Lecture 17:
Constraints and Data Bindings, 1

05-431/631 Software Structures for User Interfaces (SSUI)

Fall, 2021

Happy Halloween!

- Take 2 candies!

Logistics

- Last lecture audio didn’t work at all ☹ – sorry!
  - Let me know if you need it fixed.
- Try narrow format for slides, so don’t have to worry about the video window

Constraints

- Relationships defined once and maintained by the system
- Useful for keeping parts of the graphics together.
- Also for passing values around
- Typically expressed as arithmetic or code relationships among variables.
  - Variables are often the properties of objects (left, color)
- Types:
  - "Dataflow" constraints; Choices:
    - Single-Output vs. Multi-output
    - Types: One-way, Multi-way, Simultaneous equations, Incremental, Special purpose
    - Cycles: supported or not
  - Others: AI systems, scheduling systems, etc.

Historical Note: “Active Values”

- Old Lisp systems had active values
  - Attach procedures to be called when changed
- Similar to today’s “Listeners” or “Observer pattern”
- Like the “inverse” of constraints
  - Procedures are attached to values which change instead of values where needed
  - Push vs. Pull
- Inefficient because all downstream values are re-evaluated, possibly many times
  - E.g., when x and y values change

Important Historical Constraint Systems

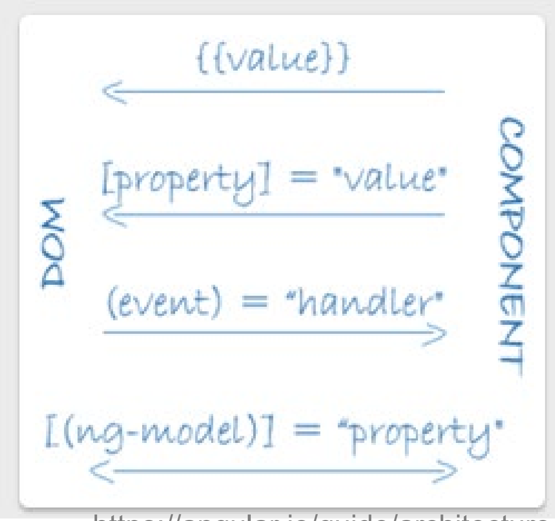
- Alan Borning's ThingLab (1979)
- Spreadsheets (~1979)
- Peridot (1987) (Myers)
- Garnet & Amulet (1989, 1994) (Myers)
  - Graphics *and* "data bindings"
- DeltaBlue (1990) (Freemen-Benson)
  - SkyBlue (1994) (Michael Sannella)
- subarctic (Hudson) (1991)
- Gleicher's (1993)
- …

