# Introduction to User Interfaces

COSC346

# User Interface

- A common boundary or interconnection between a machine and a human being
  - Its purpose is to allow a user to control the machine effectively

# Hardware UI


d16 group
OPTIONS
ABOUT
MIDI ACT.
MIDI LEARN
PREV
NEXT
BROWSE
SAVE
SAVE AS
SETT.
LAYER SELECT
ENABLE
MUTE/SOLO
SYNTHESIS
MODULATION MATRIX
MASTER MIXER
LuSH-101
Multitimbral Polyphonic Synthesizer
MASTER VOLUME
LAYER 1
LAYER 2
LAYER 3
LAYER 4
LAYER 5
LAYER 6
LAYER 7
LAYER 8
FX 1
FX 2
FX 3
FX 1 - REVERB
FX 2 - DELAY
FX 3 - CHORUS
PITCH BENDER
MOD WHEEL

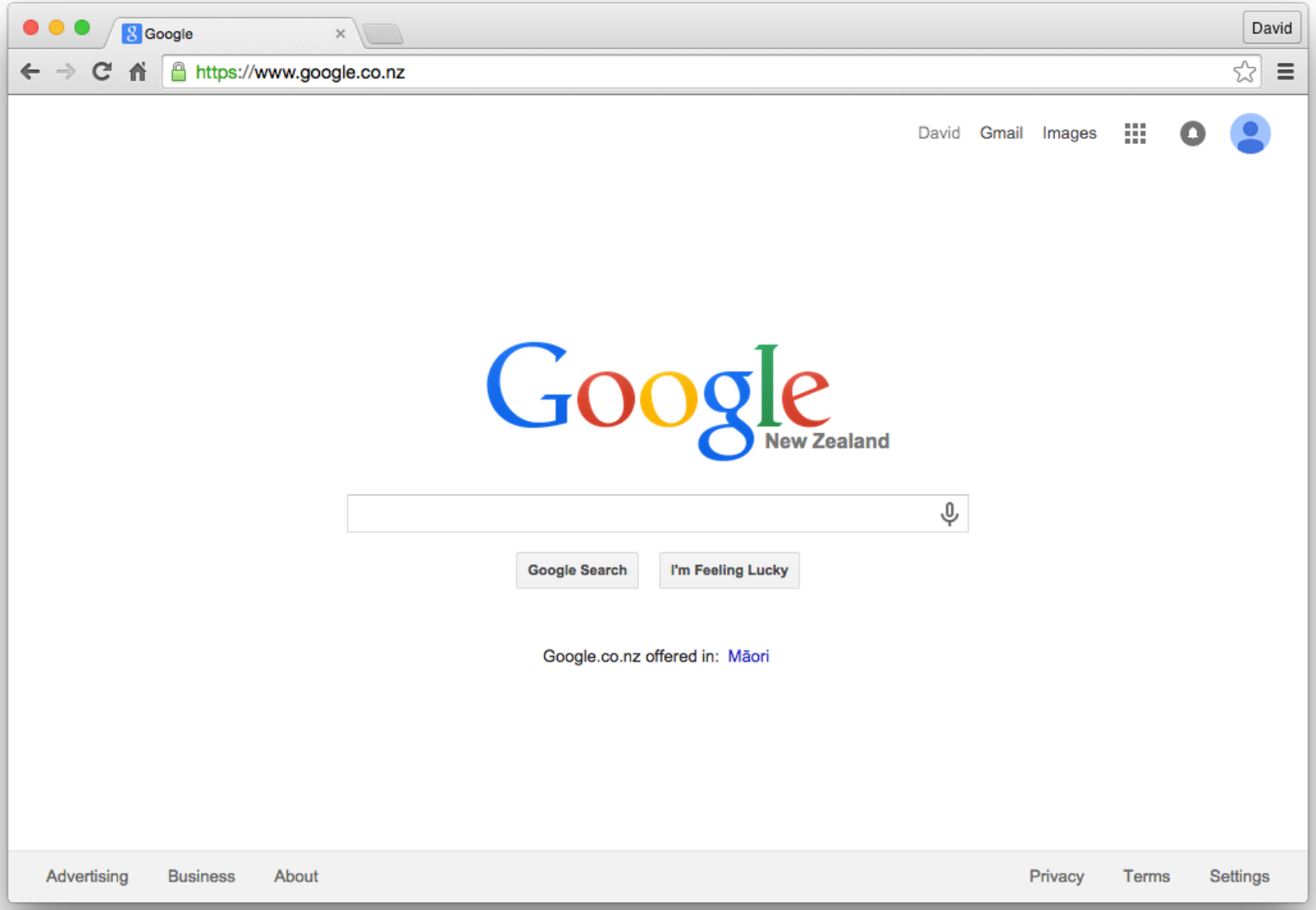
# Software UI


Performance
08 crazy guitar
11 bongos 06
11 Bass-01
SHOW FILTER
FACTORY CONTENT
AUDIO FILES
drum loop
Name
Rating
Key
Tempo
Bars & Beats
23 drumloop 089
26 Drumloop 094
27 Drumloop 093
27 Drumloop 096
36 Drumloop 100
37 drumloop 096
38 Drumloop 097
38 Rock Drum Loop
40 Drumloop 098
40 drumloop 128
42 Drumloop 098
43 Drumloop 108
46 Drumloop 118
1 - HALion Sonic SE
Clarinet
HALION SONIC SE
[GM 072] Clarinet
steinberg
Tone Color
Emphasis
Staccato
Pan Key Follow
EQ Low Gain
EQ Low Mid Gain
EQ High Mid Gain
EQ High Gain
FabFilter Simplon
No Preset
Filter Routing
Filter 1 Freq
Filter 1 Peak
Filter 1 Slope
Filter 2 Freq
INSERT FX 1
SEQUEL

# Complex UI

http://99designs.com/designer-blog/2012/06/20/7-user-interface-design-trends-you-need-to-know-about/

