Clipit: An Intelligent Clipboard Manager for Programmers

CSC 510 - Apr 7 Presentation

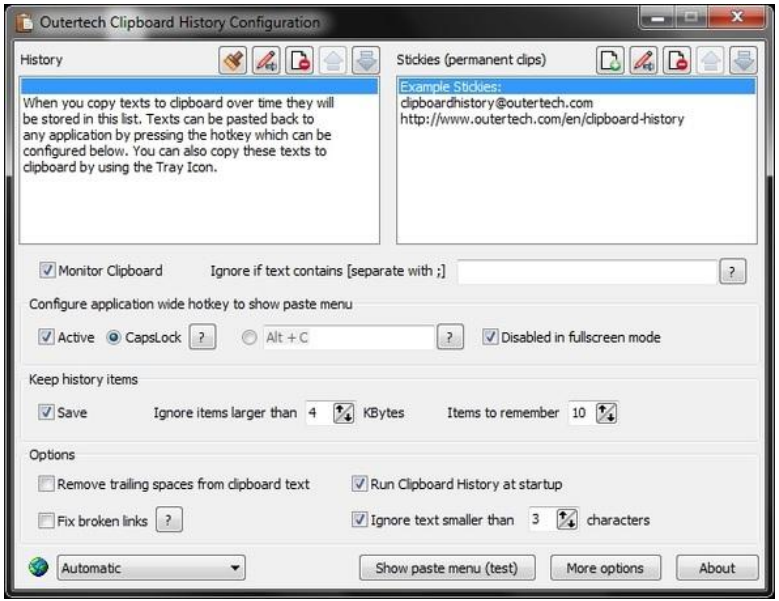
Group G

Research Questions

- Programmers copy and paste… a lot
  - 16 times per hour on average
- Problems with copy and paste
  - Introduces *code clones*
  - Encourages *window switching*
  - Only holds a *single item*
  - People make *mistakes*
- Our Feb 1 solution
  - Clipboard managers

Clipboard Managers

- Existing clipboard managers
  - System-wide
  - Not focused on *Programmers* tasks
  - Out of date (or poorly designed)
- Project goal
  - Create a base clipboard manager
    - Designed for programmers
  - Design/develop 3 versions from this base



Our Approach

- Choosing a programming environment
  - Eclipse, Sublime, Atom... oh my!
- Atom was the clear choice
  - Completely **open-source**
  - Active package development community
  - Package development support
  - Multi-platform support
- Clipboard manager packages already exist for Atom

Base Version, aka Clipit

- Strong existing clipboard manager package
  - clipboard-history
  - Allows access to clipboard history through visual interface
  - Provides search and clear history functions
  - Large copies are displayed in snippets (preview on hover)
- Packages in Atom
  - Written in Coffeescript
  - Open-source

```coffee
class ClipboardHistoryView extends SelectListView

  editor: null
  forceClear: false

  # Public methods
  ########################################
  initialize: (@history, @editorView) ->
    super
    @addClass('overlay clipboard-history from-bottom')
    {@editor} = @editorView
    @_handleEvents()

  copy: ->
    @forceClear = false
    if @editorView.active
      selectedText = @editor.getSelectedText()
      if selectedText.length > 0
        @_add selectedText

  paste: ->
    # Check OS clipboard
    clipboardItem = atom.clipboard.read()
    if clipboardItem.length > 0 and not @forceClear
      exists = false
      for item in @history
        if item.text is clipboardItem
          exists = true

      if not exists
        @_add clipboardItem

    # Attach to view
    if @history.length > 0
      @setMaxItems(15)
```

How can we make this BETTER

- Great basic starter project
  - Lots of room for improvement
- 3 new versions inspired by Feb 1 research
  - Clipit-order
  - Clipit-panel
  - Clipit-cmd

