{ExcelLikeVim}] \texttt{vim} keybindings for MS Excel
36:   \end{itemize}
37: \end{frame}
38:
39: \begin{frame}
40:   \frame{title[Yes, seriously]}
41:   \begin{description}
42:     \item It's versatile --- lightweight text editor or almost-universal I
43:       DE (or anything in between).
44:     \item It's "intuitive" --- after getting used to it, you can use \texttt{
45:       vim} almost like a language.
46:   \end{description}
47: \end{frame}
48: \end{document}
49:
50: pres.tex
51: prog D:\Documents\prog\test.c
```

```
1: #include <stdio.h>
2: #include <math.h>
3: #include <stdlib.h>
4:
5: int main(int argc, char *argv[])
6: {
7:     printf("Hello World!\n");
8:
9:     return 0;
10: }
```

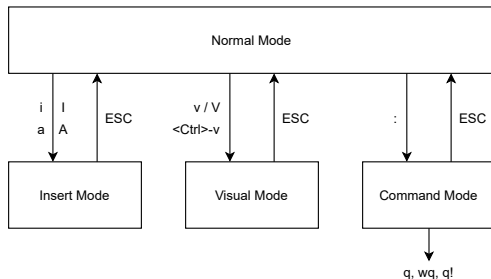
# Basics - Modes

**NORMAL** navigate within the file, issue text commands  
(yank/change/delete)

**COMMAND** open, close, or save a file

**INSERT** direct input (i. e. actually typing text)

**VISUAL** text highlighting, text blocking



# Basics

## ► Actions (“verbs”)

**yank** copy a movement

**change** replace object with a new object

**delete** remove an object

## ► Modifiers

**NUM** number of objects to perform action on

**i** perform action **i**nside an object

**a** perform action **a**round an object

**t** perform action up **t**o an object

**f** perform action up to and including an object

## ► Text Objects (“nouns”)

**word** string of non-whitespace characters surrounded by whitespace

**sentence** ends with **!**, **?**, **.** followed by newline

**paragraph** consists of several sentences, delimited by empty lines

# Commands

- ▶ Basic commands have the form Action — Modifier — Object
- ▶ Examples
  - `y2w` copy the next two words
  - `dis` delete the current sentence (i. e. the sentence the cursor is in)
  - `cp` change (i. e. delete and enter INSERT mode) the current paragraph

# Search and Replace

- ▶ Search
  - ▶ Forward search with `/string`. Backward search with `?string`.
  - ▶ Next result with `n`, previous with `N`.
- ▶ Search and Replace (substitute)
  - ▶ Basic syntax: `: [range] s/search/replace/`
  - ▶ `[range]` is a line, range of lines (`start`, `end`), or entire buffer (`%`).
  - ▶ Global replace with `g` or with confirmation (`gc`).



# Regular Expressions

- . match one of any character
- [range] matches any one of the characters in range
  - \* match arbitrary many repetitions of preceding character
  - \+ match one or more of the preceding character
  - \= match none or one of the preceding character
- {m,n} between m and n repetitions
  - ^ match start of a line
  - \$ match end of a line

## `.vimrc` / `init.vim`

- ▶ Configuration file for vim settings.
- ▶ Written in VimScript, an interpreted programming language.
- ▶ Read and executed at editor startup (provided it exists).
- ▶ Allows various customization options:
  - ▶ font, syntax highlighting and color scheme
  - ▶ define or redefine keybindings
  - ▶ modify various preset options

# VimScript Basics

`(i/c/v/n)map` (re)map a key combination  
`set` (re)define an option  
`let` (re)define a variable  
`source` “include” an additional .vim file  
`:help xyz` open documentation on xyz