





# Vim

Adarsh Pyarelal

May 28, 2020

IVILab Summer Bootcamp 2020

# Motivation

	Regular tools	Power tools
Saws		
Text editors		

# Motivation

- A capable text editor might mean different things in different contexts. E.g.
  - Emacs for LISP devs
  - IntelliJ for Java and Scala devs
- Reasons for learning vi:
  - Available everywhere, no need for GUI (I was forced to learn Vim to work with UA HPC)
  - Lightweight compared to IDEs
  - The vi philosophy of modal text editing transcends editors - vi keybindings are available for Emacs (evil-mode) and IDEs (from experience, IntelliJ IDEs and Jupyter both support vim bindings), so an investment in learning vi will pay off no matter what.
  - You will become *way* faster at writing and editing code (this includes LaTeX) - at least 10X as fast, if not more.
- Initial learning curve is steep, but pays off handsomely in the long run

# Outline

- Motivation
- Modal editing
- Vim grammar
- Navigation
- Editing & working with text objects
- Customizing Vim (.vimrc)
- Plugins

For the next three weeks,  
you will use no other  
editor but Vim\*

# Modal editing

Vim has different 'modes':

- Normal (you'll spend most of your time in this one)
- Visual
- Insert
- Replace
- Command



# Vim grammar

- Instead of memorizing individual commands, learn the 'grammar' of vim, i.e. how to 'speak' to it.
- Vim 'sentences' generally follow the pattern below:

Editing: [N] [Verb] [Modifier] [Motion/Character/Object]

- N : A positive integer.
- The square brackets are there because:
  - Not all sentences will have all the components, and
  - Some values for the components are incompatible with other components. E.g. Some verbs ignore multipliers.

Reference: <https://benmccormick.org/2014/07/02/learning-vim-in-2014-vim-as-language>

# Navigation I [N] [Verb] [Modifier] [Motion/Character/Object]

- **Motion** (these motions ignore multipliers)
  - **0**: Go to the beginning of the line
  - **\$**: Go to the end of the line
  - **gg**: Go to the top of the document
  - **G**: Go to the end of the document
- **[Multiplier] Motion**
  - **h/j/k/l**: Go one character left/down/up/right
  - **w**: Go to the beginning of the next word
  - **e/ge**: Go to the next/previous character that ends a word
  - **b**: Go to the previous character that begins a word
  - **W, E/gE, B**: Same as their lowercase counterparts, but only treats whitespace as word separators (the others treat '(', '-', ')', etc. as word separators)
  - Example:
    - **3h**: move three characters to the left



# Navigation II [N] [Verb] [Modifier] [Motion/Character/Object]

- [Multiplier] Modifier Character

- The modifiers that can be used with this type of navigation pattern are:
  - **f/F** (find) - Go to the first occurrence to the right/left of the given character on the current line to the
  - **t/T** (till) - Move the cursor just left of the first occurrence to the right/left of the given character
- Examples:
  - **3fx** - Move cursor to the 3<sup>rd</sup> occurrence of the letter 'x' to the right on the current line.

- [Line number] Motion

- **Ngg/NG** - all these commands move the cursor to the N<sup>th</sup> line in the file (as does the command **:N**)

# Editing I    [N] [Verb] [Modifier] [Motion/Character/Object]

- **Verb** (these ones ignore multipliers)
  - Insert
    - **i**: Enter insert mode one character to the left.
    - **I**: Enter insert mode at the beginning of the current line (equivalent to **0i**)
  - Append
    - **a**: Enter insert mode one character to the right.
    - **A**: Enter insert mode at the end of the line (equivalent to **\$a**)
  - Open
    - **o/O**: Add a blank line below/above the current one and enter insert mode
  - **R**: enter replace mode
  - **D**: Delete everything to the right of the cursor on the current line (equivalent to **d\$**)
  - **C**: Delete everything to the right of the cursor on the current line and enter insert mode (equivalent to **c\$**)
- **[Multiplier] Verb**
  - **p/P**: paste text before/after the cursor
  - **x/X**: delete the current character/the character just to the left
  - **r**: Enter replace mode for a single character (multiple if a multiplier is given)
  - **u**: Undo last action
  - **.**: repeat last action