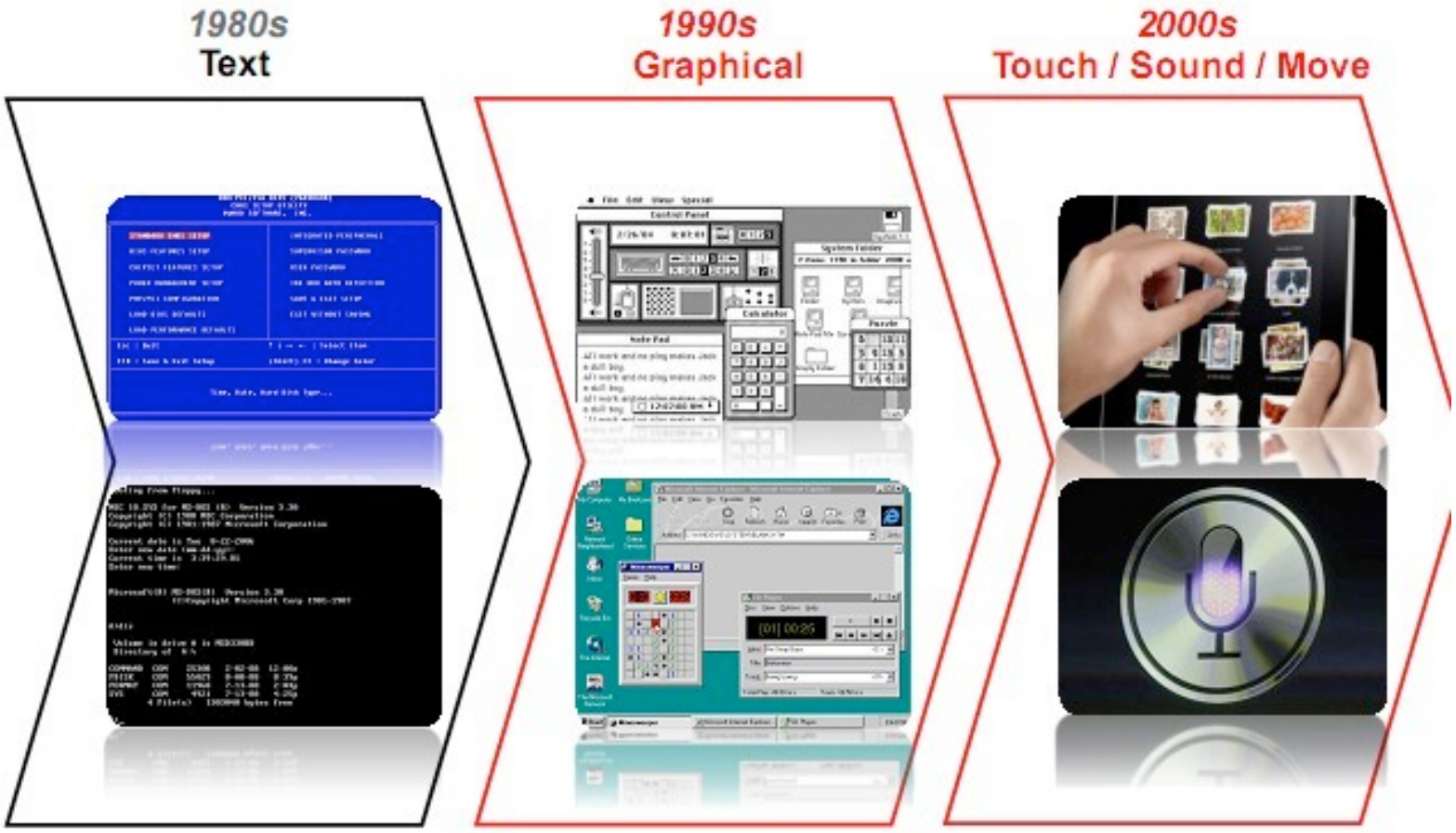
# Simple UI


Google
David
https://www.google.co.nz
David
Gmail
Images
Google
New Zealand
Google Search
I'm Feeling Lucky
Google.co.nz offered in: Māori
Advertising
Business
About
Privacy
Terms
Settings

# UI Norms

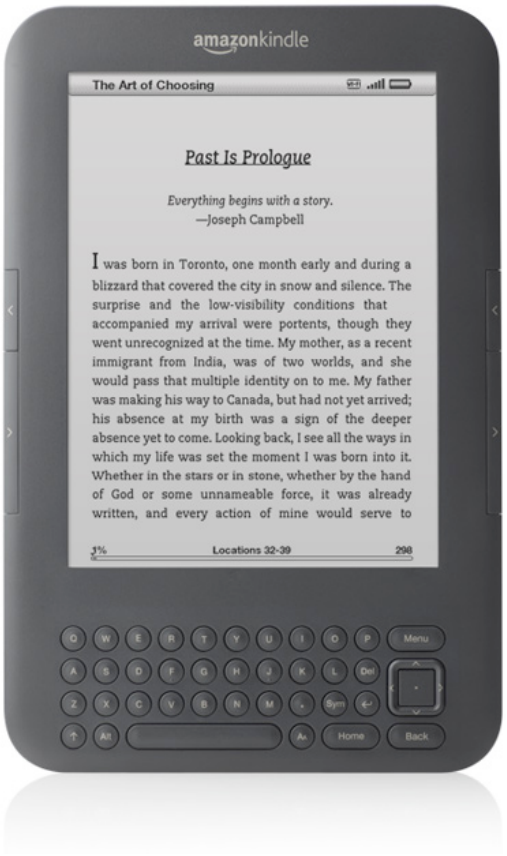

AT&T
replaytv
SONICblue
DVD
Pioneer
SONY
TV
Pioneer
Panasonic
LIGHT TOWER
UNIVERSAL
PHILIPS