Some Constraint Systems Today

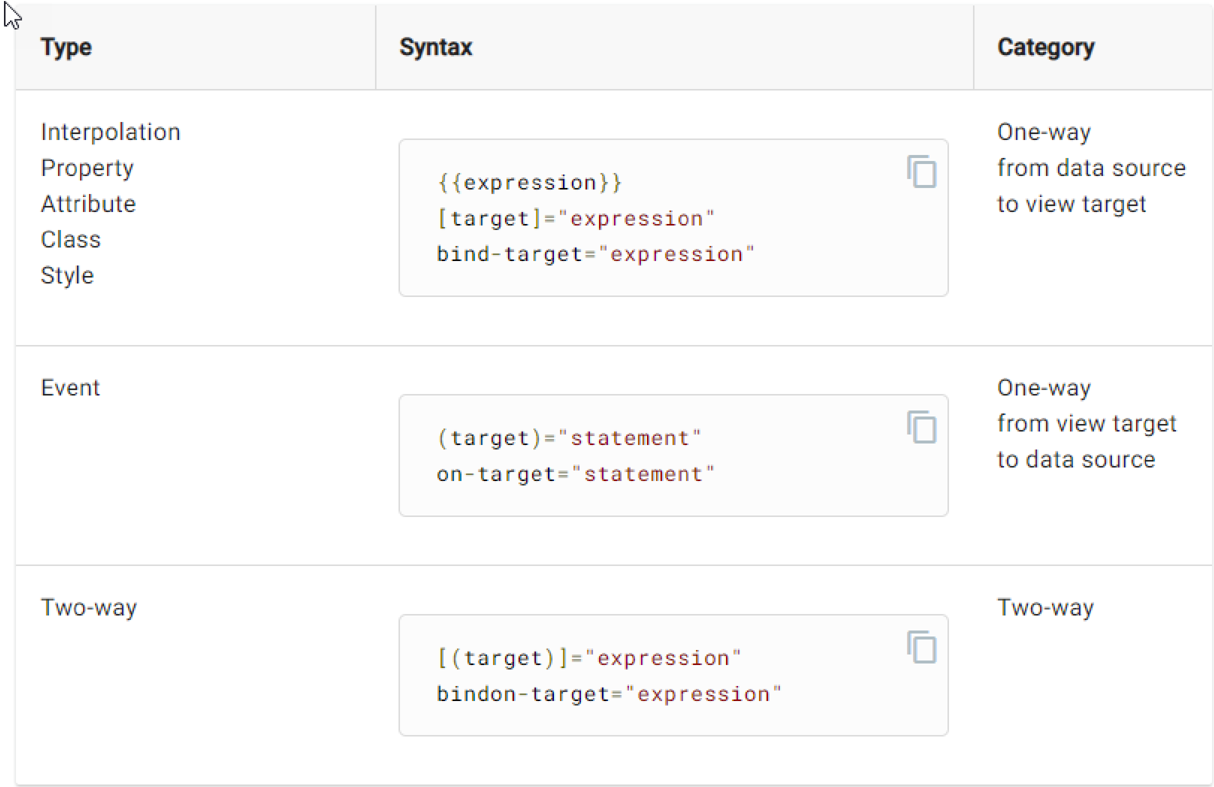
- Apple constraints for “Auto Layout”
- Toolkit and windows “layout managers”/”geometry managers” (lecture 10)
- “data bindings”
  - Adobe Flex, AngularJS
- Google’s AngularJS (before v2)
- Most AutoDesk (CAD) products, e.g., Fusion 360 for 2D & geometric
- Ember. http://emberjs.com/
  - MVC, “Computed Values” of properties
- KnockoutJS. http://knockoutjs.com/
  - “Declarative Bindings”, “Dependency Tracking”
- Research: Stephen Oney’s ConstraintJS http://cjs.from.so/ (2012)

Angular Data Bindings

- Tie DOM properties to other values
- Can be one-way or two-way
  - Use [] to bind from source to view.
  - Use () to bind from view to source.
  - Use [()] to bind in a two way sequence of view to source to view.



https://angular.io/guide/architecture-components#data-binding



Type	Syntax	Category

Interpolation
Property
Attribute
Class
Style
	`{{expression}}`
	`[target]="expression"`
	`bind-target="expression"`
		One-way from data source to view target

Event	`(target)="statement"`
	`on-target="statement"`
		One-way from view target to data source

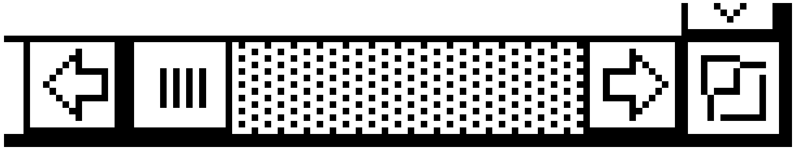
Two-way	`[(target)]="expression"`
	`bindon-target="expression"`
		Two-way

https://angular.io/guide/binding-syntax

One Way Constraints

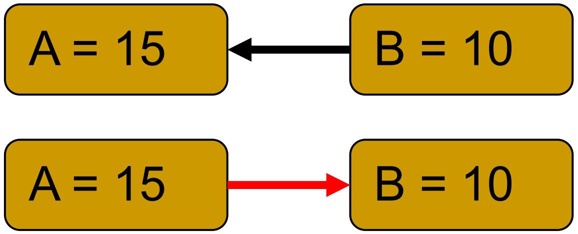
- Simplest form of constraints
- D = F(I1, I2, ... In)
- Often called *formulas* since like spreadsheets
- Can be other dependencies on D

