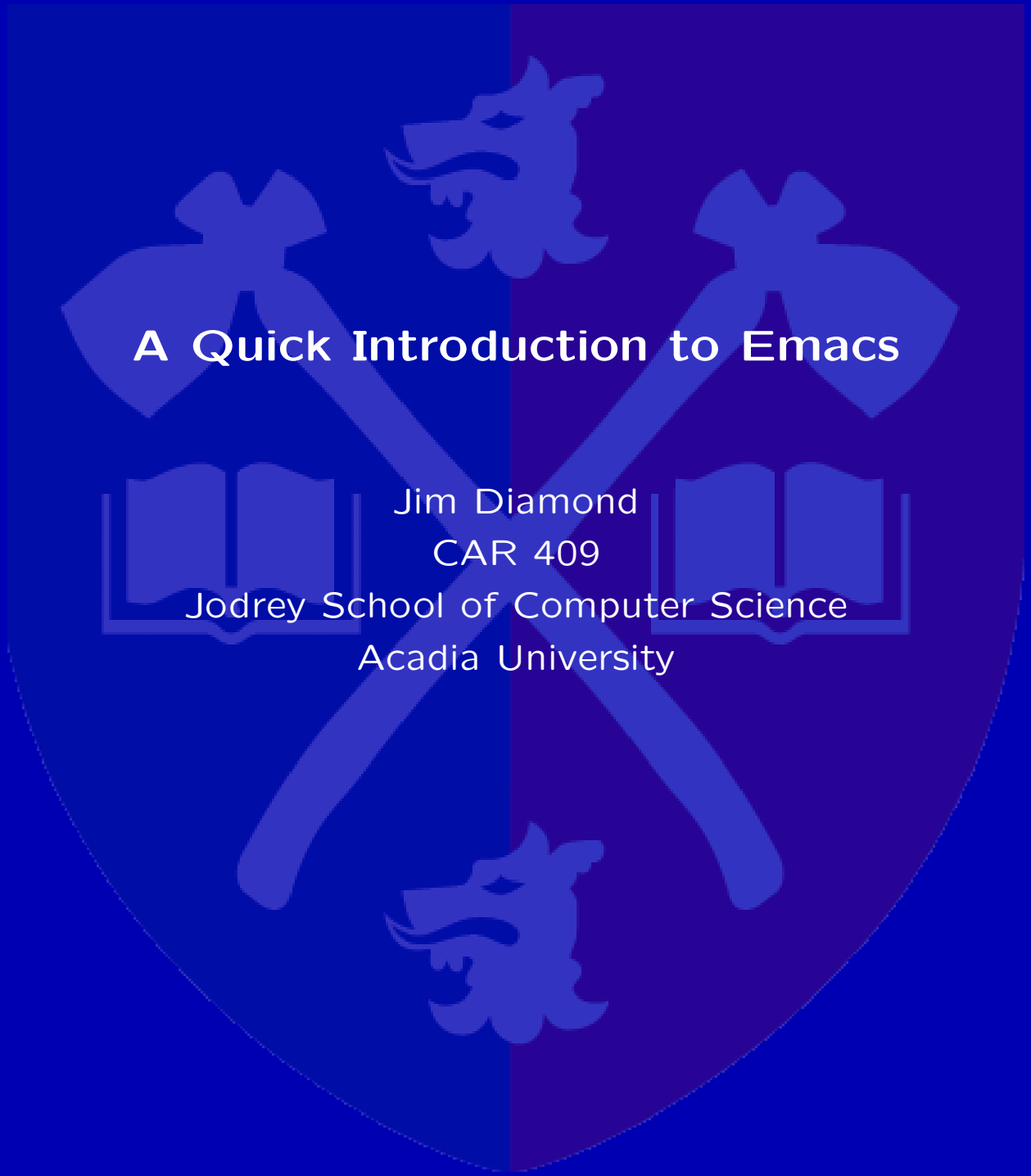


A Quick Introduction to Emacs

Jim Diamond
CAR 409
Jodrey School of Computer Science
Acadia University

Overview

- Availability
- History
- Basic Concepts
- Psychoanalysis
- Built-In Help
- My Video Tutorial
- Keyboard Layout, As God Intended
- Text File Support
- Programming Support
- Extending Emacs
- Customization
- Regular Expressions

