CS 338: Graphical User Interfaces

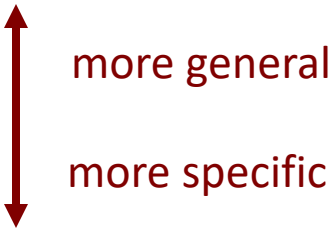
Lecture 6-1:
Theories & Guidelines

The GUI development process

- We've now seen some tools that enable us to build our interface.
- So let's get back to the GUI process...
  - How do we go from idea to design?
  - How do we specify and evaluate this design?
  - How do we go from design to implementation?

Focus on design

- The assumed process to this point:
  - (1) we have an idea for an interface
  - (2) we understand our intended users
    - that is, we've got personas for our system
    - let's ignore requirements gathering, etc. for now
- We need to specify an initial design, or prototype, for our interface.
- We will examine 3 constraints to guide us:
  - (high-level) theories
  - (mid-level) principles
  - (low-level) guidelines

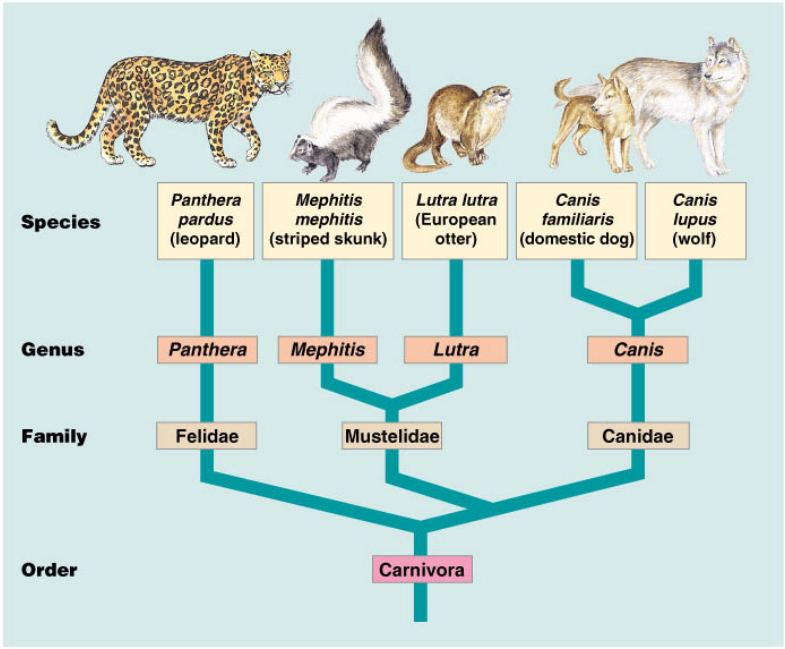


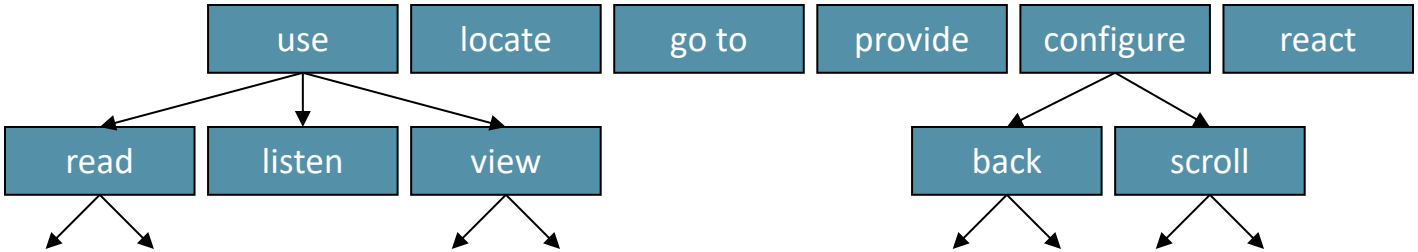
Theories

- Two types of theories useful for our purpose
  - explanatory: given a phenomenon, try to understand its causes, effects, etc.
  - predictive: given a task/interface, try to predict what will/can happen in the course of execution
- Same idea as theories in other domains!
  - physics: how does a ball fly through the air?
  - chemistry: what happens when we mix X & Y?
  - HCI: how long does it take to do task X?
    - "cognitive engineering",
      or "engineering models of user behavior"

Theories

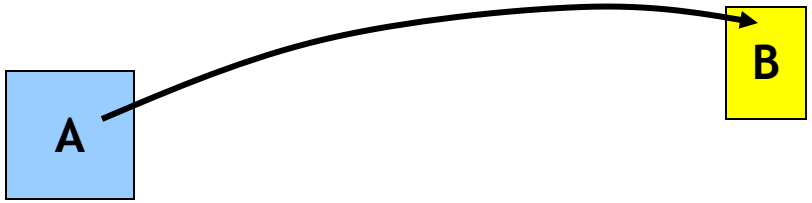
- Taxonomies
  - puts order on complex set of phenomena
  - we usually think of animals →
  - remember Byrne et al.'s "taskonomy"?





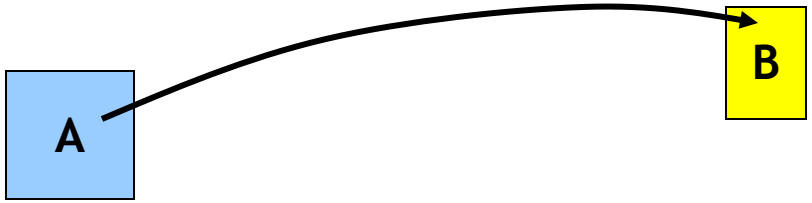
Theories

- Fitts' Law
  - how long does it take to move a mouse A→B?