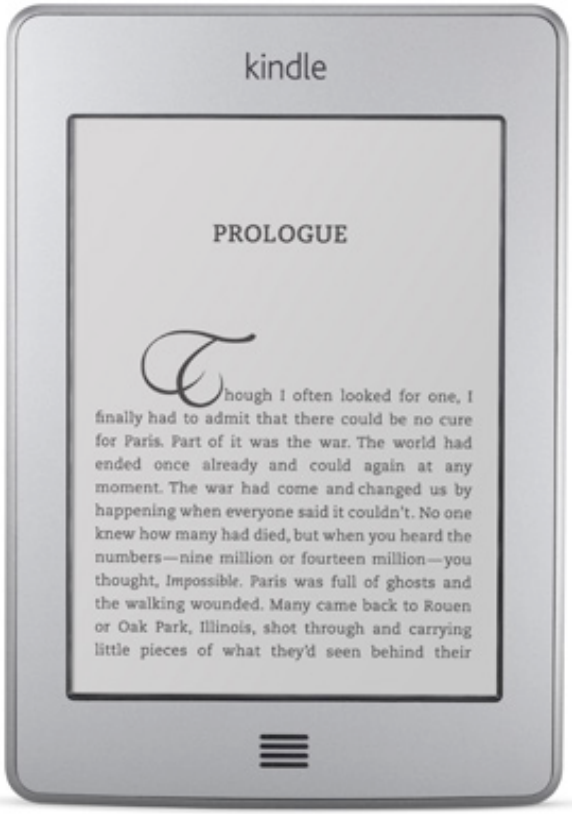
# Evolution of UI




http://www.ixwebhosting.mobi/2011/10/20/5900.html

# Evolution of UI

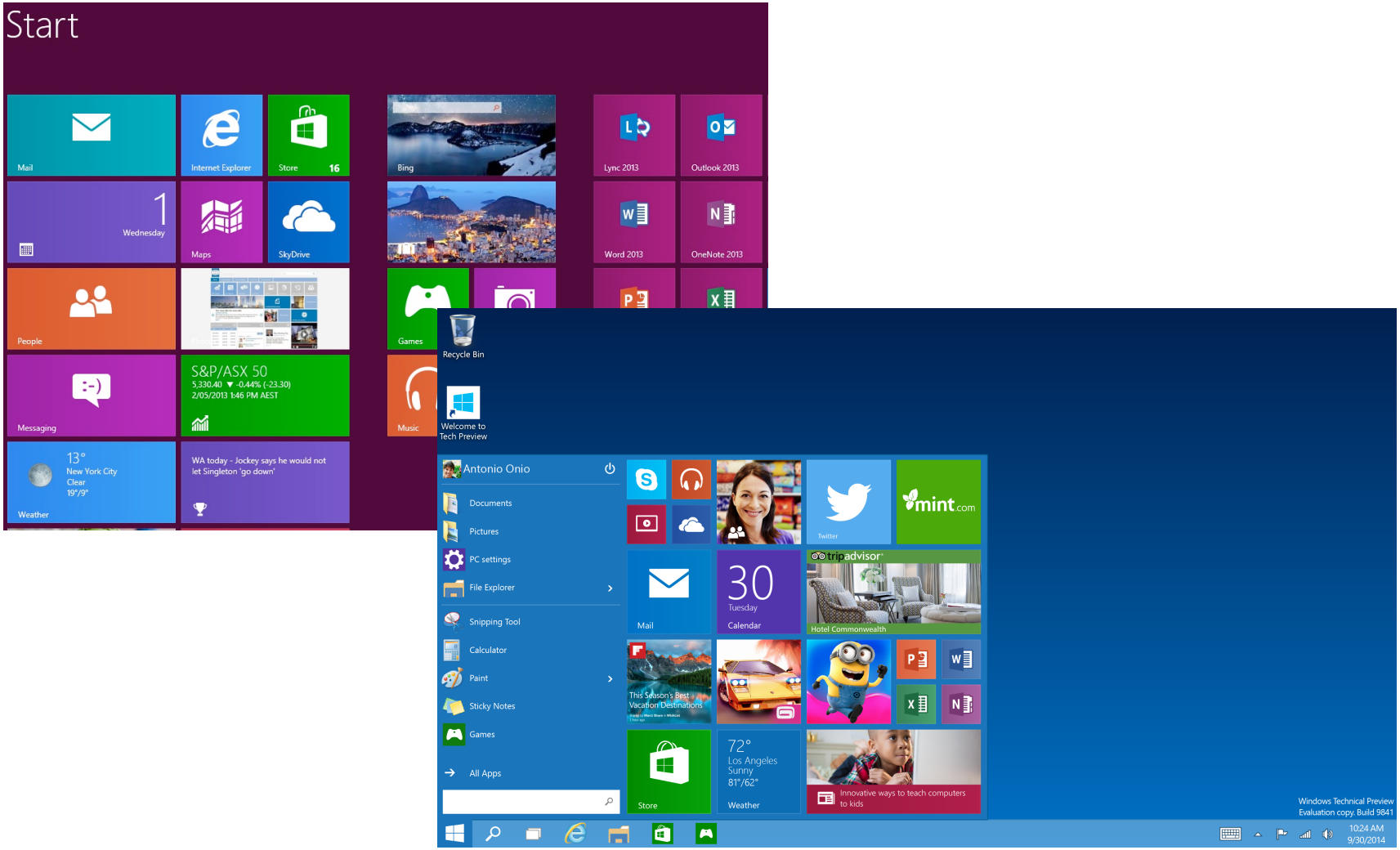


# Evolution of UI

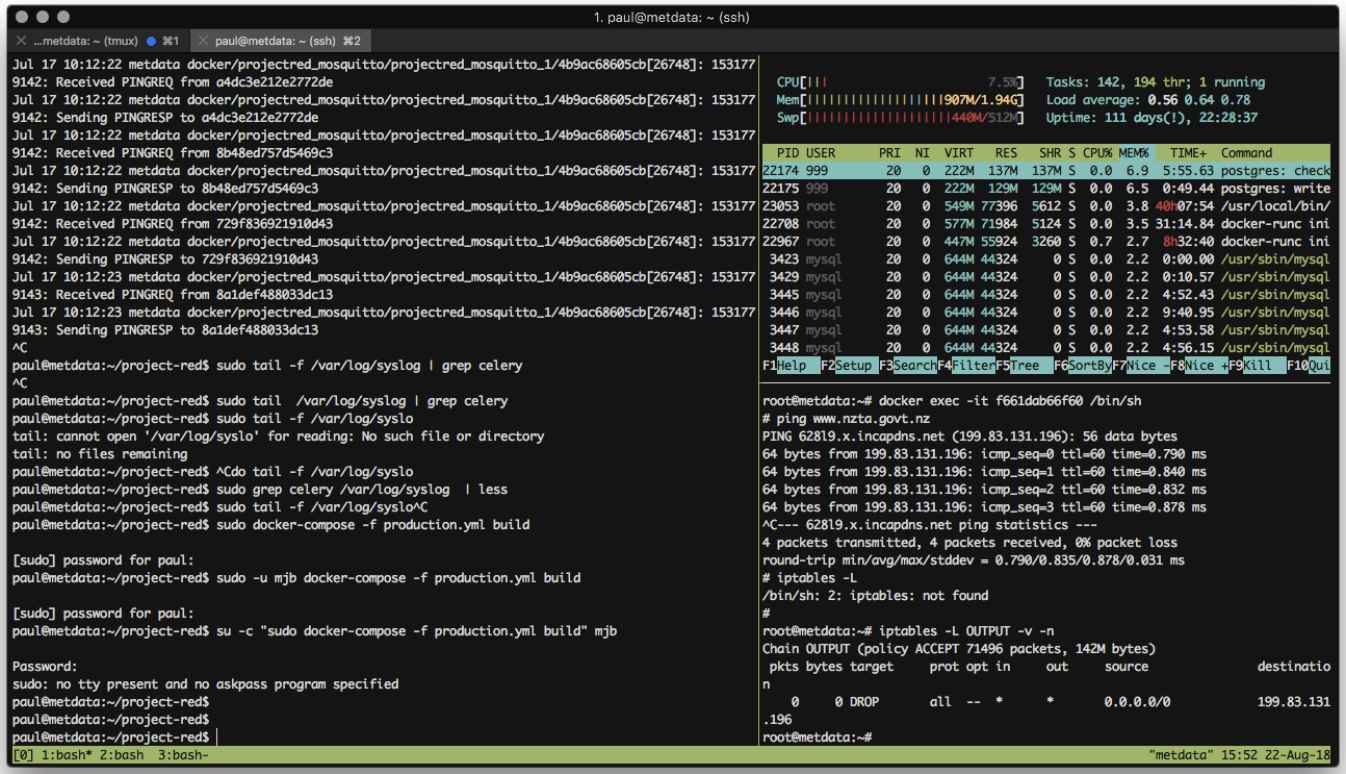