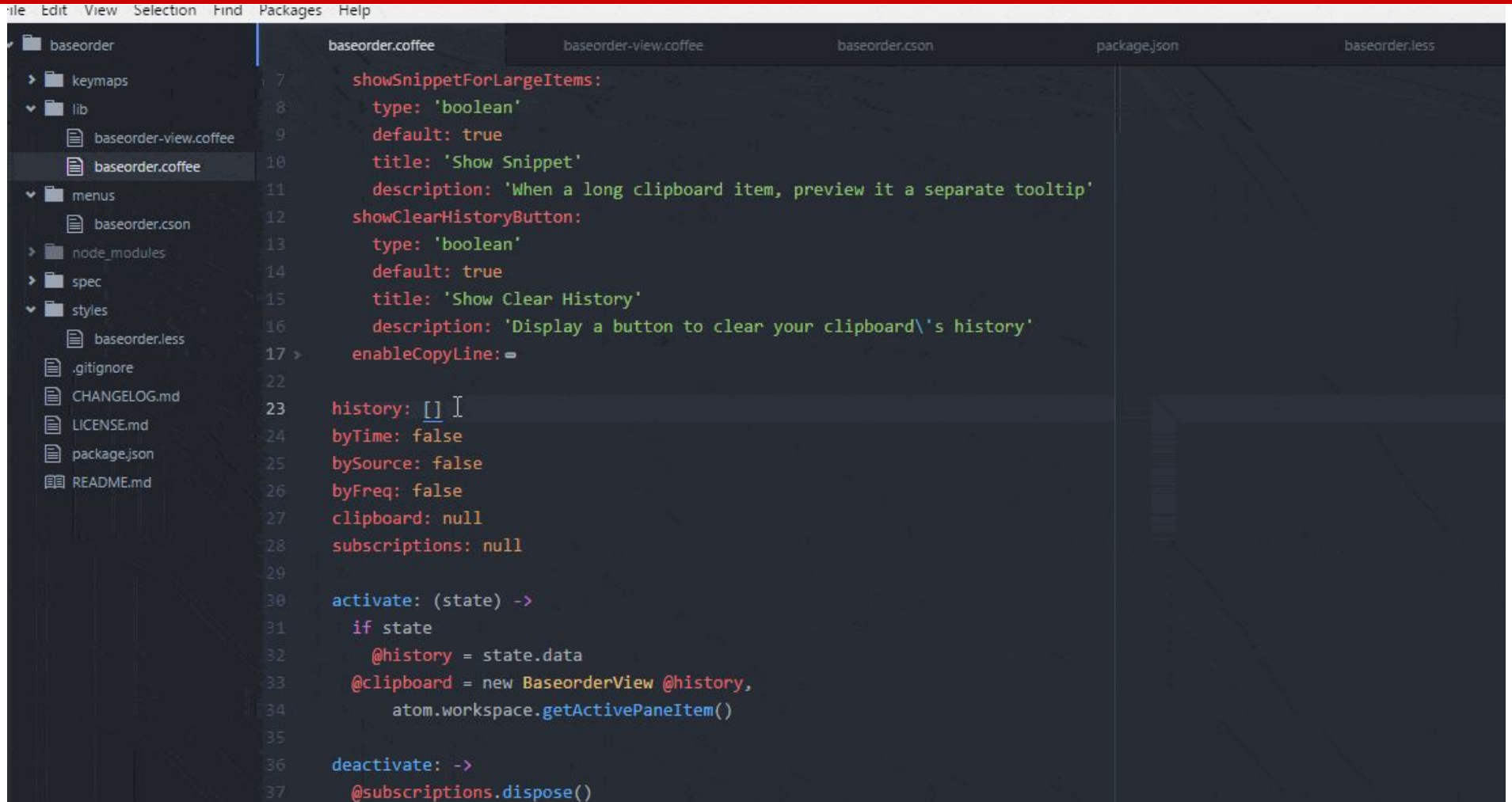
Clipit-order

A dictionary of copied items?

