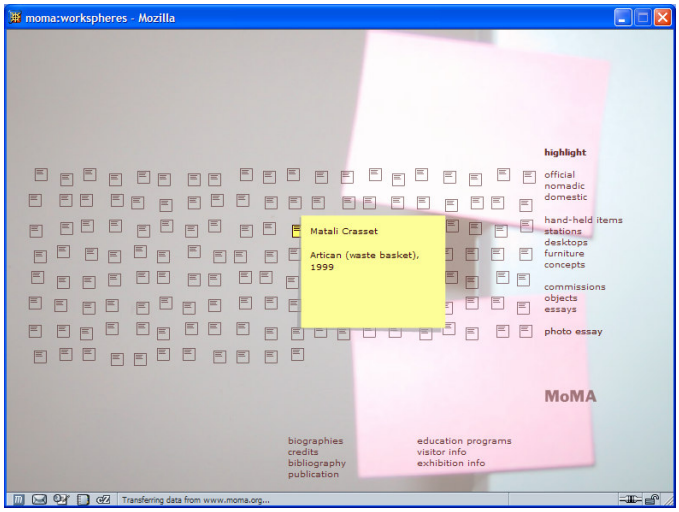
Lecture 10: Constraints and Layout

UI Hall of Fame or Shame?



Source: Vishy Venugopalan

This Flash-driven web site is the Museum of Modern Art's Workspheres exhibition, a collection of objects related to the modern workplace. This is its main menu: an array of identical icons. Mousing over any icon makes its label appear (the yellow note shown), and clicking brings up a picture of the object.

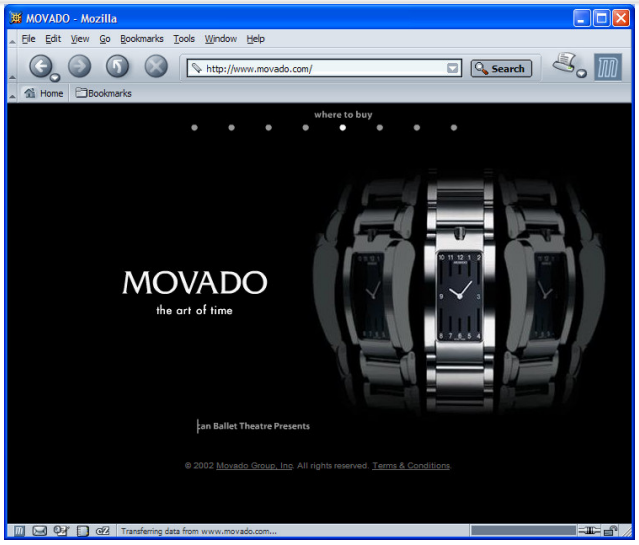
Clearly there's a **metaphor** in play here: the interface represents a wall covered with Post-it notes, and you can zoom in on any one of them.

We can praise this site for at least one reason: incredible **simplicity**. The designer of this site was clearly striving for aesthetic appeal. Nothing unnecessary was included. Note the use of whitespace to group the list of categories on the right, and the simple heading *highlight* that gives a clue to the function of the list (clicking on a category name highlights all the icons in that category).

Unfortunately, too much that was necessary was left out. Without any visible differentiation between the icons, finding something requires a lot of mouse waving. "Mystery navigation" was the term used by Vishy Venugopalan, who nominated this candidate for the UI Hall of Shame. It's hard enough to skim the display for interesting objects to look at. But imagine trying to find an object you've seen before. It's like that old card game Concentration, demanding too much **recall** from the user, rather than offering easy opportunities to **recognize** what you're looking for.

Frankly, if real Post-it notes were arranged on a wall like this, you'd probably have just as much trouble navigating it. So the choice of metaphor may be the essence of the problem.

More “Mystery Navigation”



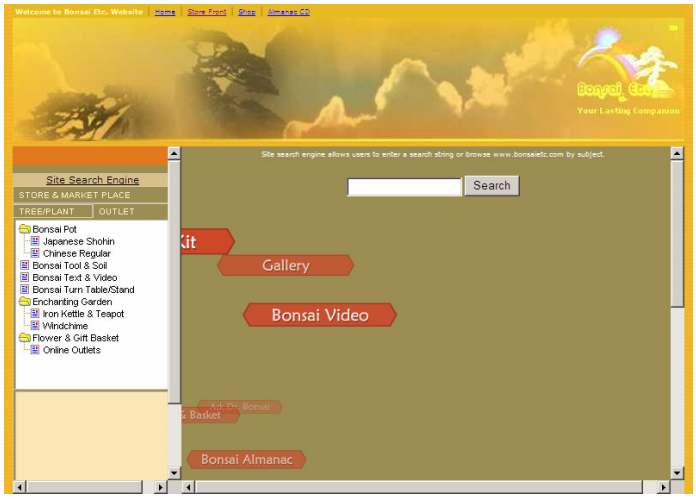
Source: Adam Champy

This is the home page for Movado, a company that makes expensive, stylish watches. The little white dots at the top of the window are menu options. If you watched the opening animation that precedes this screen, you’d see each menu label appear briefly over each dot. But if you skipped over the intro, you wouldn’t see that, and you may not even realize that a menu is hiding up there under those stylish white dots.