# Evolution of UI


# Evolution of UI? (again?)



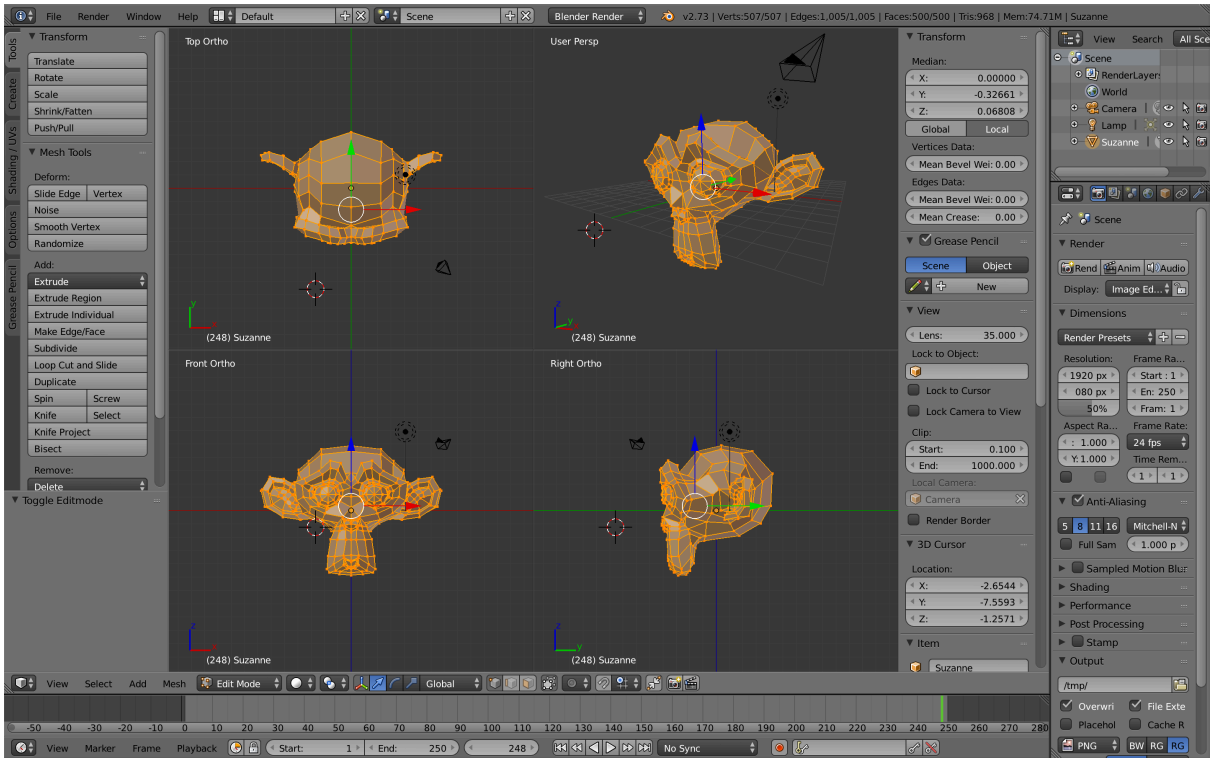
# Command Line Interface

- Means of interacting with a program through a succession of text commands



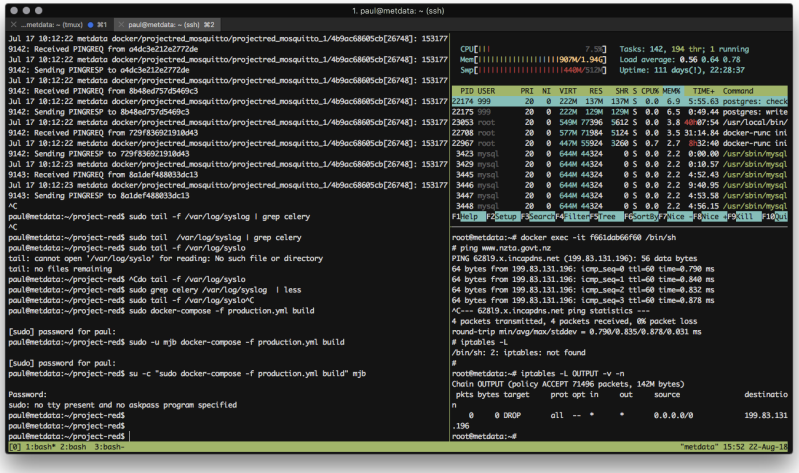
# Graphical User Interface

- User interface implemented with graphics as opposed to text based commands



