vi / vim graphical cheat sheet

~  toggle case
\ ~  goto mark
Esc  normal mode

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>

Q  ex mode
q  record macro
w  next word
W  next WORD
E  end word
e  end word
R  replace mode
r  replace char
T  'till
'till
Y  yank line
y  yank
U  undo line
u  undo
I  insert at bol
i  insert mode
O  open above
O  open below
P  paste before
p  paste after
{  begin parag.
}  begin parag.
}  end parag.

A  append at eol
a  append
S  subst line
s  subst char
D  delete to eol
d  delete
F  "back" find ch
f  find char
G  eol/goto ln
g, extra cmds
H  screen top
h  ←
j  join lines
J  help
K  help
L  screen bottom
l  →

Z  quit
\Z  extra cmds
X  delete space
C  change to eol
V  visual mode
B  prev V
b  prev
N  next (find)
n, next (find)
M  set mark
m  set mark
<  unindent
>  indent
?  find
/

motion

moves the cursor, or defines the range for an operator

direct action command, if red, it enters insert mode

operator

requires a motion afterwards, operates between cursor & destination

special functions, requires extra input

commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line, mk = mark, yank = copy

words: \uuxx{foo, bar, baz};
WORDS: \uuxx{foo, bar, baz};

Main command line commands ('ex'):
:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:s/s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :new (new file in vim),

Other important commands:
CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)

Visual mode:
Move around and type operator to act on selected region (vim only)

Notes:
(1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*')
(e.g.: "ay$ to copy rest of line to reg 'a')
(2) type in a number before any action to repeat it that number of times
(e.g.: 2p, 2uw, 5i, d4j)
(3) duplicate operator to act on current line
(dd = delete line, >> = indent line)
(4) ZZ to save & quit, ZQ to quit w/o saving
(5) zt: scroll cursor to top,
zb: bottom, zz: center
(6) gg: top of file (vim only),
gf: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Visual Studio
Basics:

1. h, j, k, l are vi/vim cursor keys – use them as they are much closer than regular cursor keys!

2. Use i to enter insert mode, cursor turns from a block into a vertical line, and you can type in text. Use Esc to return to normal mode.

3. Use x to delete the current character, or X to delete the one to the left

4. Use A to go insert text at the end of the line (wherever you are in the line!)

(Note: insert mode is actually very similar to a regular editor, you can use cursor/navigation keys, backspace, delete...)

Extras:

1. u to undo the last action – traditional vi has a single level, while vim supports unlimited undo (CTRL-R to redo)

2. 0 jumps directly to the beginning of the line, $ to the end, and ^ to the first non-blank

3. Use w, b, e to move along 'words'. A 'word' is a sequence of all alphanumeric or punctuation signs: quux([foo, bar, baz])

4. Use W, B, E to move along WORDs. A 'WORD' is a sequence of any non-blank characters: quux(foo, bar, baz)

5. Use R to enter insert mode with an overstrike cursor, which types over existing characters.

: w and press enter to save, : q and enter to quit.

For the rest of the tutorial & a full cheat sheet, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Visual Studio.
**Basics:**

- \f, followed by another key, moves the cursor to the next instance of that character on the current line, \F does the same backwards. 
- \t and \T do the same, but they stop right before the character.
- \d (delete), followed by any motion deletes the text between the cursor and that motion's destination \d w, \d f –...). 
- \c (change) does the same, but leaves you in insert mode.
- Some motions, such as \j and \k, are linewise – deletion includes the full start/end lines.
- \s repeats the last editing action: text input, delete or change, etc... motion is recalculated at the new place.

**Extras:**

Prepend a count to any command/motion to repeat it that number of times:

- \d 2 \w to delete up to the second word.
- \d 2 \l , to delete up to but not including the second comma.
- \d 2 \l repeats the text after you press (Esc) to finish the input session.

Repeat operator (\c \c or \d \d) to operate on the current line.

Only in vim, \V enters visual mode. Move around with motions, the text will be highlighted. Press an operator to operate on that selection.

\V enters visual-lines mode – like \V, but selecting whole lines.

CTRL - \V selects rectangular blocks.

For the rest of the tutorial & a full cheat sheet, go to [www.viemu.com](http://www.viemu.com) - home of ViEmu, vi/vim emulation for Microsoft Visual Studio
vi/vim lesson 3 - yank & paste

Basics
Use \texttt{Y} followed by any motion to ‘yank’ (copy).
Use \texttt{p} to paste after (if charwise, to the right, if linewise, below).
Use \texttt{P} to paste before.
\texttt{yy} copies the current line.
\texttt{Y} also works in visual mode.
Text deleted with \texttt{d}, \texttt{c}, \texttt{x} ... is also copied!

Extras
\texttt{"} and an \texttt{a} - \texttt{z} character before any yank/delete/paste command chooses a register.
An \texttt{A}-\texttt{Z} register before yank/delete means “append-copy”.
\texttt{"*} or \texttt{"±} select the system clipboard.
\texttt{O} enters insert mode in a new empty line below the current one.
\texttt{O} does the same above the current line.