- Each copied item is labeled
- Quickly look up the dictionary

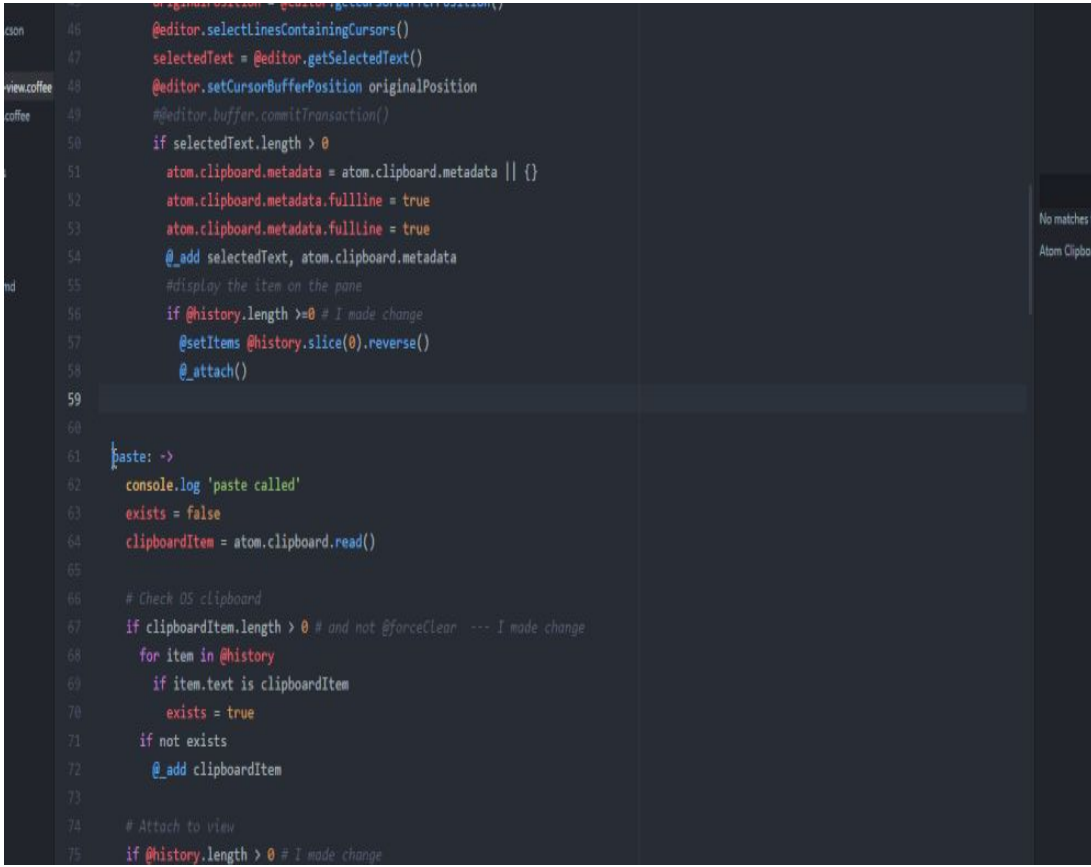
Words are sorted by lexical order,

can we sort copied items by *properties*?


File Edit View Selection Find Packages Help
baseorder
keymaps
lib
baseorder-view.coffee
baseorder.coffee
menus
baseorder.cson
node_modules
spec
styles
baseorder.less
.gitignore
CHANGELOG.md
LICENSE.md
package.json
README.md
baseorder.coffee
baseorder-view.coffee
baseorder.cson
package.json
baseorder.less
showSnippetForLargeItems:
type: 'boolean'
default: true
title: 'Show Snippet'
description: 'When a long clipboard item, preview it a separate tooltip'
showClearHistoryButton:
type: 'boolean'
default: true
title: 'Show Clear History'
description: 'Display a button to clear your clipboard\'s history'
enableCopyLine:
history: []
byTime: false
bySource: false
byFreq: false
clipboard: null
subscriptions: null
activate: (state) ->
if state
@history = state.data
@clipboard = new BaseorderView @history,
atom.workspace.getActivePaneItem()
deactivate: ->
@subscriptions.dispose()

Clipit-panel

- *User interactive* display panel to show copied items statically.
- Programmers can browse the list of copied items, and paste directly from the panel.
- Supports *system wide* copy-pasting.



Clipit-cmd

- Overview
  - Treats history as a *stack*
  - Command interactions
- Hypothesis
  - Fastest feature for power users
  - Higher learning curve
- Commands
  - Cmd-shift-j = Stack pointer--
  - Cmd-shift-k = Stack pointer++
  - Cmd-v = Paste from stack pointer

```
class Test
{
  #this is the first copy
  public void testCopy1()
  {
    sampleCall();
  }

  #this is the second copy
  public void testCopy2()
  {
    sampleCall();
  }

  #this is the third copy
  public void testCopy3())
  {
    sampleCall();
  }

  #this is the fourth copy
  public void testCopy4())
  {
    sampleCall();
  }


}

```

cmd-c

Experimental Design

- Build the base version of Clipit!
- Participants
  - 8 total, 2 for each feature
  - NCSU CS grad students
- Experiment structure:
  - Pre-survey
  - Brief tutorial
  - 10 guided tasks
  - Post-survey

Example Tasks

Task 1

Task 7

[copy source from multiple windows][inside editor copy]

[copy source from multiple windows][paste across multiple windows][inside editor copy]

In /keymaps/clipboard-history.cson you will find keymaps for our package, however, we only have

First we are going to find some missing parts of package.json, you have two copies you need to make that are located in /source/source1 and source2, these are labeled with appropriate comments and their paste locations are also labeled inside package.json.