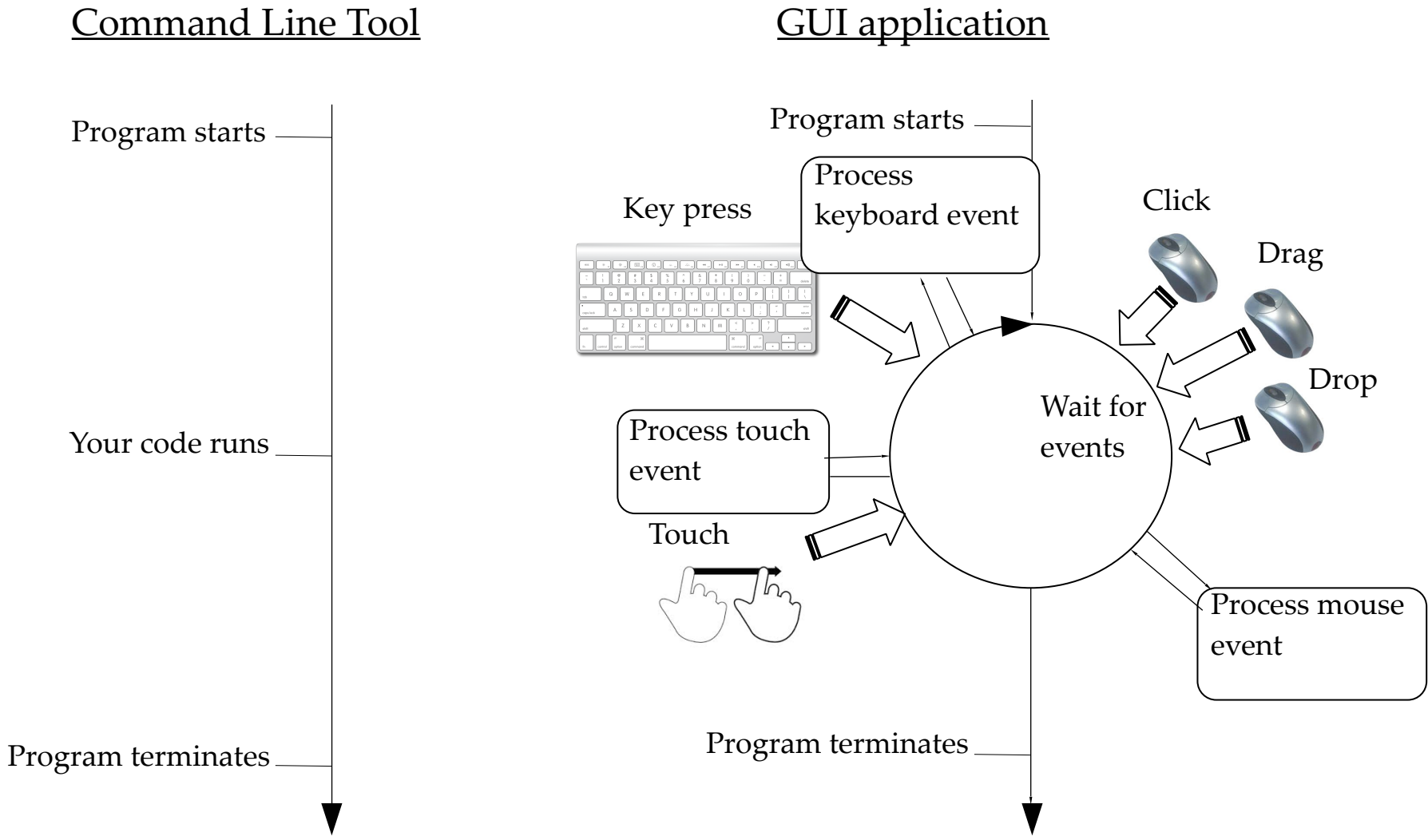
# Command Line Tools vs. GUI Applications

- Command line tools are characterised by:
  - Serial interaction
  - Keyboard input
  - Text-based interface
  - One tool per program
  - Low user expectations