Availability

- The editor is freely available (for Linux, windows, macOS) at
  https://www.gnu.org/software/emacs/
- The source code is available for free at
  ftp://prep.ai.mit.edu/pub/gnu/emacs
- Most (all?) distributions of Linux come with emacs (pre-compiled)
- Pre-compiled versions for various other OS's are available can be found

Basic Concepts

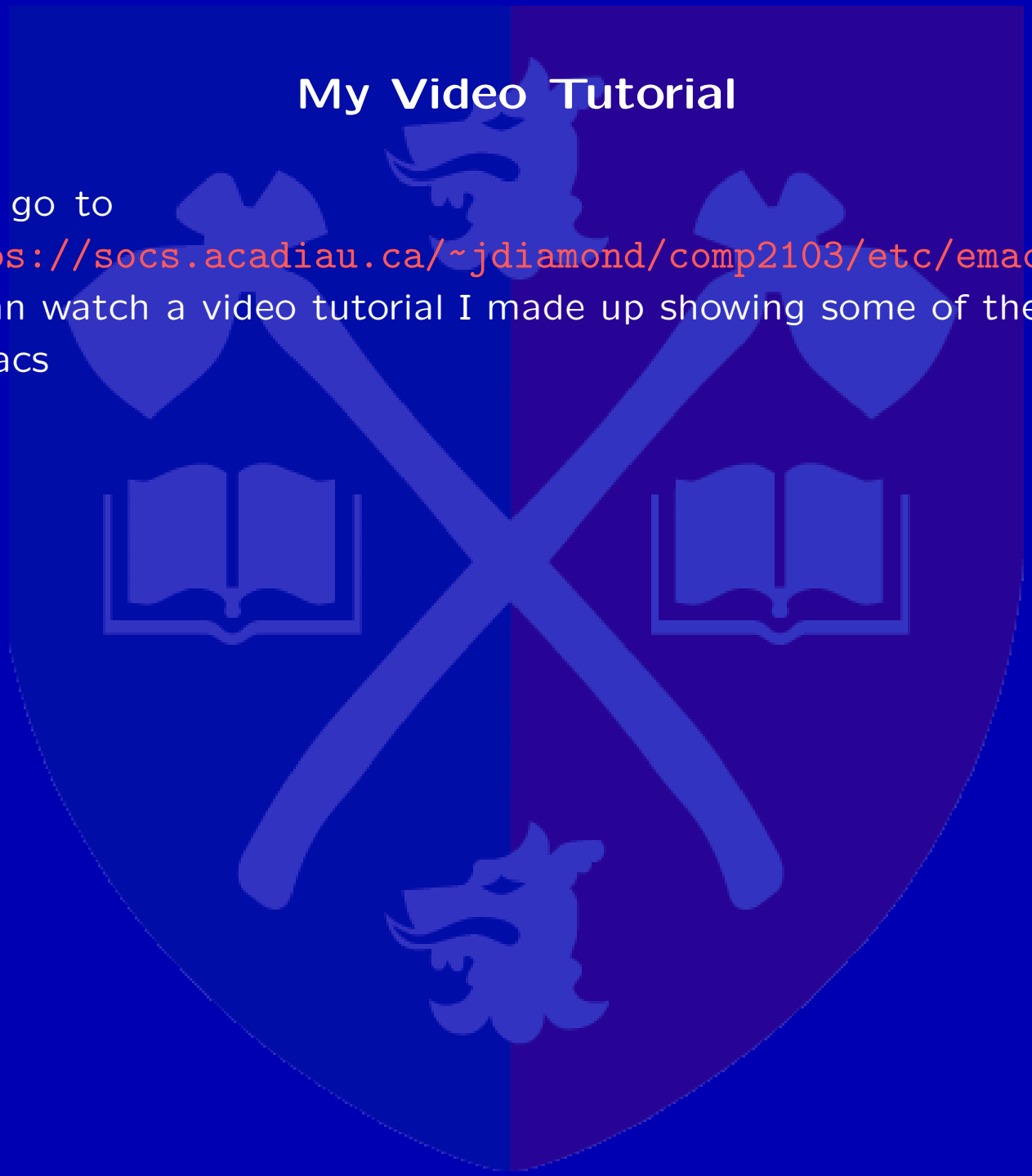
- Can edit multiple files concurrently
- You can open multiple windows (emacs calls them *frames* for historical reasons)
  - each window can be split into one or more "panes" (emacs calls them *windows* for historical reasons)
- The same file can appear in multiple windows or frames
- No "input mode" vs. "command mode" dichotomy
  - commands introduced with control- or meta- (alt) chars
- *Major modes* provide functionality tailored for particular tasks: C programming, Java programming, plain text entry, ...
- *Minor modes* provide additional functionality which might be useful in various major modes: abbrev expansion, on-the-fly spell checking, ...
- Multi-level undo (more complex/capable than most other tools)
- An initialization file is read when emacs starts up (formerly it was `~/.emacs`, now one should use `~/.emacs.d/init.el`)

