GUI Builders & No-Code

CS 695 / SWE 699: Programming Tools

Fall 2023

GEORGE MASON UNIVERSITY

Today

- Part 1 (Lecture)(~60 mins)
  - 10 min break!
- Part 2: Tech Talk - Microsoft Power Automate (15 mins)
- Part 3: (In-Class Activity)(60 mins)

Logistics

- HW 2 due today
- No class next week
- HW 3 due in 3 weeks

Overview

- Visual Programming
- GUI Builders

Visual Programming

Definitions

"Visual Programming Language"

"Any system where the user writes a program using two or more dimensions"
[Myers, 1990]

"A visual language manipulates visual information or supports visual interaction, or allows programming with visual expressions"
[Golin , 1990]

"A programming language that lets users create programs by manipulating program elements graphically rather than by specifying them textually".

"A set of spatial arrangements of text-graphic symbols with a semantic interpretation that is used in carrying out communication actions in the world".
[Lakin, 1989]

Motivation from Psychology

Language determines thought and that linguistic categories limit and determine cognitive categories [1]

In longer sentences meaning of each word may be clear, but the way in which they are strung together makes little sense imposes a tremendous mental workload to understand. [2]

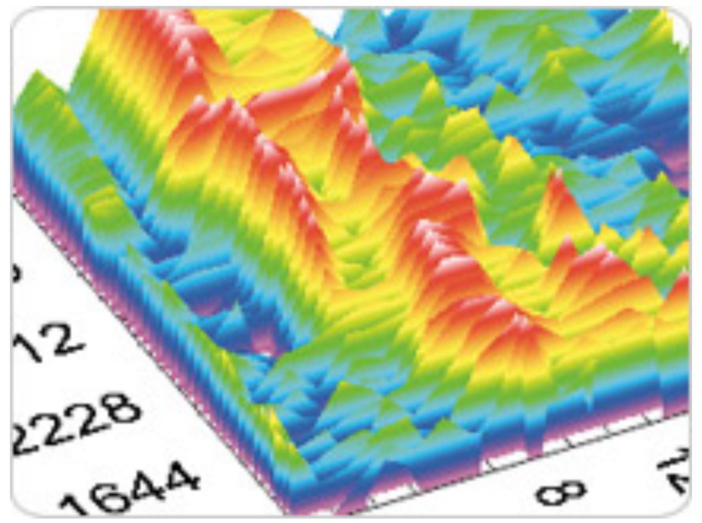
Most design tasks require 3 cognitive skills: search, recognition and inference.
Diverse set of views (and studies) exist today about whether VPLs aid in search or cognition. [3]

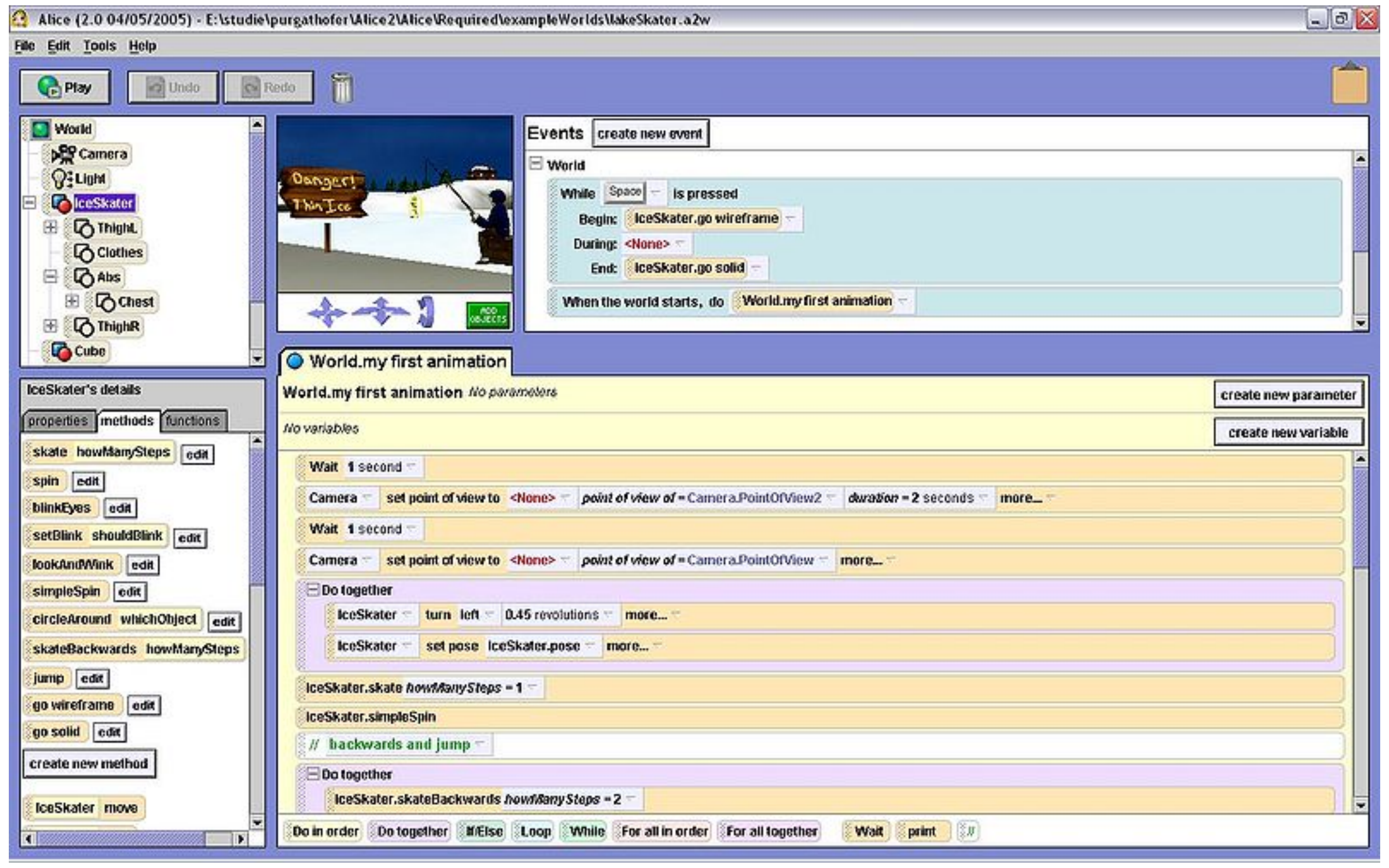
[1] Sapir, E. (1929): 'The Status of Linguistics as a Science'. In E. Sapir (1958): *Culture, Language and Personality* (ed. D. G. Mandelbaum). Berkeley, CA: University of California Press
[2] Christopher D. Wickens, “Engineering Psychology and Human Performance”, 3rd Edition
[3] J. H. Larkin and H. A. Simon. Why a diagram is (sometimes) worth ten thousand words. Cognitive Science, 11:65-99, 1987.