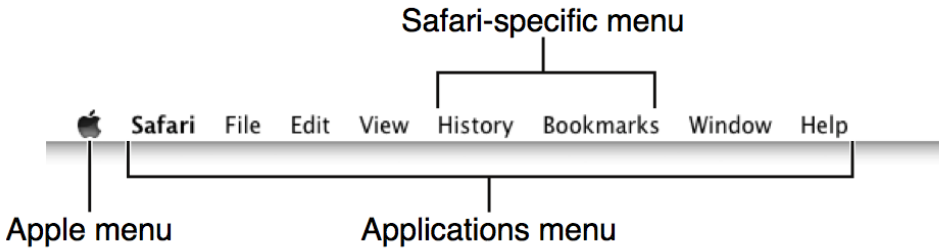

- GUI applications are characterised by:
  - Multiple points of interaction
  - Mouse/keyboard input
  - Graphics-based interface
  - Multiple tools per application
  - High user expectations

# Command line tools vs. GUI applications



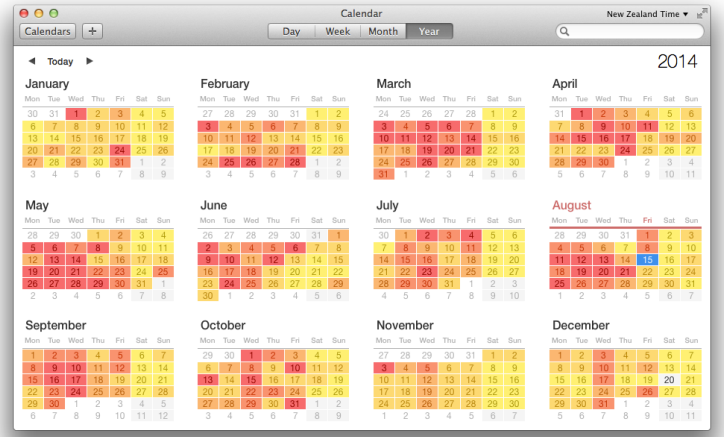
# Anatomy of an Application

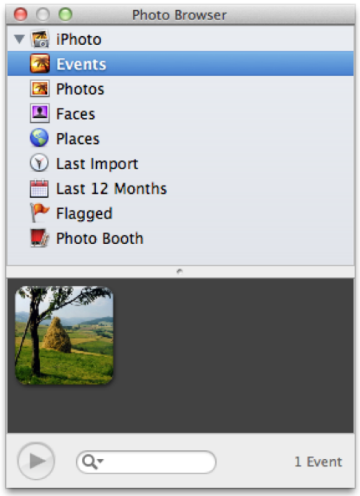
- Applications are often expected to have certain features, typically provided in the menu bar:
  - **General**: about, preferences, …
  - **File**: load, save, print, …
  - **Edit**: undo, redo, cut, copy, paste, …
  - **Format**: font size, style, …
  - **View**: toolbars, full-screen, …
  - **Window**: zoom, minimise, bring to front, …
  - **Help**: should do something useful!
- Cocoa provides support for most of these common tasks



# Windows

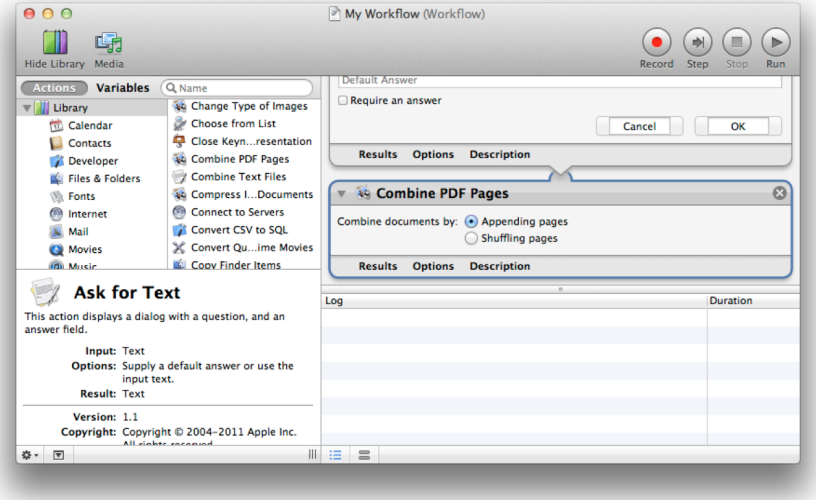
- An application usually displays information via different types of windows:
  - **Application/Document windows**:
    - Main application window.
    - For document-based applications, will contain user data to be manipulated/edited.
  - **Panels**:
    - For showing tools or controls that affect document/application windows.
  - **Dialog boxes**:
    - Requests information from the user that is required before program can continue.




# Controls

- Within a window, you can use different types of controls to interact with the user.
  - Examples include: push buttons, checkboxes, radio buttons, date pickers, progress indicators, text fields, tabs, …




  - Controls come in different sizes, shapes, behaviours and are even designed to look good in different situations.

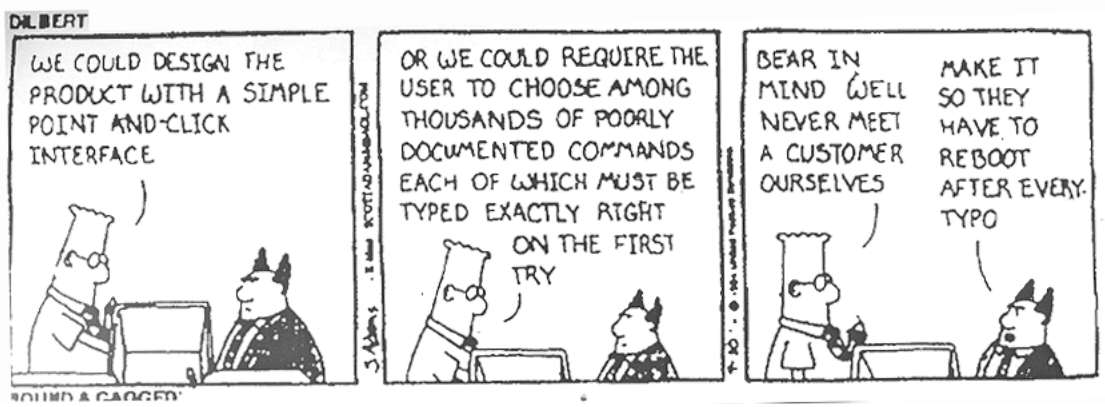
# Input

- User input is via the keyboard and mouse
- Windows can be *active*, *key*, or *click-through*:
  - The **active window** is the window currently responding to mouse events
  - The **key window** is the window currently responding to keyboard events—usually active window is also key window
  - A **click-through window** is inactive but can still respond to mouse events
- Controls handle the key and mouse events themselves  or you can intercept key and mouse events explicitly
- One of the main differences between Cocoa and Cocoa Touch is the support of key/mouse input versus touchscreen input