Theories

- Fitts' Law
  - how long does it take to move a mouse A→B?



for B with width *w* and at a distance *d*…

$$T = c_1 + c_2 \log\left(\frac{2d}{w}\right)$$

Theories

- GOMS modeling
  - Goals, Operators, Methods, Selection Rules
  - "keystroke-level" model of human performance
    - one version of GOMS, anyway (KLM)
  - primary operators
    - K: keystroke 280 ms
    - P: point with mouse to a target 1100 ms
    - B: press or release mouse button 100 ms
    - BB: click mouse button 200 ms
    - H: "home" hands to kb/mouse 400 ms
    - M: "mental" act of thinking 1200 ms

Theories

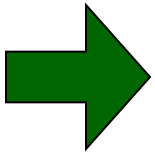
- GOMS modeling
  - phone example

Description	#	Type	Time
"turn on phone"	1	M	1.20
press 'Phone'	1	K	.28
"dial number"	1	M	1.20
type digits	10	K	2.80
"send number"	1	M	1.20
press 'Send'	1	K	.28
		Total Time	6.96

Theories

- Consistency through grammars
  - consistent command set → simple grammar

delete/insert character
delete/insert word
delete/insert line
delete/insert paragraph



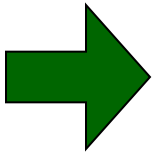
action -> delete
action -> insert

object -> character
object -> word
object -> line
object -> paragraph

command -> *action object*

  - inconsistent command set → complex grammar

delete/insert character
remove/insert word
destroy/create line
kill/create paragraph



ugly!

Principles

- Theories provide general descriptions of human behavior, performance, etc.
  - very valuable for understanding
  - so far, have had more limited practical value
- Principles embody particular aspects of theories applied in context
  - still somewhat general, but more prescriptive
    - they tell you what to do
    - though they may not tell you specific steps to doing it
- Let's look at some recognized principles of design (as found by practitioners)...

Principle #1

- P#1: Recognize the diversity
- We've emphasized this from the beginning
- All design depends on the user's...
  - age
  - education
  - physical abilities
  - cultural background
  - training
  - motivation
  - personality
  - goals

Thinking about individuals is often not feasible. Can we construct *classes* of users with these parameters?

Principle #1

- Novice / first-time users
  - novice are truly novice; first-timers have skills, but just don't know the interface
  - rules of thumb...
    - restrict vocabulary to simple, necessary terms
    - keep (sub-)tasks to minimal number of actions
    - provide informative feedback, simple error messages, and on-line step-by-step help

Principle #1

- Intermittent / intermediate users
  - have the basic knowledge, but don't see the interface often enough to become expert
  - rules of thumb...
    - "take off the training wheels"
    - consistent structure of views & menus
    - consistent terminology
    - visuals emphasize recognition over recall
    - "protection from danger"
    - reference materials (on-line or printed)

Principle #1

- Frequent / expert users
  - "Power" users thoroughly familiar with both task and interface
  - rules of thumb...
    - rapid response times all around
    - macros (built-in or recordable) for frequent sequences of commands
    - shortcuts: keystrokes, abbreviations, etc.

Principle #2

- P#2: Use the Eight Golden Rules of Interface Design
- (1) Strive for consistency (again).
  - check all aspects of the interface for consistency
    - terminology in prompts, menus, help
    - text/image color, layout, caps, fonts
  - there are exceptions... but limited!!
- (2) Enable frequent users to use shortcuts.
  - just said this, but worth repeating

Principle #2

- (3) Offer informative feedback.
  - every action (within reason) should get feedback
  - frequent/minor actions → subtle feedback
    - e.g., dragging; how about clicks/keystrokes?
  - infrequent/major actions → obvious feedback
    - though limit use of most obvious! (e.g., dialogs)
- (4) Design interaction to yield closure.
  - after performing subtask (sequence of actions), user should feel closure
  - ("take one dip and end it!")

Principle #2

- (5) Offer error prevention and handling.
  - users should not be able to make a serious error
    - OS's do this (memory protection: no "poke 1,0")
    - UI's are no different!
  - if system detects error, recovery should be quick
    - e.g., don't re-type command -- fix faulty part
- (6) Permit easy reversal of actions.
  - one word: UNDO
  - increases comfort level, decreases frustration
  - encourages exploration of the unfamiliar

Principle #2

- (7) Support internal locus of control.
  - make the user feel like they're in control
  - surprises are good for horror movies, not UI's
  - "avoid acausality" (huh?)
    - if you know the cause & effect, you're happy
- (8) Reduce short-term memory load.
  - humans remember 7±2 things
    (normal humans, not Drexel students ☺ --
    they're more likely around 10 or 12)
  - the more to recall, the more expertise is needed for efficient use
  - let users find functions, not recall them

Guidelines

- Guidelines are more specific, prescriptive methods for designing interfaces
  - line between principles and guidelines is fuzzy
  - actually, some use the terms interchangeably
- You be the judge...
- Let's look at some sample guidelines...
  - and maybe some of this will sound familiar...

Guidelines for Display Organization

- (1) Consistency of data display
- (2) Efficient information assimilation
  - use common formatting (e.g., left vs. right justification) for text, numbers, etc.
- (3) Minimal memory load on user
- (4) Compatibility of display with data entry
  - clear ordering, clear links
- (5) Flexibility for user control of display
  - allow users to change columns, sorting, etc.

Guidelines for Data Entry

- (1) Consistency of entry transactions
- (2) Minimal user input actions
- (3) Minimal memory load on user
- (4) Compatibility of display with data entry
- (5) Flexibility for user control of entry
  - but beware: when does flexibility interfere with consistency guideline?

Putting it all together

- We have theories, principles, & guidelines.
  - actually, theories → principles → guidelines
- But is there something **specific** to follow?
- Yes! — the various "look-and-feel" guidelines
  - Apple's Human Interface Guidelines
    - the original, the classic (1987 to today)
    - See: https://developer.apple.com/design/human-interface-guidelines/
  - Microsoft's Windows Human Interface Guidelines
  - Java Swing Interface Guidelines
  - all are useful because they're detailed, comprehensive, standard, and publicly available
  - Let's look at Apple's for a minute...