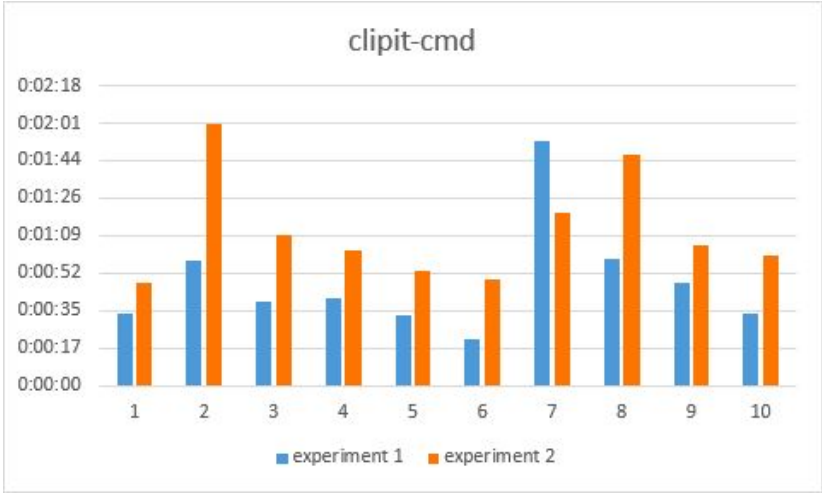
Head on over to /source/source1 to grab your first copy, the windows keymaps. Copy this (remember

Next we need to find some missing pieces of /lib/clipboard-history-view.coffee. There are 2 total copies here and they are located inside /source/source1 and source2, their paste locations are inside clipboard-history-view.coffee and are labeled with comments.

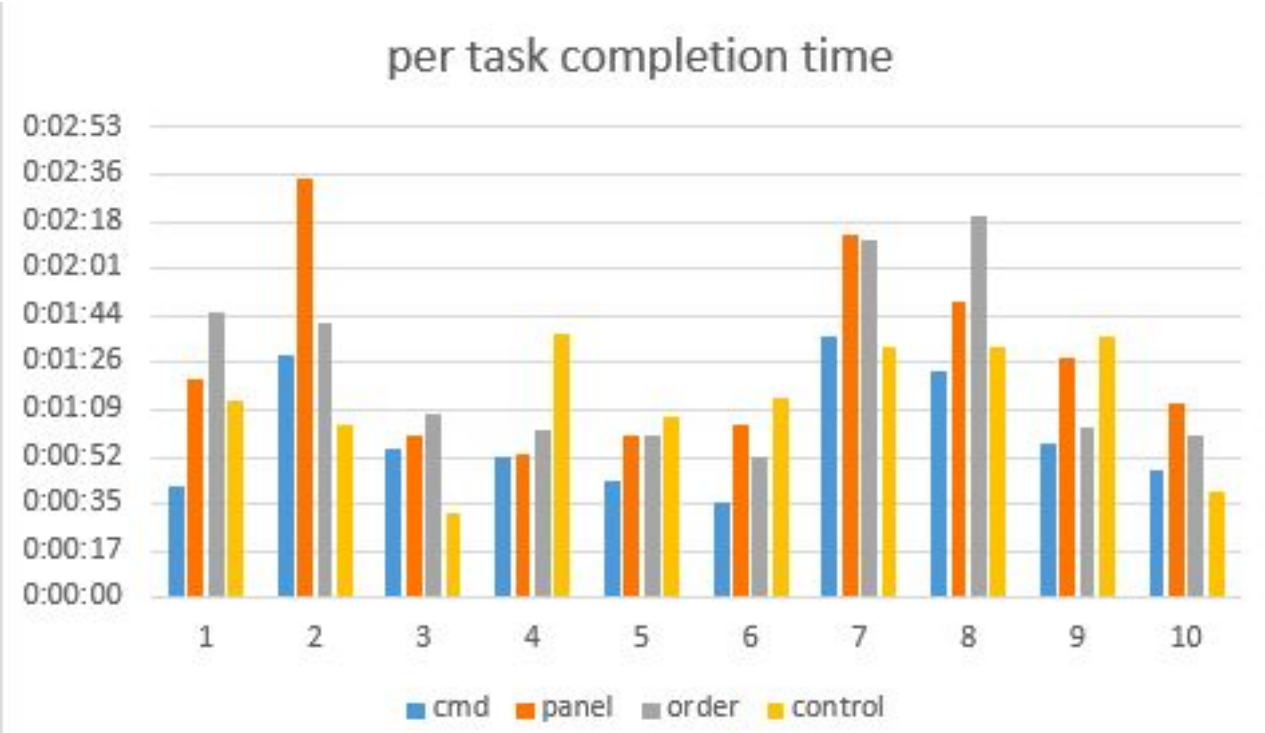
As well head over to /source/source2 to grab the second copy, our linux maps, and repeat the exercise, copying and pasting the block underneath the windows block this time.

