Why Vim is My Favourite Text Editor

CT3112 Professional Skills: Assignment 01

Andrew Hayes (ID: 21321503)

University of Galway

2024–02–22

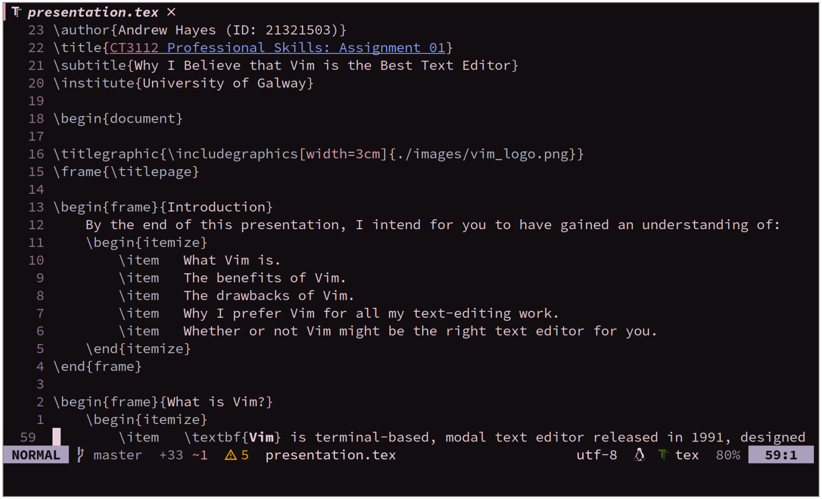
Introduction

By the end of this presentation, I intend for you to have gained an understanding of:

- What Vim is and how it works.
- The benefits of Vim.
- The drawbacks of Vim.
- Why I prefer Vim for all my text-editing work.
- Whether or not Vim might be the right text editor for you.
- What alternatives there are to Vim.
- How you can get started with Vim.

What is Vim?

- **Vim** is terminal-based, modal text editor released in 1991, designed to be minimal & fast to use.
- It has a number of fast, mnemonic keybindings that make typical text-editing tasks much faster.
- A **terminal-based** text editor is a text-only editor that is ran from the command line or terminal.
- A **modal** text editor is one in which there are a number of different **modes** that the editor can be in at any one time.



```
presentation.tex ×
\author{Andrew Hayes (ID: 21321503)}
\title{CT3112 Professional Skills: Assignment 01}
\subtitle{Why I Believe that Vim is the Best Text Editor}
\institute{University of Galway}

\begin{document}

\titlegraphic{\includegraphics[width=3cm]{./images/vim_logo.png}}
\frame{\titlepage}

\begin{frame}{Introduction}
    By the end of this presentation, I intend for you to have gained an understanding of:
    \begin{itemize}
        \item   What Vim is.
        \item   The benefits of Vim.
        \item   The drawbacks of Vim.
        \item   Why I prefer Vim for all my text-editing work.
        \item   Whether or not Vim might be the right text editor for you.
    \end{itemize}
\end{frame}

\begin{frame}{What is Vim?}
    \begin{itemize}
        \item   \textbf{Vim} is terminal-based, modal text editor released in 1991, designed
NORMAL  master  +33 ~1  ⚠ 5  presentation.tex                utf-8   tex  80%  59:1
```

Figure: A Screenshot of Vim Being Used to Write this Presentation (in LaTeX)

What is a Terminal?

- A **terminal** is a text-based interface for a computer, originating from when computers did not have any graphics.
- It is typically interacted with via a **command line**, where text commands are written to start programs.
- To start Vim from the command line, simply type `vim file_name.txt`

Figure: A Computer Terminal from 1978

What are the Vim Modes?

- **Normal mode** (`ESC`) is for the navigation & manipulation of the text in the file being edited, via the shortcut keybindings. It is the default mode of the program.
- **Insert mode** (`i`) is for inserting new text. This mode is similar to the default to the default behaviour of a more conventional text editor such as Notepad: text can be typed or removed with the backspace key, and navigation or selection can be done with the mouse or arrow keys.
- **Visual mode** (`v`) is for the selection of text blocks for manipulation, with the selected text being highlighted, similar to selecting text with the mouse in other text editors. Visual mode has two sub-modes: **visual block** (`CTRL+v`) & **visual line** (`V`), in which text can be selected either vertically by columns or horizontally by line.