Apple Guidelines

- "Following the guidelines is to your advantage because...
  - **users will learn your application faster if the interface looks and behaves like applications they're already familiar with**
  - **users will accomplish their tasks quickly, because well-designed applications don't get in user's way**
  - **users with special needs will find your product more accessible**
  - your application will have the same modern, elegant appearance as other Mac OS X applications

Apple Guidelines

- (quote continued)
  - **your application will be easier to document, because an intuitive interface and standard behaviors don't require as much explanation**
  - **customer support calls will be reduced**
  - **your application will be easier to localize, because Apple has worked through many localization issues in the Aqua design process**
  - media reviews of your product will be more positive; reviewers easily target software that doesn't look or behave the way "true" Macintosh applications do

Apple Design Principles

- (1) **Metaphors from the real world**
  - use concrete metaphors and make them plain, so users have expectations to apply to computer environments
  - whenever appropriate, use audio and visual effects that support the metaphor
- (2) **Direct manipulation**
  - users want to feel in charge of the computer's activities
- (3) **See-and-point**
  - (instead of remember-and-type)
  - users rely on recognition, not recall; they shouldn't have to remember anything the computer already knows
  - the general form of user actions is noun-then-verb, or "Hey, you --- do this"

Apple Design Principles

- (4) Consistency
  - effective applications are both consistent within themselves and consistent within one another
- (5) **WYSIWYG**
  - (What you see is what you get)
  - there should be no secrets from the user, no abstract commands that only promise future results
  - there should be no significant difference between what the user sees on the screen and what eventually gets printed
- (6) User Control
  - user, not computer, initiates and controls all actions

Apple Design Principles

- (7) Feedback and dialog
  - keep the user informed
  - provide immediate feedback
  - user activities should be simple at any moment, though they may be complex taken together
- (8) Forgiveness
  - users make mistakes; forgive them
  - the user's actions are generally reversible; let users know about any that aren't

Apple Design Principles

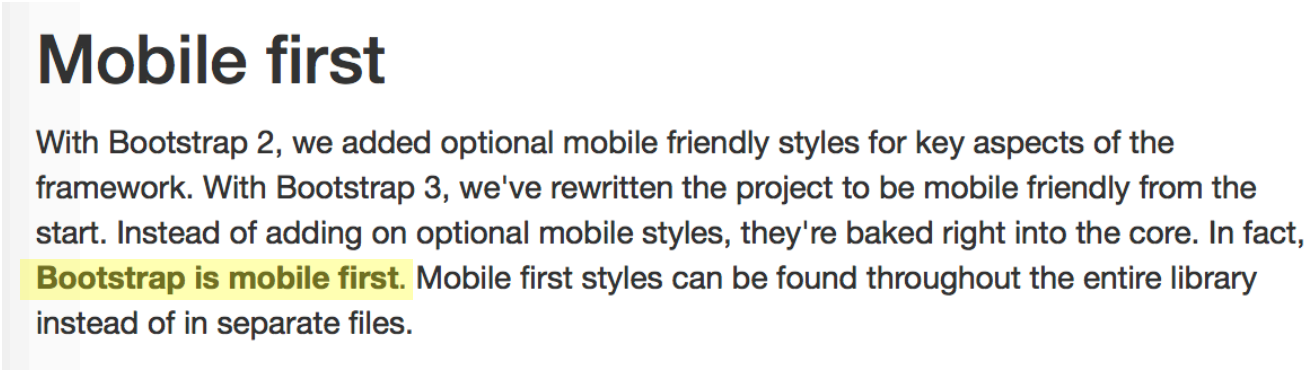
- (9) **Perceived stability**
  - users feel comfortable in an environment that remains understandable and familiar rather than changing randomly
- (10) Aesthetic integrity
  - visually confusing or unattractive displays detract from the effectiveness of human-computer interaction
  - different "things" look different on the screen
  - users should be able to control the superficial appearance of their computer workplaces -- to display their own style and individuality
  - messes are acceptable only if users make them -- applications aren't allowed this freedom

Apple Guidelines

- Ok, so far these principles sound like many of the others we've seen.
- What's the difference? Specific guidelines that go with the general principles...
  - general: "Try for a more centered approach to dialog layout as opposed to the strongly left-biased approach..."
  - specific: "Maintain a 20-pixel space between the left and right edge of the window and any controls"
  - specific: "Set 16 pixels of vertical space between groups of controls"

Bootstrap Guidelines ?

- Honestly, it doesn't seem as deeply thought-out… but there are inklings of guidelines
- For example:

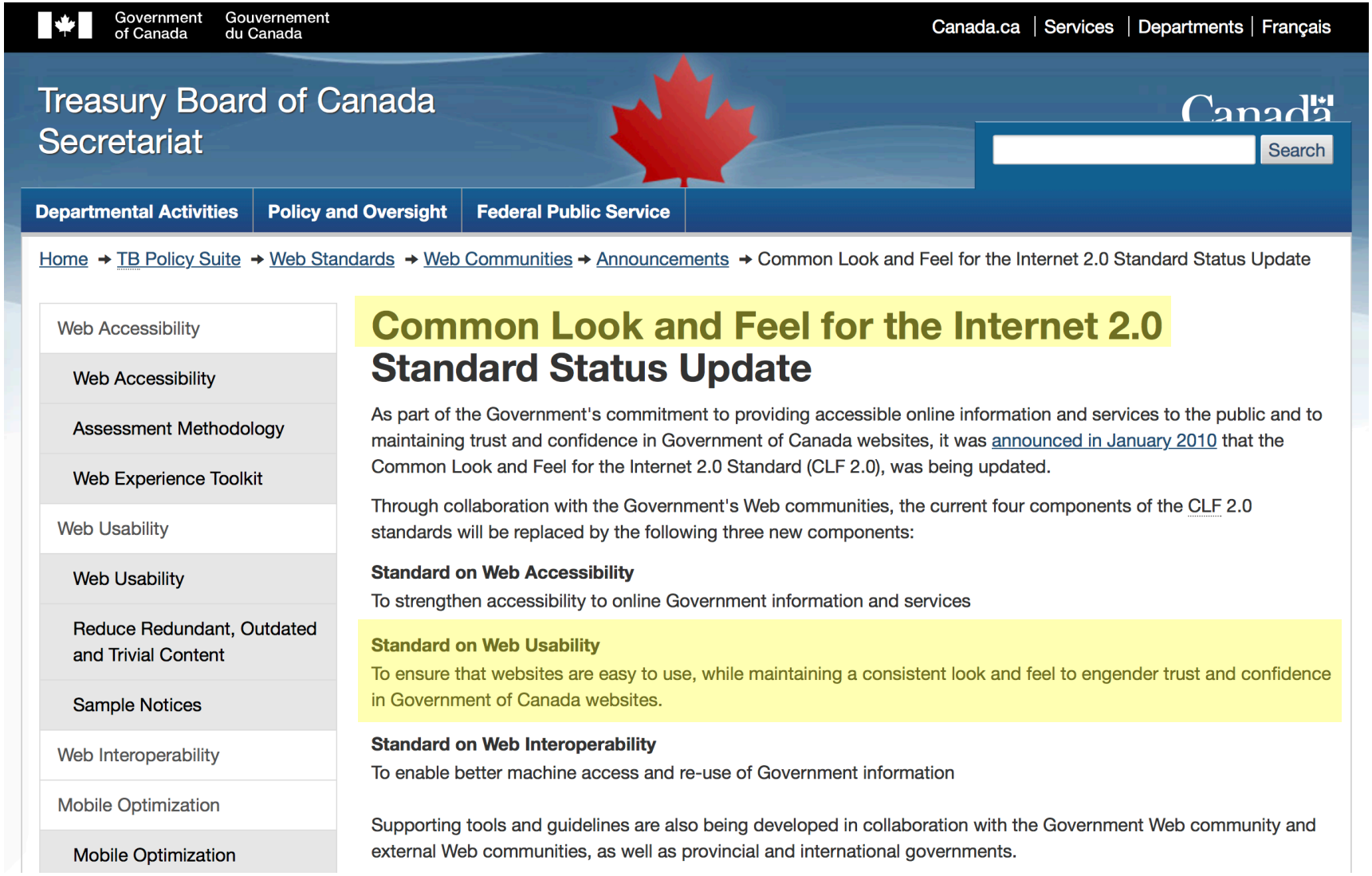


Mobile first

With Bootstrap 2, we added optional mobile friendly styles for key aspects of the framework. With Bootstrap 3, we've rewritten the project to be mobile friendly from the start. Instead of adding on optional mobile styles, they're baked right into the core. In fact, **Bootstrap is mobile first.** Mobile first styles can be found throughout the entire library instead of in separate files.

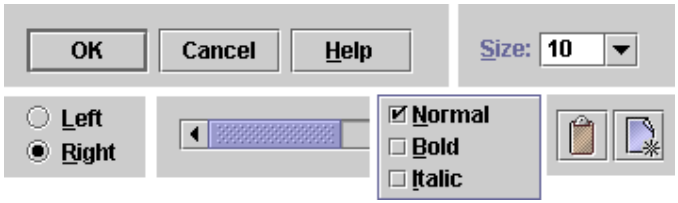
- Also, its reliance on containers and grids
- To be fair, though, its CSS sorta/kinda defines its own look & feel guidelines

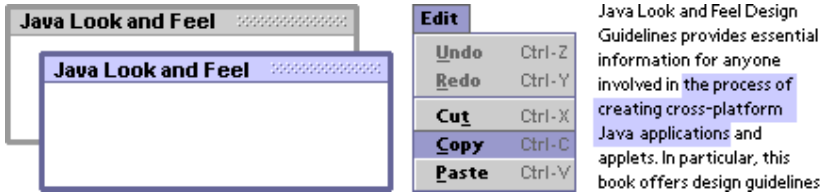
Government of Canada ??

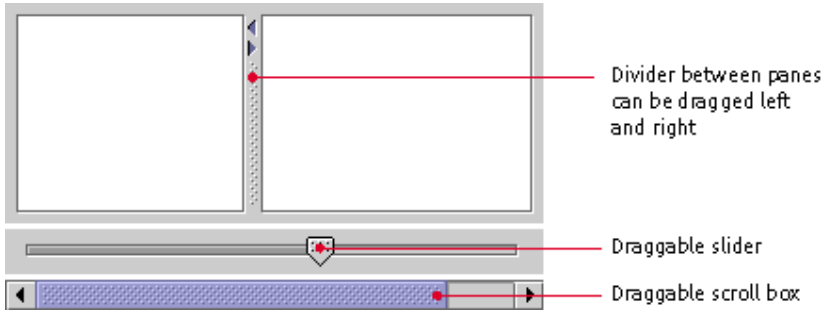


Java Guidelines

- The Java Guidelines work for their Swing GUIs
  - general and specific all relevant to Swing — namely, to their "Metal" look-and-feel
- What do the guidelines guide?
  - "flush 3D" style, "drag" texture, color...





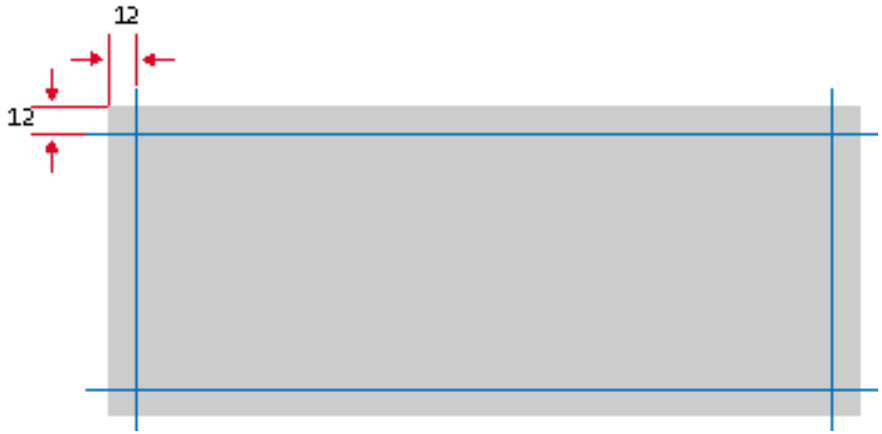


Layout process

- (1) Determine the functional requirements for your dialog box and the type and importance of the dialog box components
  - example: "Find" dialog box
  - what information do we need?
    - find text
    - options: match case? whole word?
  - what components do we need?
    - text box for find text
    - check/radio buttons (etc.) for options
    - command buttons: "Find", "Close"