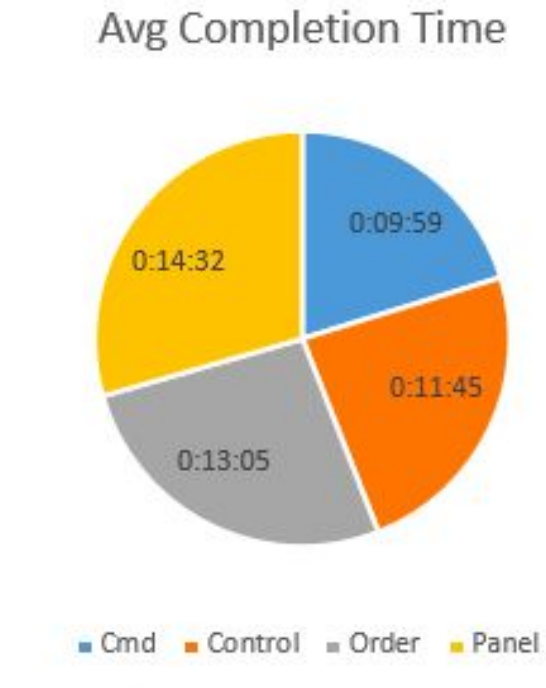
Measuring Performance (AKA Telemetry)

- Background python scripts
- Log each copy and paste
  - Content
  - Time
  - Window/File from where it was done
- Calculate per-task time for each feature

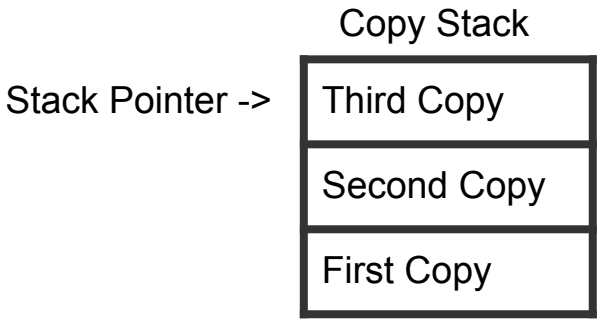


Results


per task completion time
0:02:53
0:02:36
0:02:18
0:02:01
0:01:44
0:01:26
0:01:09
0:00:52
0:00:35
0:00:17
0:00:00
1
2
3
4
5
6
7
8
9
10
cmd
panel
order
control


Avg Completion Time
0:09:59
0:11:45
0:13:05
0:14:32
Cmd
Control
Order
Panel

Best Feature = Clipit-cmd

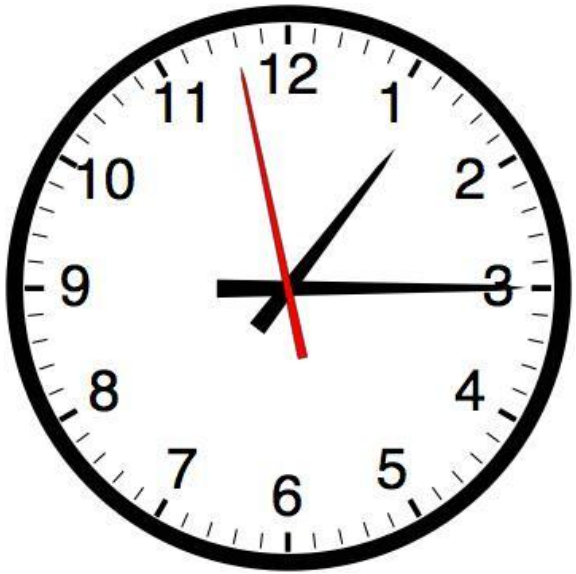


- What makes the best feature?
  - Most efficient (avg completion times)
  - User satisfaction
- Quantitatively
  - Fastest total avg completion time (by ~2 minutes)
  - Fastest of 3 features on avg for every individual task
- Qualitatively (post-surveys)
  - Highest responses to “would you use this/recommend”
  - Tied w/Clipit-order for highest response to “rate usefulness from 1-10”

Future Work

- Feature additions
  - Clipit-cmd
    - Add a visual component
  - Clipit-order
    - Contextual sorting based on contents
  - Favorite items
  - Clipit-pane
    - Formatting issues
- Experiments/testing
  - Alter task design
  - Rerun experiments

Be Agile!




"That's all Folks!"

Questions?