What are the Vim keybindings?

- Too many to list here!
- Navigation in normal mode can be done with the `hjkl` (direction) keys, which correspond to the left, down, up, & right arrow keys respectively, allowing quick navigation without removing your fingers from the home row.
- These direction keys can be combined with numbers to repeat them a certain number of times, e.g.: `5j` moves the cursor down 5 lines.
- There are also a number of other direction keys: `w` to go forward a word of text, `b` to go back a word of text, `gg` to go to the start of the file, `G` to go to the end of the file.

What are the Vim keybindings?

- Vim keybindings typically take the form **action + direction**, i.e. the first key pressed indicates the action to be performed and the second key pressed indicates the direction in which to do it.
- The most common action keys are `d` to delete text, `c` to change text, `y` to copy (or **yank**) text, and `p` to paste text.
- The action keys are combined with direction keys, e.g. `dG` deletes all the text from the cursor to the end of the file, `y5w` copies the next 5 words to the clipboard, etc.
- If you're trying to get the hang of Vim keybindings, the most important one for you to know is `u` to undo the last action you performed, in case you made a mistake!

What are the benefits of Vim?

- Lightweight & minimal.
- Speedy text editing with countless keybindings & shortcuts.
- Easy integration with other command-line programs.
- Endlessly extensible through its configuration file (found at `~/.vimrc`)
- Lots of community-developed plugins to extend its functionality even further.
- It's very easy to quickly record a **macro** to perform an option repeatedly.
- Standard on most modern UNIX-like systems; if you use a Linux-based system or MacOS, you likely already have it installed!
- Most IDEs have a Vim mode, allowing you to use the Vim keybindings in many other programs too.

What are the drawbacks of Vim?

- Steep learning curve: there are so many keybindings that almost nobody knows them all.
- No graphics support: because it's terminal-based, there is no way to display an image inside Vim.
- Only works for plaintext editing: can't be used to edit Word documents or PowerPoint presentations.
- Very minimal by default: it requires a lot of configuration to bring it to feature parity with a modern IDE.
- Requires some knowledge of the terminal: not suited for people who have no technical background.

Why I prefer Vim

- I spend most of my time on my computer editing plaintext code files, so the speed gained from using the Vim keybindings is invaluable.
- As a Linux user and programmer, I spend a lot of time in the terminal, so having a terminal-based editor is very convenient for me.
- Often for internship work, I will have to remotely connect to a server to edit its configurations and the only text editor available is Vim.
- I enjoy the fine-grained control over the program's behaviour that the Vim configuration file affords me.
- I think it looks cooler than other text editors!

Is Vim right for you?

- Vim is right for you if spend a lot of time editing plaintext files, using the terminal, or remotely accessing servers and you don't mind its steep learning curve. If you want fine-grained control over the behaviour of your text editor, and the ability to endlessly extend it, then Vim is a good choice. It helps a lot if you have a technical background, but anyone can learn Vim!
- Vim is not right for you if you rely heavily on graphical editors, e.g. Microsoft Word or PowerPoint, if you don't want to spend time learning the keybindings, or if you want a text editor that is feature-rich out of the box without any customisation. If you rarely use the terminal in your day-to-day life, then a terminal-based text editor likely is not suitable for your workflow.

What alternatives are there to Vim?

Similar text editors to Vim include:

- Emacs
- Vi
- Helix
- Kakoune

More conventional text editors include:

- Notepad
- Atom
- VSCode
- Fully-featured IDEs such as Eclipse or Intellij.

How can I get started with Vim?

- Download & install Vim from `https://www.vim.org/download.php`
- Open a terminal emulator / command prompt on your computer.
- Enter `vim file_name.txt`, substituting `file_name.txt` for the path to the file you want to edit / create, and start editing!
- To get a more in-depth tutorial of how to use Vim, run the command `vimtutor` from your command prompt to start the tutorial.

Summary

By now, I hope that you have gained an understanding of:

- What Vim is and how it works.
- The benefits of Vim.
- The drawbacks of Vim.
- Why I prefer Vim for all my text-editing work.
- Whether or not Vim might be the right text editor for you.
- What alternatives there are to Vim.
- How you can get started with Vim.