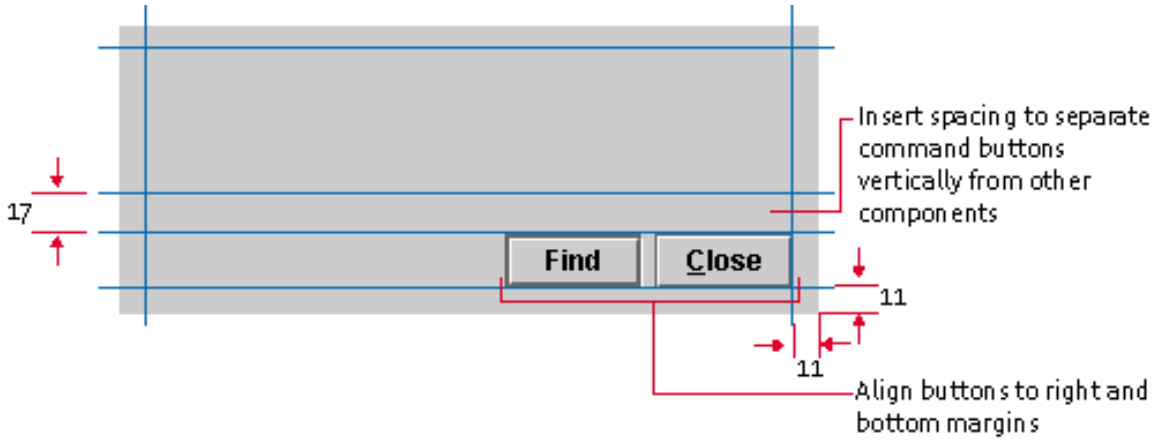
Layout process

- (2) Create a window with 12-pixel margins between the border of the dialog box and its components
  - Include 12 pixels between the top and left borders of a dialog box and its components.



Layout process

- (3) Arrange the command button row and separate it vertically from the rest of the components
  - Place 17 pixels of vertical space between the command button row and the other components.
  - Buttons must have the same width and must right-aligned along the bottom margin.



(We interrupt this program...)

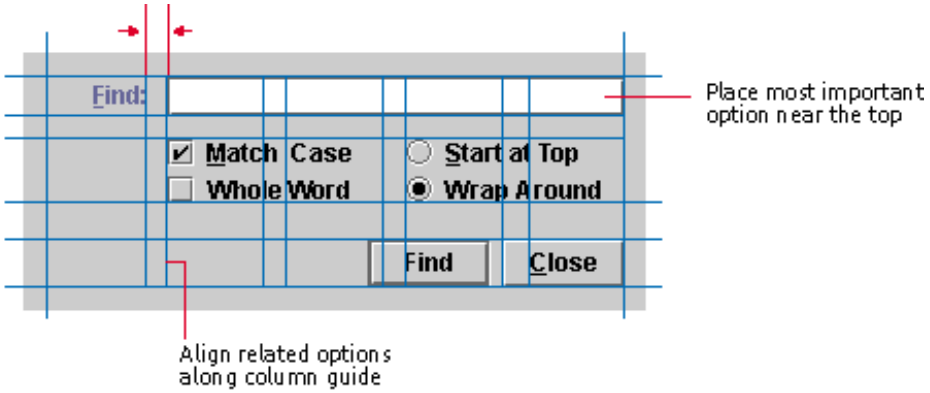
- If you read the Java Human Interface Guidelines, you'll see things like:
  - "Include 11 pixels between the bottom and right borders of a dialog box and its command buttons. (To the eye, the 11-pixel spacing appears to be 12 pixels because the white borders on the lower and right edges of the button components are not visually significant.)"
- They even worry about 1 pixel!

Layout process

- (4) Create a design grid, using the number and width of the rest of the components to decide the number of columns and the column width for the grid.
  - design grid = rows & columns to structure design
    - we're not talking a strict grid
    - we're not talking GridLayout's
    - essentially, we're talking guides to help with horizontal and vertical alignment

Layout process

- (4) Continued...
  - design grid example for "Find" box



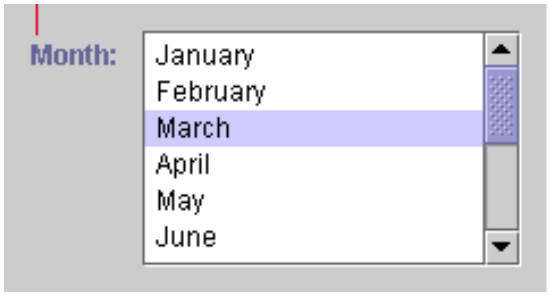
  - guidelines suggest that grid derives from command buttons and spacing
  - we won't abide by this — but having a grid roughly this size is a good idea
    - that is, pick a grid and stick to it

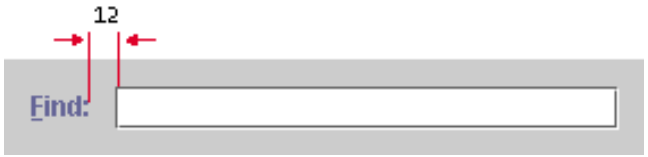
Layout process

- (5) Add the components to the dialog box in their logical sequence.
  - When designing a dialog box, place the most important options, or those you expect users to complete first, prior to others (in reading order).
- (6) Align related components using the design grid.
  - Align related components vertically using a chosen design grid column, and horizontally using a chosen design grid row.