## Interlude: text objects

- IMHO the most ‘bang-for-the-buck’ thing to learn in Vim.
- There are a few built-in text objects:
  - Word (w)
  - Sentence (s)
  - Paragraph (p)
  - Selection inside/around parentheses, braces, quotes, etc.
  - Lines are treated specially - to apply an action to a line, typically you would just repeat the verb (e.g. yy, cc, dd)
- Additional ones are available via plug-ins (e.g. LaTeX environments)

# Editing II [N] [Verb] [Modifier] [Motion/Character/Object]

- [N] Verb [Modifier] Motion/Object
  - The verbs that can be used with this pattern are:
    - **y**: (yank) - copy text
    - **d**: (delete) - cut text
    - **c**: (change) - cut text and enter insert mode.
    - **gq**: Format text
    - **v**: visual selection
  - A **Modifier** can be one of:
    - A positive integer **N** (for repeating)
    - **t/T/f/F** (see slide 9)
    - **i/a** - For use with text objects. You can think of them as 'in' and 'a' (or 'around')
  - Examples:
    - **2yw**: Yank all the text till the beginning of the next word, twice - i.e., yank the next two words.
    - **2y3w**: Yank three words twice - i.e. yank the next six words.
    - **2ctf**: Cut all text between the current cursor position and the second occurrence of the character 'f' to the right of the cursor.
    - **ci"**: Delete all text inside the double quotes (assuming the cursor is between the quotes) and enter insert mode
    - **ca(**: Delete all text inside the parentheses as well as the parentheses themselves, and enter insert mode

