GSoC 2018 Neovim Proposal

Improve Vim functionality in Visual Studio Code with `nvim`

Summary

This is a proposal to implement "Vim mode" in Visual Studio Code (VS Code) using `nvim`'s remote UI protocol. VS Code currently has a popular extension called VSCodeVim which emulates some Vim functionality. The existing functionality is limited and in some cases error-prone due to technical challenges with emulating Vim. Changing VSCodeVim to use `nvim` will expose more of Vim's functionality to users and allow future development on VSCodeVim to focus on tuning performance and improving user experience. Using `nvim` will also give VSCodeVim support for existing `.vimrc` files and many native Vim plugins. In short, using `nvim` in VSCodeVim will combine the power of Vim and the convenience of VS Code to improve the everyday lives of developers.

Benefits to Community

VS Code is a popular open source and cross-platform text editor. Since its release in 2015 it has gained over 2.6 million monthly active users. VSCodeVim is an extension for VS Code which provides Vim emulation. With over 1.8 million downloads it is one of VS Code's most-installed extensions. Because VSCodeVim only *emulates* Vim functionality it provides a limited subset of Vim features. The goal for this proposal is to use `nvim`'s remote UI protocol from within VSCodeVim which will:

1. Increase the number of Vim commands and features available in VS Code
2. Decrase the number of emulation bugs affecting users
3. Improve the VSCodeVim codebase for future contributions
4. Make VS Code a more welcoming editor for developers who rely on Vim

In my experience, VS Code provides the best editor features for web developers and Vim provides my favorite text editing capabilities. Combining VS Code with a true Vim experience through `nvim` will create the ultimate development environment for developers like me.

Deliverables

The beginning work of this proposal will take place in the VSCodeNeovim project. (VSCodeNeovim is a fork of VSCodeVim and is owned by this proposal's mentor, Chillee.) Once VSCodeNeovim is working sufficiently well, we will merge our `nvim` integration back into VSCodeVim to expand our user base. After the `nvim` integration is stable, we will phase out the existing Vim emulation. We will wrap

up the end of the proposal with time for bug fixes and enhancements. Ultimately, we will provide a stable, predictable, and complete Vim editing experience to VS Code users.

Completion of this proposal will yield:

TODO: Divide the following list into "Required" and "Optional" tasks. Put "Required" tasks at the beginning of the timeline and "Optional" tasks later in the timeline. (@Chillee I think you should be involved in deciding "Required" versus "Optional".)

1. A working `nvim` integration within VSCodeVim, with as many of the following addressed as possible:
   - Handle operators without storing state in VSCodeVim
     - For example: `2dd` should cause two lines to be deleted without any special logic in VSCodeVim
     - Related: https://github.com/neovim/neovim/issues/6166
   - Cross-file support
     - Actions such as `gd` should keep `nvim` and VSCodeVim in sync
   - Handle split and fold
     - `:sp` and similar commands should work as-expected
     - Fold commands should work as-expected
     - Override `j`, `k`, `gj`, `gk` and related commands
   - Settings
     - Support `.vimrc` files
     - Determine whether VSCodeVim or `nvim` should be the source of truth for settings
   - Autocomplete/Snippets
     - Research best solution
     - Handle any auto-expanded text as if the user typed it
     - Related: https://github.com/lunixbochs/ActualVim/issues/97
   - Automated testing framework
   - Filter commands that VSCodeVim should handle and do not send them to `nvim`
   - Handle file opening and other events
     - Use autocommands to sync VS Code and `nvim` state
   - Performance
     - To improve performance we need diffs from `nvim`
     - Related: https://github.com/neovim/neovim/pull/5269
     - Related: https://github.com/neovim/neovim/pull/7917
   - Key Remapping
     - (@Chillee: I don't understand this task from the PDF you sent me in Slack. Please advise.)
   - Handle read-only files
     - Research best solution
   - Search highlighting
     - Research best solution

    - Multicursor support
        - Once implemented we will need to ensure that actions are sequential
        - Related: https://github.com/neovim/neovim/issues/211
2. Improvements to https://github.com/neovim/node-client where necessary
3. Improvements to https://github.com/neovim/neovim where necessary

Table 1: Proposed Timeline

Week	Dates	Tasks
1	14-18 May	1. Gain familiarity with existing codebase 2. Make VSCodeNeovim documentation beginner-friendly to invite contributions from anyone 3. Fix automated build scripts 4. Make automated tests for existing codebase
2	21-25 May	1. Handle operators without storing state in VSCodeVim 2. Handle splits and folds
3	28 May - 1 Jun	1. Add cross-file support 2. Synchronize file opening and other events between VS Code and `nvim`
4	4-8 Jun	1. Add autocomplete/snippet support 2. Add settings support
5	11-15 Jun	1. Add ability to filter VSCodeVim-only keystrokes 2. Release `nvim` integration as opt-in beta
6	18-22 Jun	1. Fix bugs reported by beta 2. Improve performance with diffs from `nvim`
7	25-29 Jun	1. Fix bugs reported by beta 2. Add key remapping
8	2-6 Jul	1. Fix bugs reported by beta 2. Handle read-only files
9	9-13 Jul	1. Release `nvim` integration as default in VSCodeVim 2. Fix bugs reported by general release
10	16-20 Jul	1. Fix bugs reported by general release 2. Improve search highlighting
11	23-27 Jul	1. Fix bugs reported by general release 2. Add multicursor support

Week	Dates	Tasks
12	30 Jul - 3 Aug	1. Fix bugs reported by general release 2. Finish outstanding tasks 3. Ensure accurate documentation 4. Build roadmap for future development

Related Work

The majority of the coding in this proposal will take place in the VSCodeNeovim and VSCodeVim projects. VSCodeNeoim already contains a starting point for a VS Code/`nvim` integration.

The ActualVim project is an extension for Sublime Text which accomplishes some of what this proposal sets out to do. We will reference it when appropriate.

Neovim and Neovim Node Client are two projects which our proposal depend on. We will debug and contribute to both projects as-necessary to accomplish the goals of this proposal.

Table 2: Related Projects Summary

Project Name	Description
Neovim	Vim fork which provides the `nvim` executable. https://github.com/neovim/neovim
Neovim Node Client	NodeJS bindings for `nvim`. https://github.com/neovim/node-client
VS Code	Graphical text editor. https://github.com/Microsoft/vscode
VSCodeVim	VS Code extension which emulates some Vim functionality. https://github.com/VSCodeVim/Vim
VSCodeNeovim	Fork of VSCodeVim which holds the starting codebase for this proposal. https://github.com/Chillee/VSCodeNeovim
ActualVim	`nvim`-backed Vim extension for Sublime Text. This project provides good implementation ideas. https://github.com/lunixbochs/actualvim

Biographical Information

I am a long-time user of the VSCodeVim extension and have also contributed to it on Github. I feel the need for a fundamental improvement to VSCodeVim and believe that integrating with `nvim` will improve the extension for both contributors and users. Having worked professionally with web technologies for several years, I can navigate the VSCodeVim codebase and write maintainable Typescript with ease. My work in UI/UX has given me skills to tune performance and I have the development experience to make efficient and maintainable design decisions. As a daily VS Code user, I care about the outcome of this proposal beyond deliverable checkmarks - I want Vim-lovers to feel at home in VS Code!