When you mouse over a dot, you actually have to wait for a cute little animation (a watch hand sweeping around the dot) before the menu label appears. Each little animation takes 2 seconds. So scanning the entire menu to look at all the options takes 16 seconds!

Clearly this is even worse than MOMA’s approach, since it starts with an invisible menu interface and makes it **inefficient** to boot. More tellingly, MOMA only cares about your eyeballs, but Movado actually wants to sell you a watch. If you can’t figure out their menu, or lose patience with it, you may be headed elsewhere.

Let's Play a Menu Game



Source: Anson Tsai

Here's our last entry: Bonsai Etc, a website that sells bonsai trees and equipment. In this site's Flash animation, the menu options **move**: some horizontally, some vertically. Worst of all, their paths overlap, so the items pass each other on the screen. At least they're labeled. It's a fun game, for a little while. But if you have a serious reason for visiting this web site – say, spending some money – do you really want to chase down every menu option you want to click?

One lesson you might draw from these examples is that Flash animation is bad, but that's too simplistic. Flash is a powerful tool that can be used for good or ill.

A better lesson might be that aesthetic appeal does not automatically confer usability. Effective graphic design is an important element of usability, but it isn't the whole story by any means.

Today's Topics

- Automatic layout
- Constraints

Layout

- Determining the positions and sizes of graphical objects

Layout Ranges in Difficulty

- Fixed constants
  - Many Windows dialog boxes
- Directly computable from model
  - Checkerboard from PS2/PS3
- One pass algorithm
  - Java layout managers, HTML tables
- Dynamic programming
  - paragraph flow with hyphenation
- Nonlinear optimization
- NP-hard
  - Graph with fewest edge crossings

Reasons to Do Layout Automatically

- Window resizing
- Screen resolution
- Font changes
- Widget changes
- Internationalization

Layout Managers

- Also called geometry managers (Tk, Motif)
- Abstract
  - Represents a bundle of constraint equations
- Local
  - Involve only the children of one container in the view hierarchy

Layout Propagation Algorithm

- layout(Container parent, Rectangle parentSize)
  - for each child in parent,
    - get child's size request
  - apply layout constraints to fit children into parentSize
  - for each child,
    - set child's size and position

Kinds of Layout Managers

- Packing
  - one dimensional
  - Tk: pack
  - Java: BorderLayout, FlowLayout, BoxLayout
- Gridding
  - two dimensional
  - Tk: grid
  - Java: GridLayout, GridBagLayout, TableLayout
- General
  - Java: SpringLayout

Important Concepts

- Anchoring
- Expanding vs. padding
- Invisible components
  - Struts
  - Glue
  - Springs
- Nested containers

Hints for Layout

- Use packing layouts when alignments are 1D
  - borders for top-level
  - nested boxes for internal
- Reserve gridding layouts for 2D alignment
  - unfortunately common when fields have captions!
  - TableLayout is easier than GridBag

Constraints

- Constraint: relationship expressed by the programmer and automatically maintained by the UI toolkit
- Uses
  - Layout
    - field.left = label.right + 10
  - Value propagation
    - deleteAction.enabled = (selection != null)
  - Synchronization of views to models
  - Interaction
    - rect.corner = mouse

One-Way Constraints

- Also called formulas, after spreadsheet
  - y = f(x1, x2, x3, ...)
  - Y depends on (points to) x1, x2, x3, ...
- Algorithms
  - Data-driven
    - Reevaluate formulas when a value is changed
  - Demand-driven
    - Reevaluate formulas whenever a value is requested
  - Lazy
    - When dependent value changes, **invalidate** all values that depend on it
    - When invalid value is requested, **recalculate** it

Variants

- Multi-output formulas
  - (y1, y2, ...) = f (x1, x2, x3, ...)
- Cyclic dependencies
  - Detect cycles and break them
- Constraint hierarchies
  - Some constraints stronger than others
- Side effects
  - If f has side effects, when do they happen?
    - Lazy evaluation makes side effects unpredictable
  - Amulet: eager evaluation

Multiway Constraints

- Each constraint is a multivariate relationship
  - rect.right = rect.left + rect.width − 1
  - Any variable may be used as target (different *method* for each target variable)
  - Planning step decides which variables to target

Variants

- Constraint hierarchy
  - Which value should be changed?
  - Each constraint has a priority
  - "Stay" constraints (highest priority) are used for constants
- Inequalities
  - Label.right <= field.left