# Splits

- Vertical split - :vsp
- Horizontal split - :sp
- Split navigation:  
<Ctrl>-w + h/j/k/l
- Making splits equally sized:  
<Ctrl>-w=

```
s/Mission.h s/Mission.cpp s/Utils.h buffers
1 #include "MissionSpec.h"
2 #include "AgentHost.h"
3 #include <boost/filesystem.hpp>
4 #include <string>
5 #include <unordered_map>
6
7 namespace tomcat {
8
9     class LocalAgent; // Forward declaration to deal with circular dependency
10
11     class TomcatMissionException : public std::exception {
12     public:
13         enum ErrorCode {
14             CONNECTION_NOT_ESTABLISHED,
15             TOMCAT_ENV_VAR_NOT_SET,
16             ERROR_STARTING_MISSION,
17             WORLD_DIR_NOT_FOUND
18         };
19
20         TomcatMissionException(const std::string& message, ErrorCode error_code)
21             : message(message), error_code(error_code) {}
22
23         ~TomcatMissionException() throw() {}
24         ErrorCode get_error_code() const { return this->error_code; }
25         std::string get_message() const { return this->message; }
26         const char* what() const throw() { return this->message.c_str(); }
27     };
28
29     NORMAL master src/Mission.h cpp utf-8 4% 8/172 1
30     1 #pragma once
31     2
32     3 #include <string>
33     4
34     5 namespace tomcat {
35     6
36     7     template <class AssociativeContainer, class Value>
37     8     bool in(AssociativeContainer container, Value value) {
38     9         return container.count(value) > 0;
39     10     }
39
40     11     template <class AssociativeContainer, class Key, class Value>
41     12     Value get(AssociativeContainer container, Key key, Value default_value) {
42     13         return in(container, key) ? container[key] : default_value;
43     14     }
44
45     15 } // namespace tomcat
46
47 src/Utils.h cpp utf-8 11% 2/17 1
48     1 #include "Mission.h"
49     2 #include "FileHandler.h"
50     3 #include "LocalAgent.h"
51     4 #include "utils.h"
52     5 #include <boost/date_time/posix_time/posix_time.hpp>
53     6 #include <boost/filesystem.hpp>
54     7 #include <boost/uuid/uuid.hpp>
55     8 #include <boost/uuid/uuid_io.hpp>
56     9 #include <fmt/format.h>
57    10 #include <nlohmann/json.hpp>
58    11 #include <sstream>
59
60    12 using namespace malmo;
61    13 using namespace std;
62    14 using namespace boost;
63    15 using namespace boost::posix_time;
64    16 using namespace boost::chrono;
65    17 using namespace std::this_thread;
66    18 using namespace std::chrono;
67    19 using namespace std::chrono;
68    20 namespace pt = boost::posix_time;
69    21 namespace fs = boost::filesystem;
70
71    22 namespace tomcat {
72    23
73    24     Mission::Mission(string mission_id_or_path,
74    25                     unsigned int time_limit_in_seconds,
75    26                     unsigned int self_report_prompt_time_in_seconds,
76    27                     unsigned int level_of_difficulty,
77    28                     int port_number,
79    29                     bool record_observations,
80    30                     bool record_commands,
81    31                     bool record_rewards,
82    32                     bool multiplayer,
83    33                     string uuid) {
84    34
85    35         this->mission_id_or_path = mission_id_or_path;
86    36         this->time_limit_in_seconds = time_limit_in_seconds;
87    37         this->self_report_prompt_time_in_seconds =
88    38             self_report_prompt_time_in_seconds;
89    39         this->level_of_difficulty = level_of_difficulty;
90    40         this->port_number = port_number;
91    41         this->record_observations = record_observations;
92    42         this->record_commands = record_commands;
93    43         this->record_rewards = record_rewards;
94    44         this->multiplayer = multiplayer;
95    45         if (uuid.compare("") == 0) {
96    46             boost::uuids::uuid u;
97    47             this->uuid = boost::uuids::to_string(u);
98    48         }
99    49         else {
100    50             this->uuid = uuid;
101    51         }
102    52     }
103
104    53     void Mission::add_listener(shared_ptr<LocalAgent> tomcat_agent) {
105    54
106    55         src/Mission.cpp cpp utf-8 0% 1/366 1
107         1 #include "Mission.h"
108         2 #include "FileHandler.h"
109         3 #include "LocalAgent.h"
110         4 #include "utils.h"
111         5 #include <boost/date_time/posix_time/posix_time.hpp>
112         6 #include <boost/filesystem.hpp>
113         7 #include <boost/uuid/uuid.hpp>
114         8 #include <boost/uuid/uuid_io.hpp>
115         9 #include <fmt/format.h>
116        10 #include <nlohmann/json.hpp>
117        11 #include <sstream>
118
119        12 using namespace malmo;
120        13 using namespace std;
121        14 using namespace boost;
122        15 using namespace boost::posix_time;
123        16 using namespace boost::chrono;
124        17 using namespace std::this_thread;
125        18 using namespace std::chrono;
126        19 using namespace std::chrono;
127        20 namespace pt = boost::posix_time;
128        21 namespace fs = boost::filesystem;
129
130        22 namespace tomcat {
131        23
132        24     Mission::Mission(string mission_id_or_path,
133        25                     unsigned int time_limit_in_seconds,
134        26                     unsigned int self_report_prompt_time_in_seconds,
135        27                     unsigned int level_of_difficulty,
136        28                     int port_number,
137        29                     bool record_observations,
138        30                     bool record_commands,
139        31                     bool record_rewards,
140        32                     bool multiplayer,
141        33                     string uuid) {
142        34
143        35         this->mission_id_or_path = mission_id_or_path;
144        36         this->time_limit_in_seconds = time_limit_in_seconds;
145        37         this->self_report_prompt_time_in_seconds =
146        38             self_report_prompt_time_in_seconds;
147        39         this->level_of_difficulty = level_of_difficulty;
148        40         this->port_number = port_number;
149        41         this->record_observations = record_observations;
150        42         this->record_commands = record_commands;
151        43         this->record_rewards = record_rewards;
152        44         this->multiplayer = multiplayer;
153        45         if (uuid.compare("") == 0) {
154        46             boost::uuids::uuid u;
155        47             this->uuid = boost::uuids::to_string(u);
156        48         }
157        49         else {
158        50             this->uuid = uuid;
159        51         }
160        52     }
161
162        53     void Mission::add_listener(shared_ptr<LocalAgent> tomcat_agent) {
163        54
164        55         src/Mission.cpp cpp utf-8 0% 1/366 1
```

## .vimrc

...or, why does my vim window not look like the screenshots here???

The file ~/.vimrc holds the customization settings for vim. Here's mine:

<https://github.com/adarshp/dotfiles/blob/master/.vimrc>

Feel free to copy over the whole thing into your .vimrc or just the parts you like.

Eventually, you should probably know what the different lines in the vimrc mean :)

# Plugins

- You'll probably want a plugin manager for Vim - plugins can provide powerful functionality.
- I use <https://www.github.com/junegunn/vim-plug> - the first few lines of my ~/.vimrc (see slide 15) - automatically install vim-plug if it's not already installed

# Things you probably will want to know about later

- Screen navigation (<Ctrl>-f, <Ctrl>-b)
- Tabs (:tabe, gt)
- Macros
- Registers
- ctags