# Interface Design

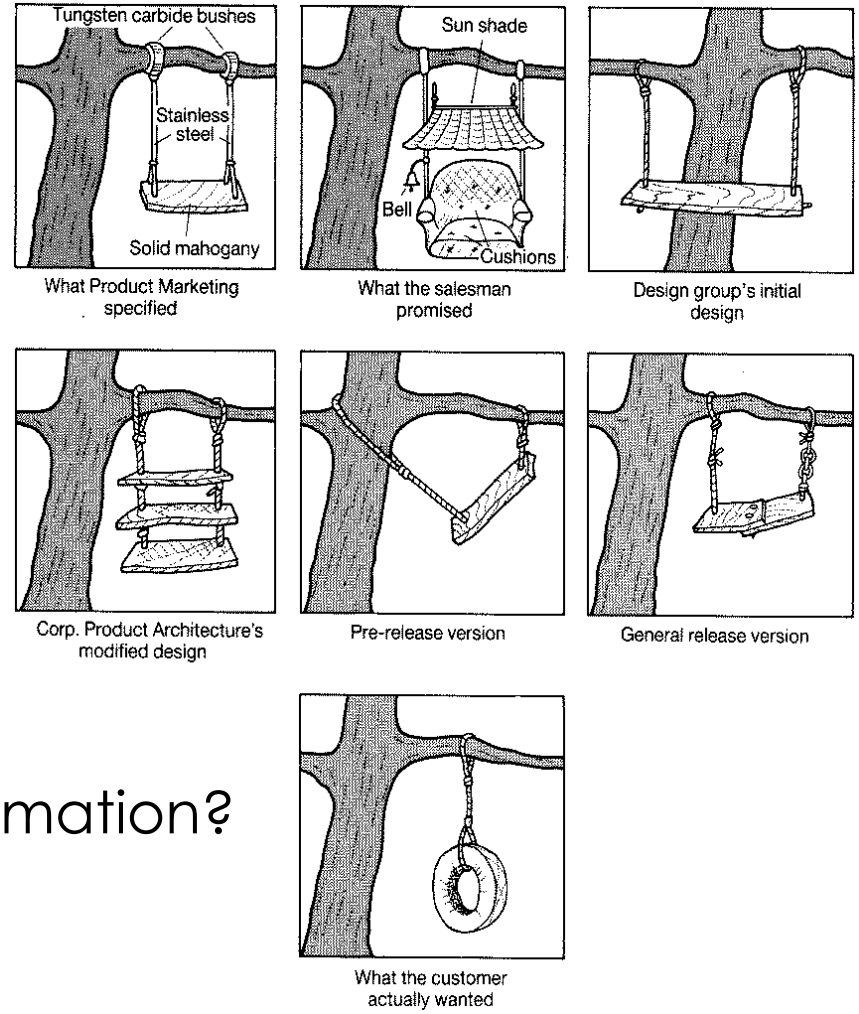
- In addition to how and application works and what it looks like, we need to make it *usable*
  - *Interface Design* is the how you make your application usable

- To make your application usable, you must (at a minimum):
  - Know your users
  - Organise your content
  - Provide easy navigation
  - Provide easily understood layouts with functional controls