CurrentSliderVal = mouse.X - scrollbar.left
scrollbar.left = window.left + 200
scrollbar.visible = window.has_focus



Data flow graph

- Nodes for variables (values) grouped into objects
- Lines for data flow for the constraints
  - Reverse direction of lines for "dependencies"
  - E.g., A = B+5
    - B's value flows to A
    - A's value depends on B
- Often need back-pointers too to clean up when change

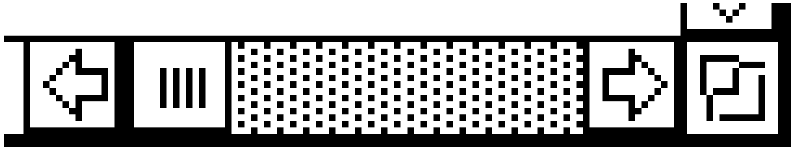


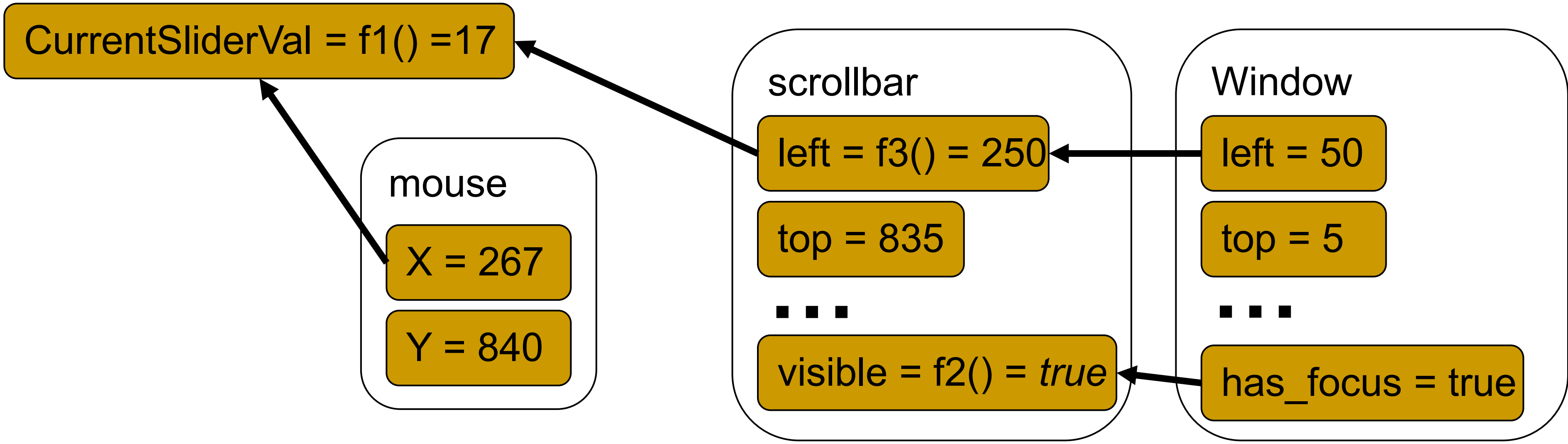
One Way Constraints

CurrentSliderVal = mouse.X - scrollbar.left

scrollbar.left = window.left + 200

scrollbar.visible = window.has_focus





CurrentSliderVal = f1() =17

mouse

X = 267

Y = 840

scrollbar

left = f3() = 250

top = 835

...

visible = f2() = *true*

Window

left = 50

top = 5

...