Component specifics

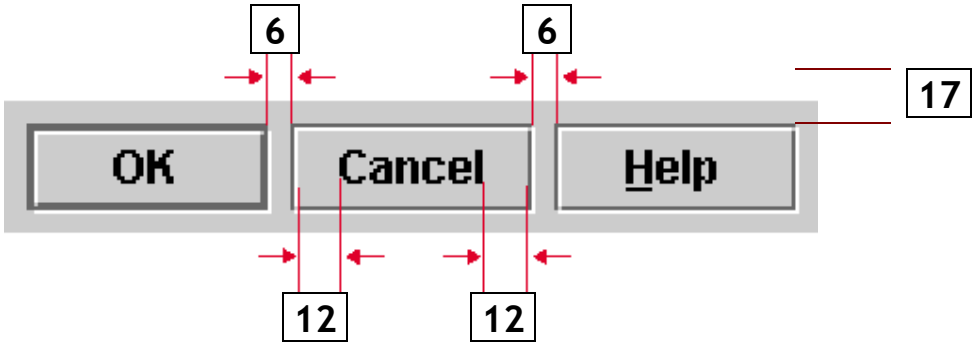
- Labels
  - Place labels before the component to which they refer (generally, above and to the left.)
  - Align labels with the tops of their components.
  - Insert 12 pixels between the trailing edge of a label and any associated components.
  - Left-align all vertically aligned labels.
    - Insert 12 pixels after the edge of the longest label.





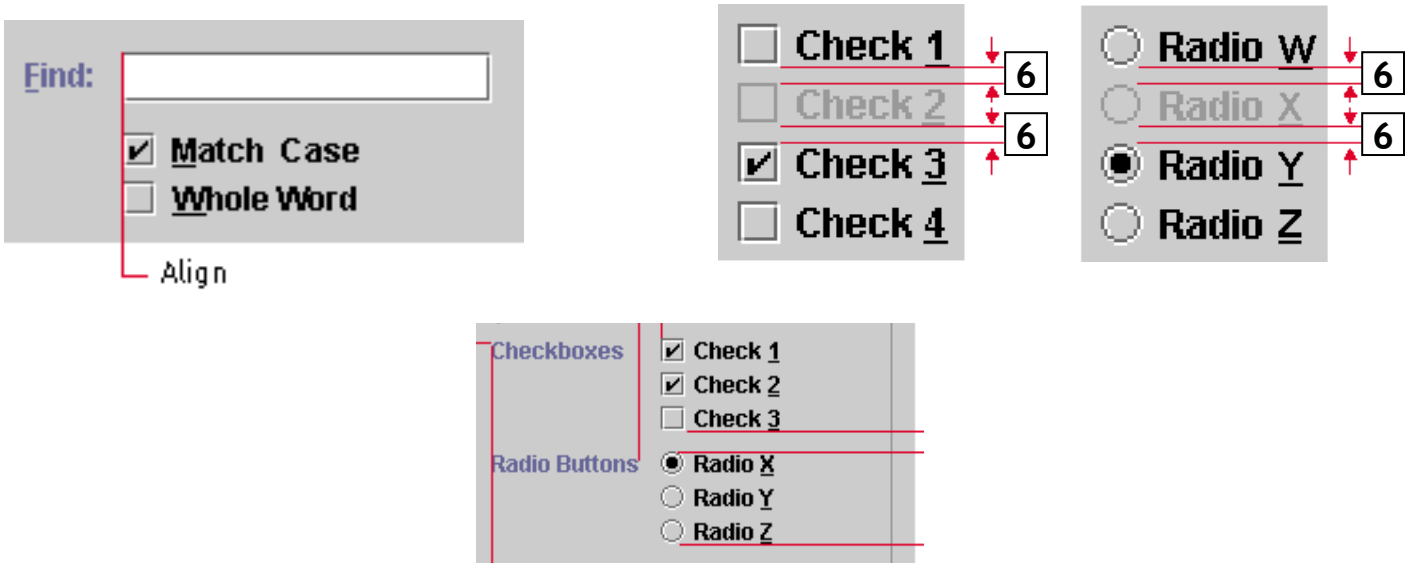
Component specifics

- Command buttons
  - Center all button text, with a 12-pixel left & right boundary between button label and button edge.
  - Make all grouped command buttons equal width, which doesn't change when window is resized.
  - Space buttons in a group 6 pixels apart.
  - Make the default button the left-most button.
  - Insert 17 pixels between these & rest of window.



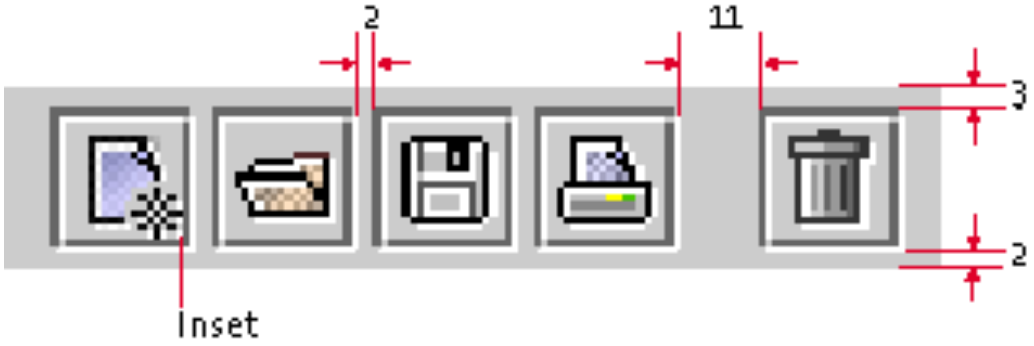
Component specifics

- Radio & check-box buttons
  - Align the leading edge with that of other components.
  - Space grouped buttons 6 pixels apart.
  - Space separate groups 12 pixels apart.



Component specifics

- Toolbar buttons
  - Space individual toolbar buttons 2 pixels apart.
  - Space groups of toolbar buttons 11 pixels apart.
  - Include 3 pixels of space above and below all toolbar buttons.