Psychoanalysis and Other Diversions

- doctor (psychoanalysis) (type `M-x doctor <return>` to start it up)
- dunnet (adventure-like game)
- go (strategy game... use `gomoku`)
- hanoi (Towers of Hanoi)
- life (cellular automata simulation)
- tetris (arcade game)
- ... and others.

My Video Tutorial

- If you go to
  https://socs.acadiau.ca/~jdiamond/comp2103/etc/emacs-demo.mp4
  you can watch a video tutorial I made up showing some of the features of emacs

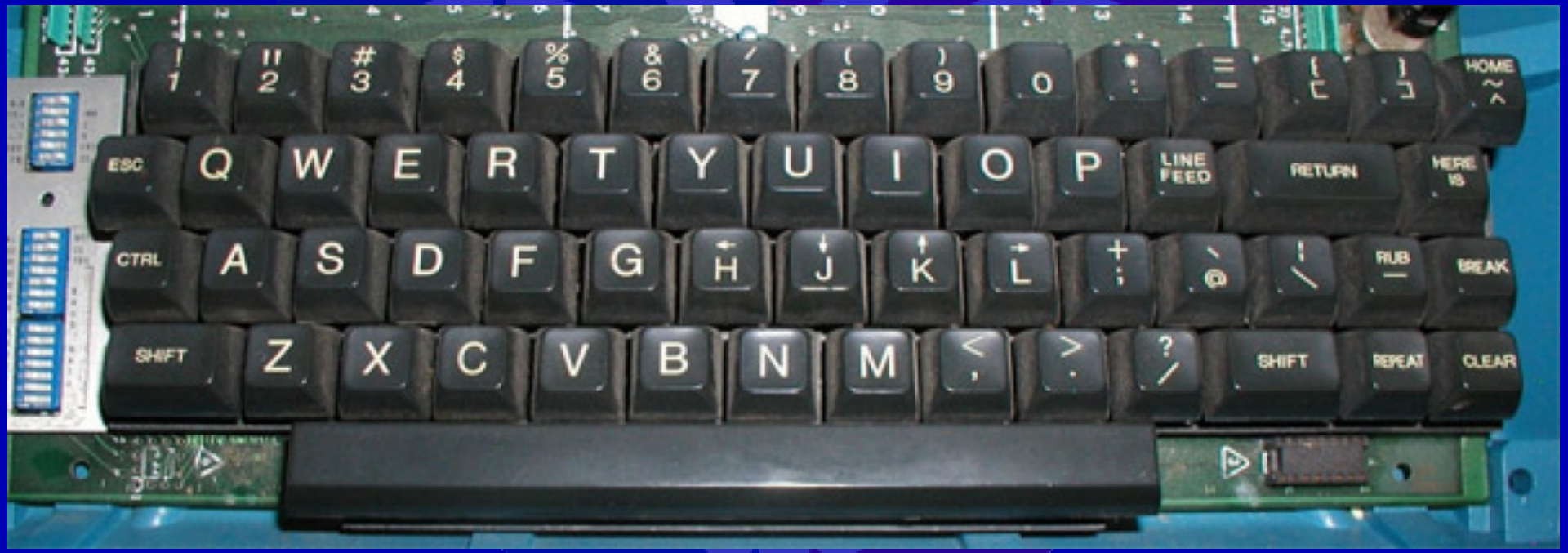


Built-In Help

- Extensive built-in help is available (`C-h ...`)
  - Emacs comes with a build-in tutorial (`C-h C-h t`)
- *Apropos* (look for descriptions matching a given keyword)
- Describe key (`C-h k`)
  - in case you wonder what some key or key combination does when you type it
- Describe variable (`C-h v`)
- Describe function (`C-h f`)
- "Info" system (`C-h C-h i`)
- Emacs comes with a six page reference card (`refcard.pdf`)

Keyboard Layout, As God Intended

- Emacs does use a lot of Ctrl-keys
- This is because the Ctrl key was easy to get at before IBM displayed exceptionally bad taste with the PC keyboard layout
- Here is how God intended things to look:



  - (ok, the Alt key is missing here, but the Ctrl key is in the right place)

Text File Support

- Spell-checking and related tools
  - you can check spelling either on command or continuously
  - automagic transposition correction is available
  - dictionary look-up
- Abbreviation expansion:
  - automagically replace something short you typed with something else (probably much longer)
- Operations for textual units
  - words, sentences, paragraphs, pages
- Filling, justification

Programming Support

- Various modes to support editing for a variety of languages and packages (Ada, awk, C, C++, Fortran, Java, lisp, make, Pascal, PS, Python, shell scripts, scheme, sql, . . .)
  - these modes "understand" aspects about the given language and their syntactic constructs to make the entry of syntactically correct programs easier
- For example:
  - help with matching parens and braces
    - *e.g.*, show matching paren/brace