has_focus = true

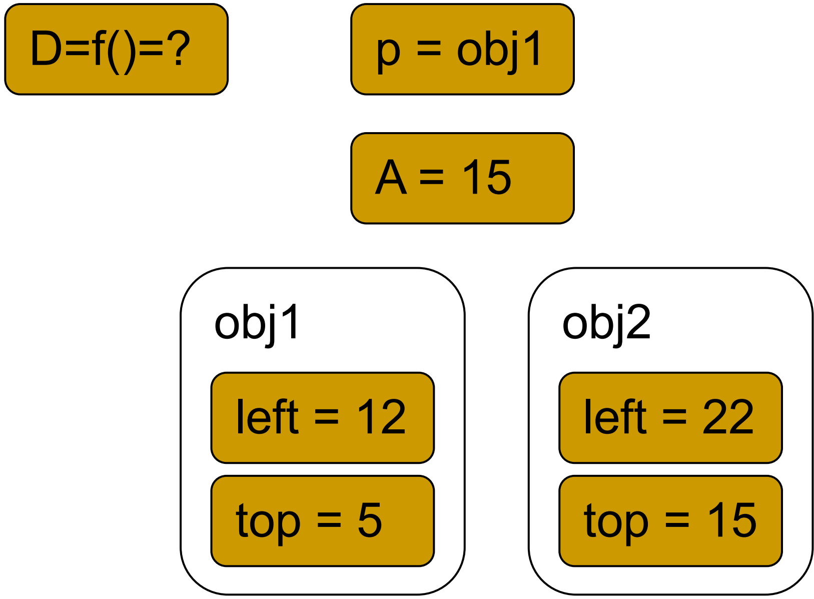
One Way Constraints, cont.

- Not just for numbers: mycolor = x.color
- Implementations:
  1. Just re-evaluate all required equations every time a value is requested
     - least storage, least overhead
     - Equations may be re-evaluated many times when not changed. (e.g, scrollbar.left when mouse moves)
     - cycles:
file_position = F1(scrollbar.Val)
scrollbar.Val = F2(file_position)
     - Objects may jitter – change X and then change Y
     - Cannot detect when values change (to optimize redraw)
  2. More efficient algorithms next lecture

Garnet / Amulet Constraint Solving

- Default: one-way, data flow constraints with variables in the dependencies, support for cycles, and multiple changes before solving
  - Efficient enough for ubiquitous use
  - Garnet text button widget contained 43 constraints internally, and the Lapidary graphical interface builder contained 16,700 constraints
- Also can bring in alternative solvers
  - Brad Vander Zanden’s multi-way solver [Vander Zanden 1996]
  - “Animation Constraints” [Myers 1996]
- Snippets of video for Garnet and Amulet constraints

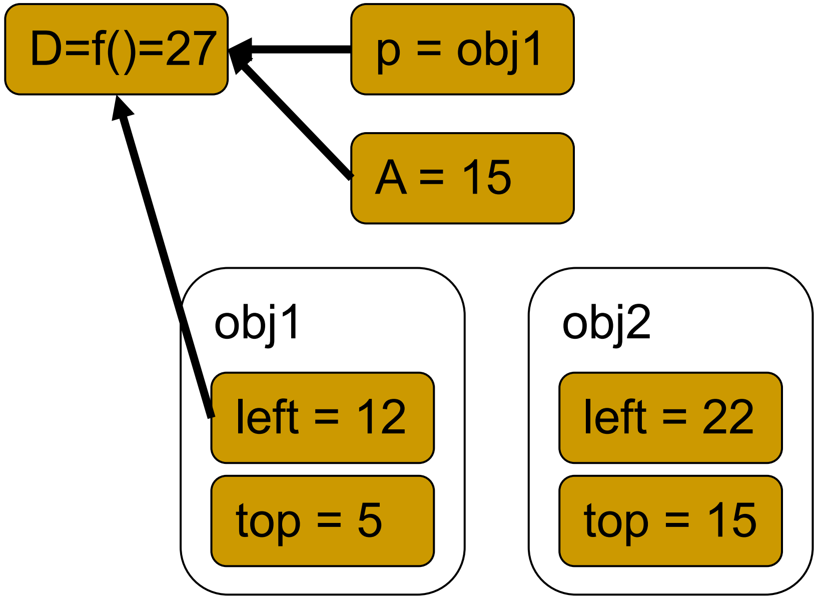
Garnet / Amulet Default Algorithm



- Variables in the dependencies
  - Example: D = p^.left + A
  - Important innovation in Garnet we invented, now ubiquitous
  - Supports feedback objects
    - outlineRect.left = selectedObject^.left …

      circle1.object_over = rect34
      circle1.left = self.object_over.right + 10
  - Supports loops: D = Max(components^)
  - Only evaluates needed part of conditionals
    width = if otherpart.value > tolerance
       then *expensive computation*
       else otherpart.width
  - Requires the dependencies be dynamically determined