Component specifics

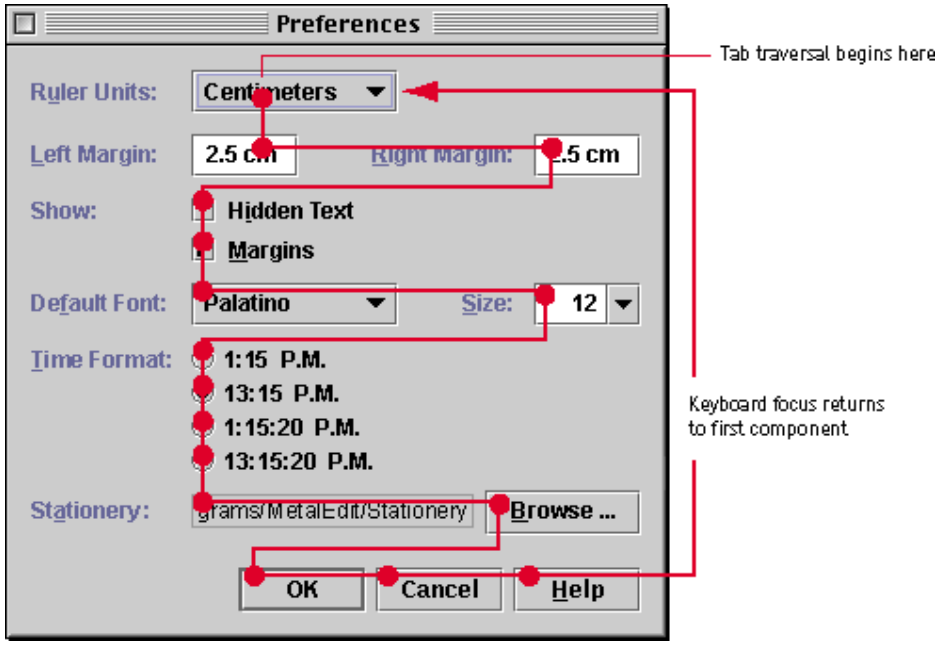
- Interface text
  - Most components use "headline" capitalization.
    - [capitalize every word except articles, conjunctions, etc.]
    - Checkbox text (Automatic Save Every Five Minutes)
    - Combo box text (Centimeters)
    - Command button text (Don't Save)
    - Icon names (Trash Can)
    - Labels for buttons or controls (New Contribution To:)
    - Menu items (Save As...)
    - Menu titles (View)
    - Radio button text (Start at Top)
    - Titles of windows and dialogs (Color Chooser)
    - Tool tips (Cut Selection)

Component specifics

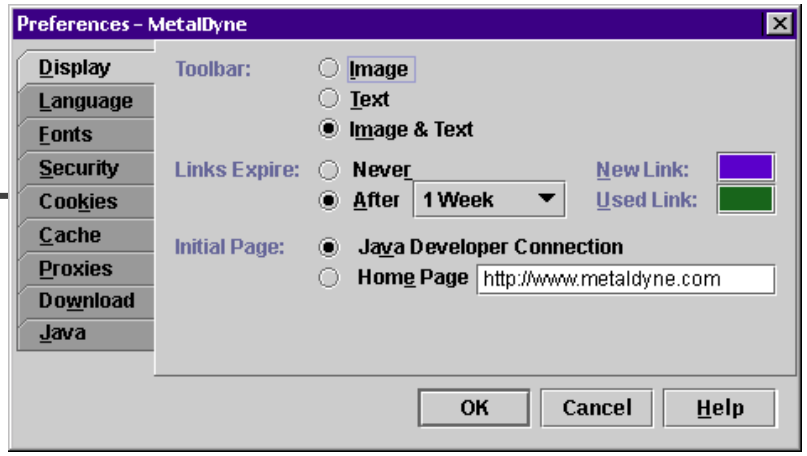
- Interface text (continued)
  - Alerts, errors, and long labels use "sentence" capitalization.
    - [capitalize only the first word in the sentence]
    - Alert box text
      ("The document you are closing has unsaved changes.")
    - Error or help messages
      ("The printer is out of paper.")
    - Labels that indicate changes in status
      ("Operation is 75% complete.")

Dialog box design

- Tab traversal order
  - Specify a logical tab traversal order for the user interface elements in a dialog box.
  - The default traversal order is the sequence in which you add the components to the dialog box.
    - do it this way, it's easiest
    - but there's also the method setNextFocusable-Component()

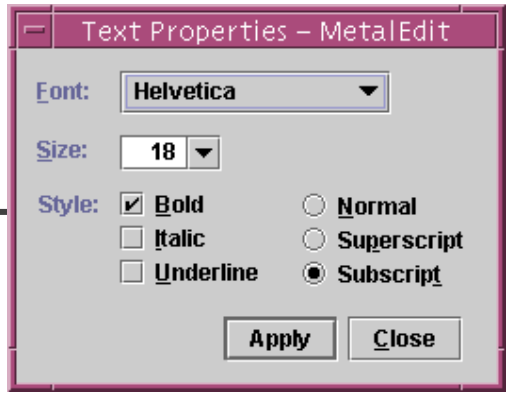


Dialog buttons



- "OK" and "Cancel" buttons
  - best for *single-use* dialogs, modal or modeless
  - in most cases, "OK" is the default button
  - When users click "OK", save the settings (or carry out the commands) and close the dialog.
    - Whenever possible, provide the button with a command name that describes the action (such as Print or Find) instead of OK.
  - When users click "Cancel", close the window and restore the settings to the state they were in when the dialog box was opened.