    - *e.g.*, skip forward or backward over blocks or parenthesized expressions
  - Abbreviation expansion — good for long identifier names
    - type the first few letters of a token and press `Alt-/`
  - Tags file — automagically find a function in some file in current project

Extending Emacs

- You can "easily" create your own modes
- You can add functionality to other modes
- You can emulate other editors (*e.g.*, two major modes emulate vi)
- Eamcs' *extension language* is called `emacs lisp` (`e-lisp`)
  - it is similar to "common lisp" that you might learn in an AI course

Extending Emacs – 2

- Example: here is an emacs lisp function which can be called to jump to a given percentage down the buffer

```
(defun
   goto-percent (percent)
   "Move point to a given percent in the file."
   (interactive "NGoto percent: ")
   (if (= percent 0)
      (goto-char (point-min))
      (goto-char (+ (/ (* (buffer-size) percent) 100) 1))
   )
)

(global-set-key   "\ep"    'goto-percent)  ; a.k.a. M-p
```

- For technical/historical reasons, when you see emacs referring to the "Meta" key, as in "Meta-p" or "M-p", you should think "Alt"

Customization

- Use e-lisp, or ...
- `M-x customize`
  - allows you to customize emacs without knowing e-lisp
  - customizations can be used for current session and/or saved for future sessions
  - customizations are saved in `~/.emacs.d/init.el`

Regular Expressions

- Not just for emacs (vi and many other programs use them)
- "reg exps": chars represent one of
  - themselves, patterns, or operators
- *e.g.*, "`.`" is a pattern that matches any single char except newline
- *e.g.*, "`*`" is an operator matching 0 or more copies of the preceding pattern
- "`a.b*c`" is a pattern representing any string composed of
  - exactly one "`a`"
  - any single non-newline character
  - zero or more "`b`"s
  - exactly one "`c`"
- Here is a reg exp that can be used to search through a program for a line that assigns `salary` a value using `rate`:

  ```
  salary *=.*rate
  ```

  (this could have false matches, e.g. if you have `salary = remunerate`)