Garnet / Amulet Default Algorithm

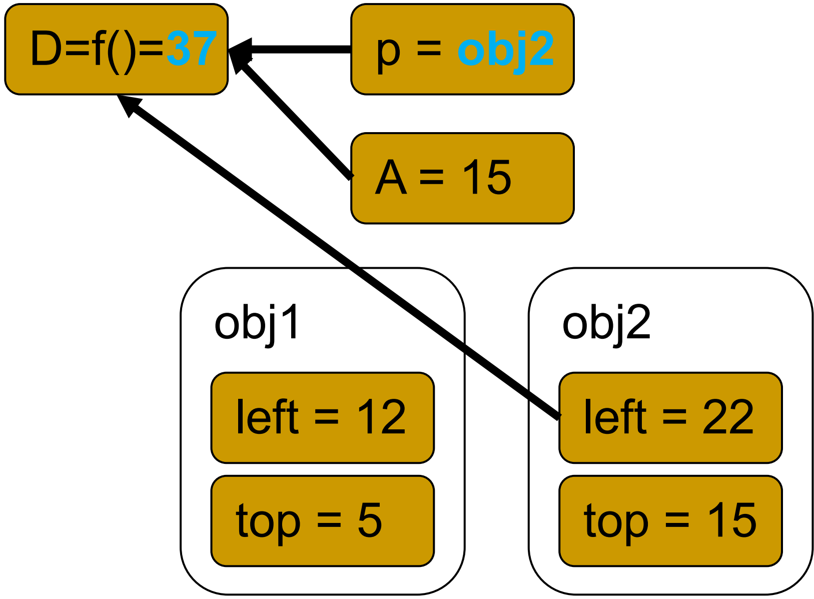


- Variables in the dependencies
  - Example: D = p^.left + A
  - Important innovation in Garnet we invented, now ubiquitous
  - Supports feedback objects
    - outlineRect.left = selectedObject^.left …

      circle1.object_over = rect34
      circle1.left = self.object_over.right + 10
  - Supports loops: D = Max(components^)
  - Only evaluates needed part of conditionals
    width = if otherpart.value > tolerance
       then *expensive computation*
       else otherpart.width
  - Requires the dependencies be dynamically determined

© 2021 Brad Myers

Garnet / Amulet Default Algorithm



- Variables in the dependencies
  - Example: D = p^.left + A
  - Important innovation in Garnet we invented, now ubiquitous
  - Supports feedback objects
    - outlineRect.left = selectedObject^.left …

      circle1.object_over = rect34
      circle1.left = self.object_over.right + 10
  - Supports loops: D = Max(components^)
  - Only evaluates needed part of conditionals
    width = if otherpart.value > tolerance
      then *expensive computation*
      else otherpart.width
  - Requires the dependencies be dynamically determined

Examples of Expressing Constraints

- Garnet:

```
(create-instance NIL opal:line
        (:points '(340 318 365 358))
        (:grow-p T)
        (:x1 (o-formula (first (gvl :points))))
        (:y1 (o-formula (second (gvl :points))))
        (:x2 (o-formula (third (gvl :points))))
        (:y2 (o-formula (fourth (gvl :points)))))
```

- Amulet:

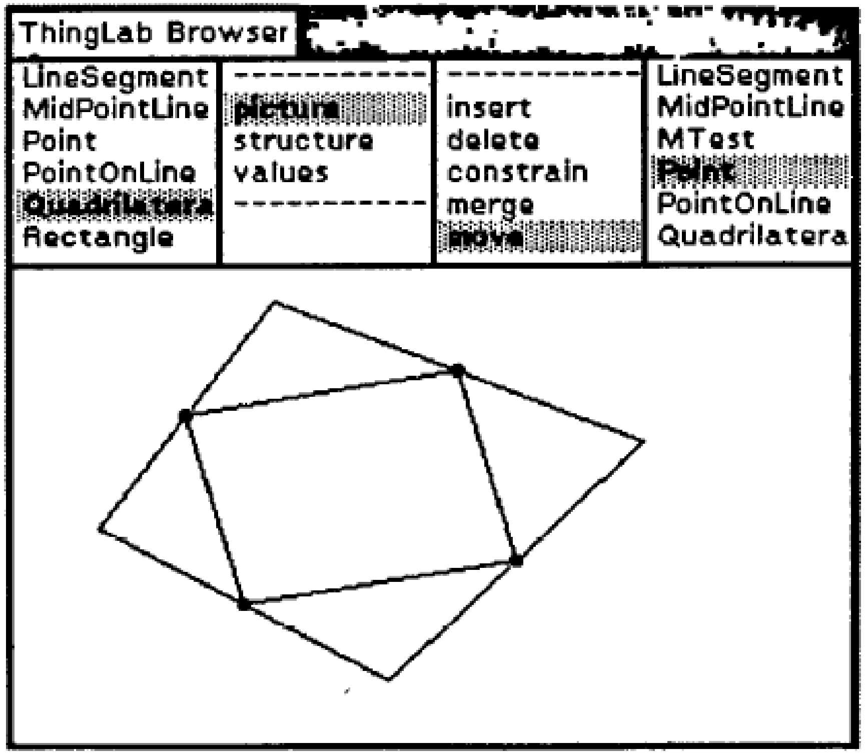
```
Am_Define_Formula (int, height_of_layout) {
  int h = (int)Am_Height_Of_Parts(self) + 2 *
((int)self.Get(Am_TOP_OFFSET));
  return h < 75 ? 75 : h;
}

am_empty_dialog = Am_Window.Create("empty_dialog_window")
    .Set (Am_LEFT_OFFSET, 5) // used in width_of_layout
    .Set (Am_TOP_OFFSET, 5) // used in height_of_layout
    .Set (Am_WIDTH, width_of_layout)
    .Set (Am_HEIGHT, height_of_layout)
        ...
```

Other One-Way Variations

- Multiple outputs
  - (D1, D2, ... Dm) = F(I1, I2, ... In)
- Side-effects in the formulas
  - useful for creating objects
  - when happen?
  - what if create new objects with new constraints
  - cycles cannot be detected
- Constant formula elimination
  - To decrease the size used by constraints

Two-Way (Multi-way) Constraints

- From ThingLab (~1979)
  - Alan Borning. “Defining Constraints Graphically,” *Human Factors in Computing Systems. Boston, MA, Apr, 1986. pp. 137-143. Proceedings SIGCHI'86.*
- Constraints are expressions with multiple variables
- Any may be modified to get the right values
- Example: A.right = A.left + A.width - 1
- Often requires programmer to provide methods for solving the constraint in each direction:
  A.left = A.right - A.width + 1
  A.width = A.right - A.left + 1
- Useful if mouse expressed as a constraint



Two-Way implementations

- Requires a *planning* step to decide which way to solve
  - Many systems compute plans and save them around since usually change same variable repeatedly
- In general, have a graph of dependencies, find a path through the graph
- How control which direction is solved?
  CurrentSliderVal = mouseX - scrollbar.left
  - "Constraint hierarchies" = priorities
    - constants, interaction use "stay" constraints with high priority
  - Dynamically add and remove constraints
- Brad Vander Zanden's "QuickPlan" solver
  - Handles multi-output, multi-way cyclic constraints in $O(n^2)$ time instead of exponential like previous algorithms

Simultaneous Equations

- Required for parallel, perpendicular lines; tangency, etc.
- Also for aggregate's size
- Numerical (relaxation) or symbolic techniques
  - Thinglab bridge (1979) (cite)

