vi-mode plugin

This plugin increase `vi-like` zsh functionality.

To use it, add `vi-mode` to the plugins array in your zshrc file:

```
plugins=(... vi-mode)
```

Settings

- `VI_MODE_RESET_PROMPT_ON_MODE_CHANGE`: controls whether the prompt is redrawn when switching to a different input mode. If this is unset, the mode indicator will not be updated when changing to a different mode. Set it to `true` to enable it. For example:

  ```
  VI_MODE_RESET_PROMPT_ON_MODE_CHANGE=true
  ```

  The default value is unset, unless `vi_mode_prompt_info` is used, in which case it'll automatically be set to `true`.

- `VI_MODE_SET_CURSOR`: controls whether the cursor style is changed when switching to a different input mode. Set it to `true` to enable it (default: unset):

  ```
  VI_MODE_SET_CURSOR=true
  ```

- `MODE_INDICATOR`: controls the string displayed when the shell is in normal mode. See Mode indicators for details.

- `INSERT_MODE_INDICATOR`: controls the string displayed when the shell is in insert mode. See Mode indicators for details.

Mode indicators

Normal mode is indicated with a red `<<<` mark at the right prompt, when it hasn't been defined by theme, *Insert mode* is not displayed by default.

You can change these indicators by setting the `MODE_INDICATOR` (*Normal mode*) and `INSERT_MODE_INDICATORS` (*Insert mode*) variables. This settings support Prompt Expansion sequences. For example:

```
MODE_INDICATOR="%F{white}+%f"
INSERT_MODE_INDICATOR="%F{yellow}+%f"
```

You can also use the `vi_mode_prompt_info` function in your prompt, which will display this mode indicator.

Key bindings

Use `ESC` or `CTRL-[` to enter `Normal mode`.

NOTE: some of these key bindings are set by zsh by default when using a vi-mode keymap.

History

- `ctrl-p` : Previous command in history
- `ctrl-n` : Next command in history
- `/` : Search backward in history
- `n` : Repeat the last `/`

Vim edition

- `vv` : Edit current command line in Vim

NOTE: this used to be bound to `v`. That is now the default (`visual-mode`).

Movement

- `$` : To the end of the line
- `^` : To the first non-blank character of the line
- `0` : To the first character of the line
- `w` : [count] words forward
- `W` : [count] WORDS forward
- `e` : Forward to the end of word [count] inclusive
- `E` : Forward to the end of WORD [count] inclusive
- `b` : [count] words backward
- `B` : [count] WORDS backward
- `t{char}` : Till before [count]'th occurrence of {char} to the right
- `T{char}` : Till before [count]'th occurrence of {char} to the left
- `f{char}` : To [count]'th occurrence of {char} to the right
- `F{char}` : To [count]'th occurrence of {char} to the left
- `;` : Repeat latest f, t, F or T [count] times
- `,` : Repeat latest f, t, F or T in opposite direction

Insertion

- `i` : Insert text before the cursor
- `I` : Insert text before the first character in the line
- `a` : Append text after the cursor
- `A` : Append text at the end of the line
- `o` : Insert new command line below the current one
- `O` : Insert new command line above the current one

Delete and Insert

- `ctrl-h` : While in *Insert mode*: delete character before the cursor
- `ctrl-w` : While in *Insert mode*: delete word before the cursor
- `d{motion}` : Delete text that {motion} moves over

- `dd` : Delete line
- `D` : Delete characters under the cursor until the end of the line
- `c{motion}` : Delete {motion} text and start insert
- `cc` : Delete line and start insert
- `C` : Delete to the end of the line and start insert
- `r{char}` : Replace the character under the cursor with {char}
- `R` : Enter replace mode: Each character replaces existing one
- `x` : Delete `count` characters under and after the cursor
- `X` : Delete `count` characters before the cursor

Known issues

Low `$KEYTIMEOUT`

A low `$KEYTIMEOUT` value (< 15) means that key bindings that need multiple characters, like `vv`, will be very difficult to trigger. `$KEYTIMEOUT` controls the number of milliseconds that must pass before a key press is read and the appropriate key binding is triggered. For multi-character key bindings, the key presses need to happen before the timeout is reached, so on low timeouts the key press happens too slow, and therefore another key binding is triggered.

We recommend either setting `$KEYTIMEOUT` to a higher value, or remapping the key bindings that you want to trigger to a keyboard sequence. For example:

```
bindkey -M vicmd 'V' edit-command-line # this remaps `vv` to `V` (but overrides `visual-mod
```