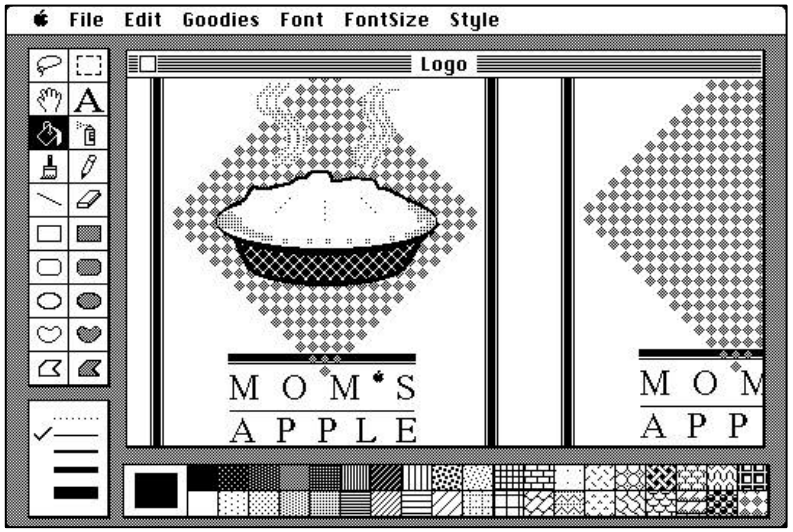
# Users

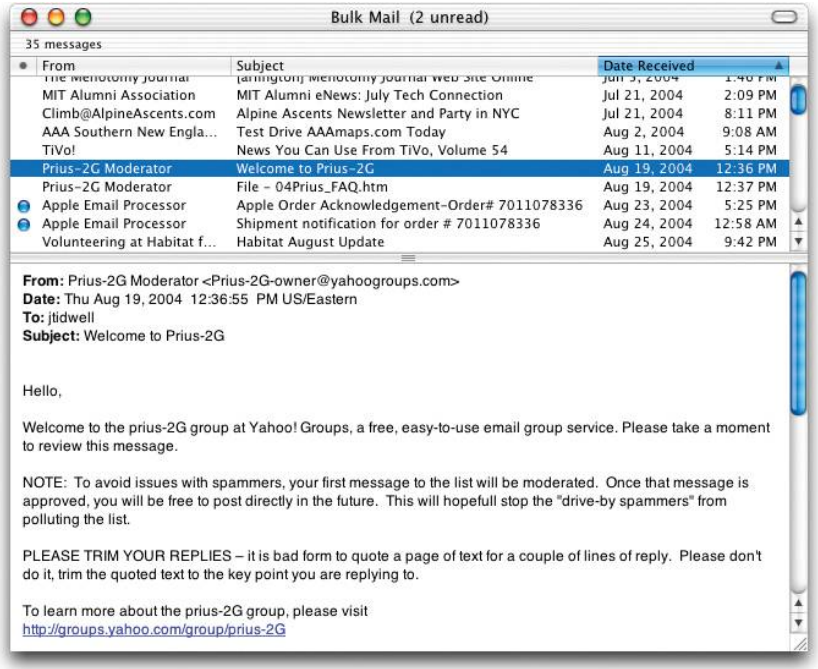
- In order to design a good user interface, you have to understand the user
  - User goals
  - Specific tasks they require
  - Language they use
  - Skill level
  - Attitude
- How do you acquire this information?
  - Observation
  - Case Studies
  - Surveys
  - Personas (for modelling)



# Content

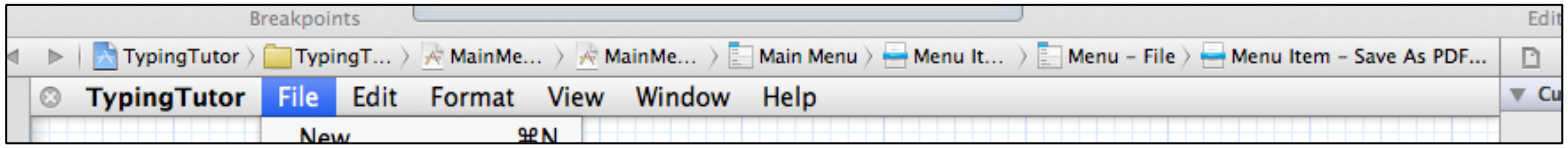
- In any application, you can organise content using various design patterns:
  - Multiple windows (*e.g.*, Microsoft Word)
  - Windows with panes (*e.g.*, Microsoft Outlook, Apple Mail)
  - Canvas & palettes (*e.g.*, Photoshop now, MacPaint long ago)
  - Wizards (*e.g.*, installers)
  - Browsers (*e.g.*, help)



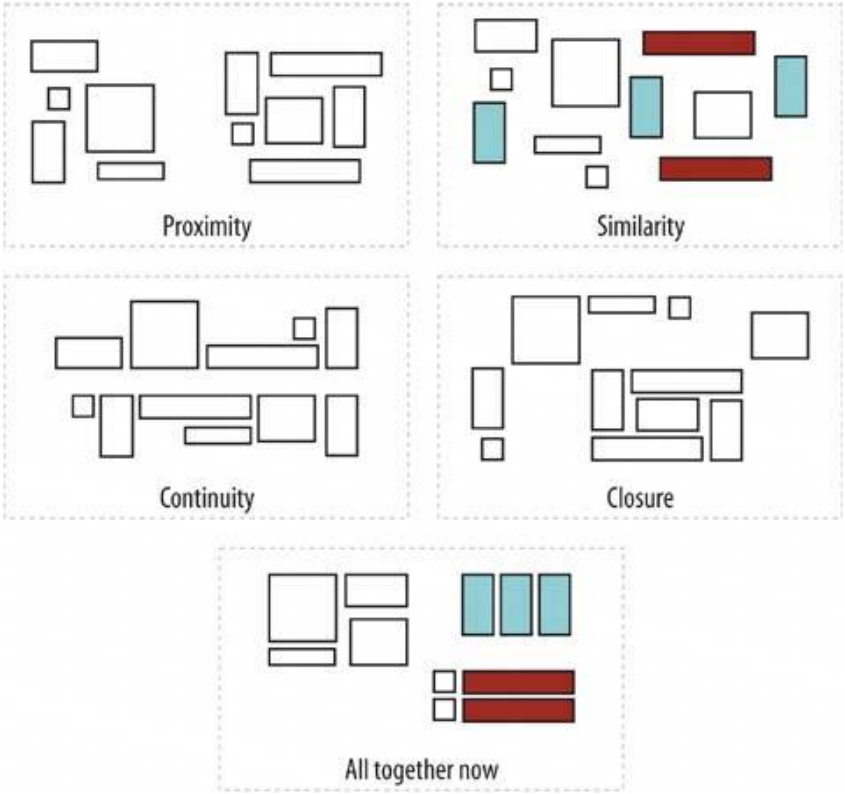
# Navigation

- A sufficiently complex application will require navigation (to get around within the content)
- Some **navigational patterns** include:
  - Signposts, clear entry points, pyramid structure, hub and spoke structure
  - Colour-coded sections, sequence maps, breadcrumbs, annotated scroll bars, escape hatches
  - Animated transitions
- For example:
  - Minimise window (animated transition), Back button (escape hatch), jump bar in Xcode (breadcrumbs)



# Layout

- How you layout your windows and dialogs determines ease of use and aesthetics
  - **Gestalt principles** help express connections within a layout
- Different types of controls have been designed for different circumstances
  - Buttons, progress indicators, various menus, action panels



# Summary

- Designing a good GUI application can be difficult!
- Technical Know-How:
  - AppKit framework
  - Model-View-Controller design
- Providing expected capability:
  - About, preferences, load, save, undo/redo, copy/paste, etc.
- Interface Design
  - Understanding user requirements/ways of thinking
  - Providing functional, aesthetic organisation, navigation, layout
- However, Xcode and Interface Builder make the process relatively painless, once you get used to the way everything works