For the rest of the tutorial & a full cheat sheet, go to \texttt{www.viemu.com} - home of ViEmu, vi/vim emulation for Microsoft Visual Studio
### vi/vim lesson 4 - searching

<table>
<thead>
<tr>
<th>Esc</th>
<th>normal mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>prev ident</td>
</tr>
<tr>
<td>$</td>
<td>eol</td>
</tr>
<tr>
<td>^</td>
<td>&quot;soft&quot; bol</td>
</tr>
<tr>
<td>*</td>
<td>next ident</td>
</tr>
<tr>
<td>0</td>
<td>&quot;hard&quot; bol</td>
</tr>
<tr>
<td>W</td>
<td>next word</td>
</tr>
<tr>
<td>W</td>
<td>next word</td>
</tr>
<tr>
<td>E</td>
<td>end word</td>
</tr>
<tr>
<td>E</td>
<td>end word</td>
</tr>
<tr>
<td>R</td>
<td>replace mode</td>
</tr>
<tr>
<td>T</td>
<td>‘till</td>
</tr>
<tr>
<td>T</td>
<td>‘till</td>
</tr>
<tr>
<td>V</td>
<td>yank</td>
</tr>
<tr>
<td>u</td>
<td>undo</td>
</tr>
<tr>
<td>i</td>
<td>insert mode</td>
</tr>
<tr>
<td>o</td>
<td>open above</td>
</tr>
<tr>
<td>p</td>
<td>paste before</td>
</tr>
<tr>
<td>p</td>
<td>paste after</td>
</tr>
<tr>
<td>A</td>
<td>append at eol</td>
</tr>
<tr>
<td>d</td>
<td>delete</td>
</tr>
<tr>
<td>f</td>
<td>find char</td>
</tr>
<tr>
<td>h</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>prev (find)</td>
</tr>
<tr>
<td>n</td>
<td>next (find)</td>
</tr>
<tr>
<td>X</td>
<td>backspace</td>
</tr>
<tr>
<td>X</td>
<td>delete char</td>
</tr>
<tr>
<td>C</td>
<td>change</td>
</tr>
<tr>
<td>V</td>
<td>visual lines</td>
</tr>
<tr>
<td>V</td>
<td>visual mode</td>
</tr>
<tr>
<td>B</td>
<td>prev WORD</td>
</tr>
<tr>
<td>b</td>
<td>prev word</td>
</tr>
<tr>
<td>?</td>
<td>find (rev.)</td>
</tr>
</tbody>
</table>

### Basics:

- Enter is the basic search motion – type the text you are searching for after the slash, and then press return. Being a motion, you can use this after an operator, or in visual mode.
- ? does the same, backwards.
- n repeats the last search in the same direction, N repeats it in the reverse direction

Be careful, because the search target is interpreted as a regular expression: a*b means one or more ‘a’s followed by a ‘b’, ^abc means ‘abc’ at the beginning of a line, [0-9] looks for the next digit, etc...  

### Extras:

- The following very useful motions work only in vim:
  - * searches forward for the next instance of the identifier under the cursor.
  - # does the same backwards.

For the rest of the tutorial & a full cheat sheet, go to [www.viemu.com](http://www.viemu.com) - home of ViEmu, vi/vim emulation for Microsoft Visual Studio
vi/vim lesson 5 - marks & macros

Marks:
Use \text{\textbf{m}} followed by a \text{a} - \text{z} character to set a mark.
Use \text{\textbf{g}} followed by a character to go to that mark.
Use \text{\textbf{1}} and a character to go to the first non-blank in that line.
\text{A} - \text{Z} marks are global, \text{a} - \text{z} per-buffer.
\text{\textbf{1}} refers to the position of the last modification.

Macros:
Use \text{\textbf{q}} followed by a \text{a} - \text{z} character to start recording.
Use \text{\textbf{q}} afterwards to stop recording.
\text{\textbf{1}} followed by a character replays that macro.
\text{\textbf{g}} \text{\textbf{g}} to repeat the last macro played.

For the rest of the tutorial & a full cheat sheet, go to \url{www.viemu.com} - home of ViEmu, vi/vim emulation for Microsoft Visual Studio
vi/vim lesson 6 – various motions

% jumps between matching pairs of ‘(‘ ‘)’, ‘[‘, ‘]’, etc...

H M L jump directly to the top/middle/bottom of the screen.

G jumps to the end of the file, or to the line # typed before it.

– / + jump to the previous/next line.

K, not technically a motion, jumps to the help for the word under the cursor: vim help, man page under unix, etc...

C and D jump to the beginning/end of the current sentence.

{ and } jump to the previous/next empty line.

I I jumps to the previous ‘{‘ in column 0.

I I jumps to the next ‘{‘ in column 0.

For the rest of the tutorial & a full cheat sheet, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Visual Studio
Basics:

- \textbf{J} joins the current line with the next one, or all the lines in the current visual selection.
- \textbf{r} followed by any character replaces the current character with that one.
- \textbf{C} is shorthand for \textbf{c s}, changes to end of line.
- \textbf{D} is shorthand for \textbf{d s}, deletes to end of line.
- \textbf{Y} is shorthand for \textbf{Y s}, yanks the whole line.
- \textbf{s} deletes the character under the cursor and enters insert mode.
- \textbf{S} clears the current line and enters insert mode.

Extras:

- \textbf{>} and a motion to indent one or more lines.
- \textbf{<} and a motion to unindent.
- \textbf{=} and a motion to reformat a range of text.

All of them work in visual mode, or can be repeated (\textbf{> >}, etc...) to operate on the current line.

Now go grab the full cheat sheet and learn the rest. Start with \textbf{1 a} and \textbf{1 i}. Piece of cake!

For the rest of the tutorial & a full cheat sheet, go to \url{www.viemu.com} home of ViEmu, vi/vim emulation for Microsoft Visual Studio.