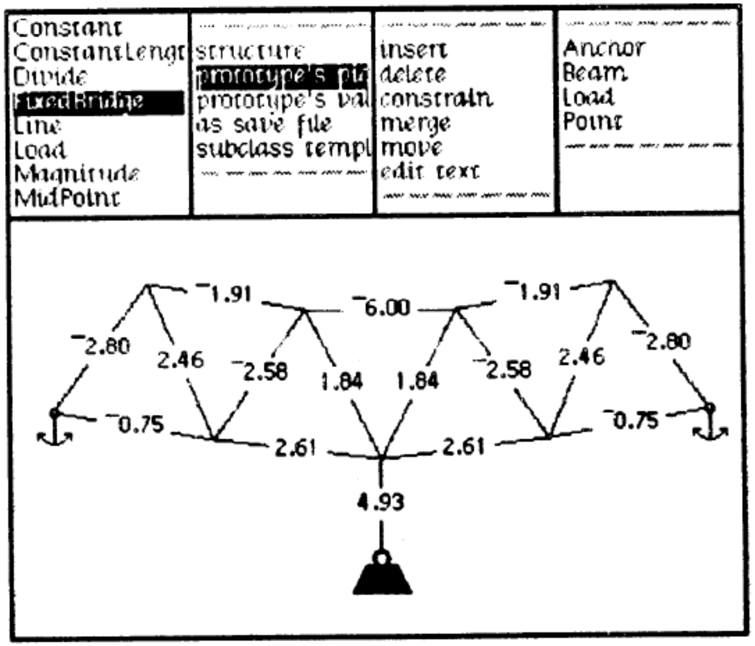
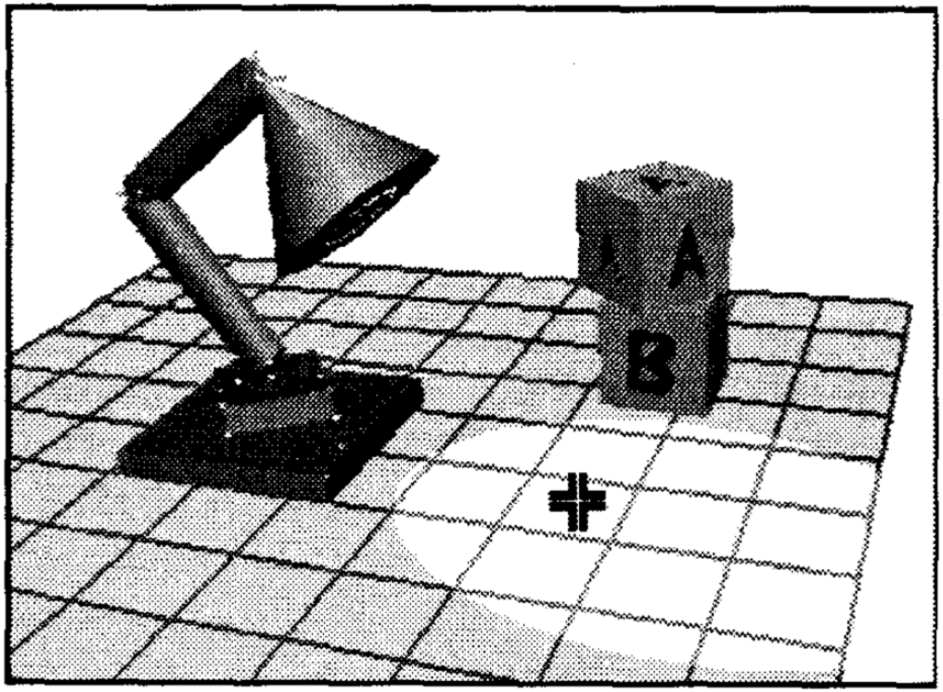


Figure 2.31 - A bridge under load

Incremental

- Michael Gleicher's PhD thesis, 1994
- Only express forward computations
- Tries to get reverse by incrementally changing the forward computation in the right direction using derivatives.
- Supports interactions otherwise not possible
- Produces smooth animations



Animation Constraints in Amulet

- Implemented using Amulet's constraint mechanism
- When slot set with a new value, restores old value, and animates from old to new value
- Usually, linear interpolation
- For colors, through either HSV or RGB space
- For visibility, various special effects between TRUE and FALSE
- *Demo*