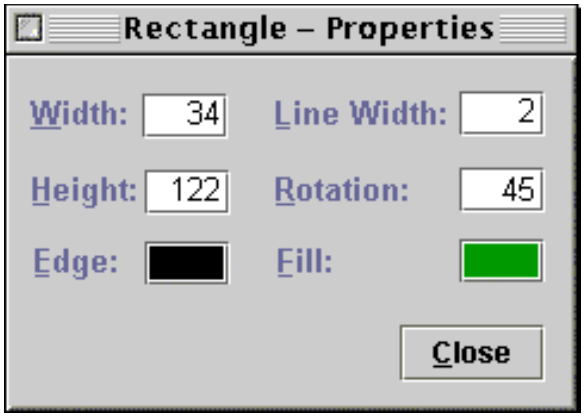
Dialog buttons



- "Apply" and "Close" buttons
  - best for *multiple-use*, modeless dialogs
  - in most cases, "Apply" is the default button
  - Use "Apply" to carry out the changes users specify in a dialog box without closing the window.
    - Whenever possible, in use a specific action name (such as "Find") instead of "Apply".
  - Include a "Close" button in a dialog box with an "Apply" (or other action) button. Close dismisses the dialog box without applying changes.
  - If the user has made changes in a dialog box and clicks Close before clicking the Apply button, display a Warning alert box to apply changes, discard changes, or cancel the close operation.

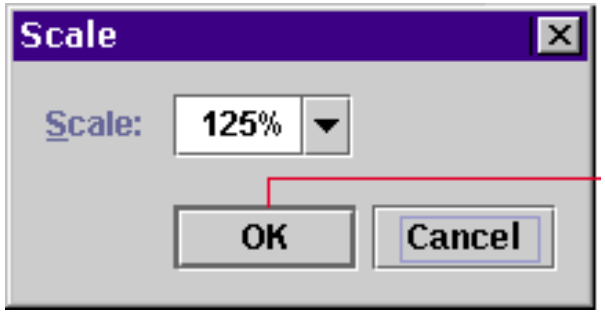
Dialog buttons

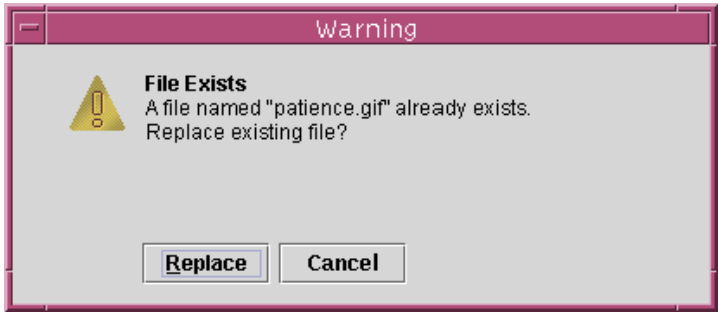
- "Close" button
  - used to dismiss informative secondary windows
  - Never use an "OK" button in a window that has a "Close" button.
  - When users click the "Close" button, dismiss the dialog box and do not make additional changes to the system.



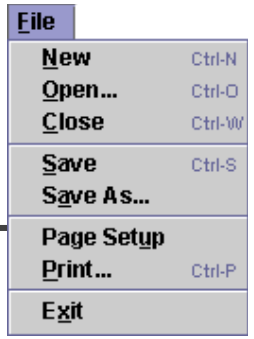
Dialog command buttons

- Default button
  - represents the most likely user action
  - If a dialog box has a default button, make it the first (left-most) command button in the group.
  - Commands that cause the user to lose data should never be the default button, even if they are the most common option.





Common menus



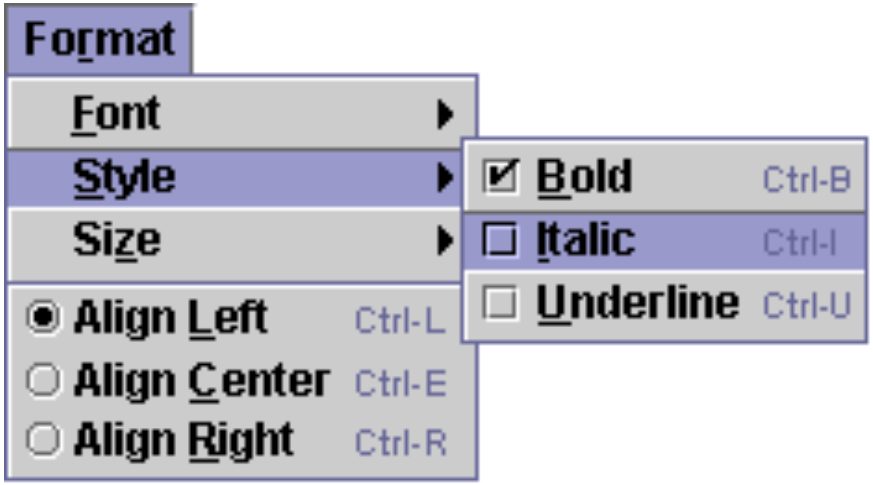
- File Menu
  - Place commands that apply to the document or the main object (or the application as a whole) in the File menu.
  - If your application manipulates "non-file" objects, give the File menu a different name.
    - e.g., "Project", "Mailbox"
  - When the Close item dismisses the active window, close any dependent windows at the same time.
  - Provide an Exit item, which closes all associated windows and terminates the application.
    - Be sure to use Exit, not Quit! (by convention)

Common menus

- Edit Menu
  - Place commands that modify the contents of documents or other data in the Edit menu.

Common menus

- Format Menu
  - Place commands that change the format of objects, usually text, in the Format menu.



Common menus

- View Menu
  - Place commands that change the view of the data in the View menu.
  - Ensure that commands in the View menu alter only the view/presentation of the underlying data without changing the data directly.

Common menus

- Help Menu
  - Place access to online information about the features of the application in the Help menu.
  - Include "About <app-name>" as the last item and place a separator before the item.
    - "About <app-name>" should display a window with the product name, version number, company logo, product logo, legal notices, and names of contributors.

That's it!

- The key to guidelines isn't which one you use… it's picking one and sticking with it
- The Apple and Java Swing guidelines come the closest to what we mean here
  - Android is another we haven't covered
- Although Bootstrap et al. unfortunately don't have guidelines as rigorous as these, these frameworks — and CSS more generally — help you to define a look & feel and stick with it