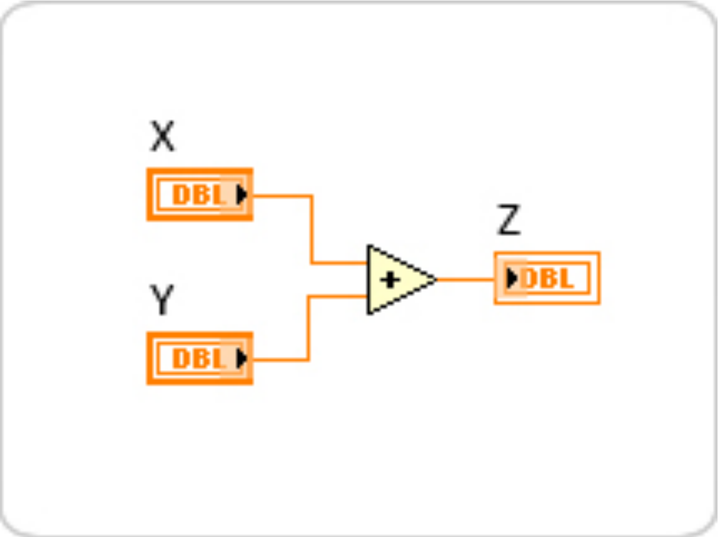
Motivation

Some applications are (believed to be) very well suited to graphical development approaches

- Scientific visualization
- Simulations
- User Interfaces
- Signal Processing
- Data Displays







(Claimed) Advantages of VPLs

- Fewer programming concepts
- Concreteness
- Explicit depiction of relationships
- Immediate visual feedback
- Parallel computation is a natural consequence of many visual programming paradigms

(Claimed) Disadvantages of VPLs

"Deutsch Limit" *

- The problem with visual programming is that you can't have more than 50 visual primitives on the screen at the same time.

Some situations in which text has superiority:

- Documentation,
- Naming to distinguish between elements that are of the same kind, and
- Expressing well-known and compact concepts that are inherently textual, e.g. algebraic formulas.

Visual Programming Techniques

- Concreteness: expressing some aspect of a program using instances
  - e.g., display the effects of computation on individual instance
- Directness: small distance between goal and actions required of the user to achieve goal
  - e.g., direct manipulation of object properties
- Explicitness: don't require inference to understand semantics
  - e.g., depict dataflow edges between variables
- Livenesss: offer automatic display of effects of program edits on output
  - e.g., after every edit, IDE reruns code and regenerates output

History of VPLs

1960

- AMBIT/G/L
- Grail
- GAL
- Graphical Program Editor
- Query by Example
- Pygmalion
- I/O Pairs

Techniques
- Graphs
- Flowcharts
- Flowchart derivatives
- FORMS
- Demonstrational

1980

- Action Graphics
- FORMAL
- ThingLab
- Hi-Visual
- LabView
- PROGRAPH
- PIGS
- Pict
- Rehearsal
- SmallStar
- Forms
- Editing by Example
- PICT
- Lotus 1-2-3
- SIL-ICON
- VisiCalc
- HiGraphs
- Miro
- StateMaster

Techniques
- Graphs
- Flowcharts
- Flowchart derivatives
- FORMS
- Demonstrational
- Data Flows
- Spreadsheets
- Matrices
- Jigsaw Puzzles
- Petri nets
- Flowchart derivatives

1990

- Cube
- Cantata
- SchemePaint
- CODE 2.0
- Iconicode
- MViews
- AVS
- Mondrian
- ChemTrains
- Vampire
- VIPR
- SPE

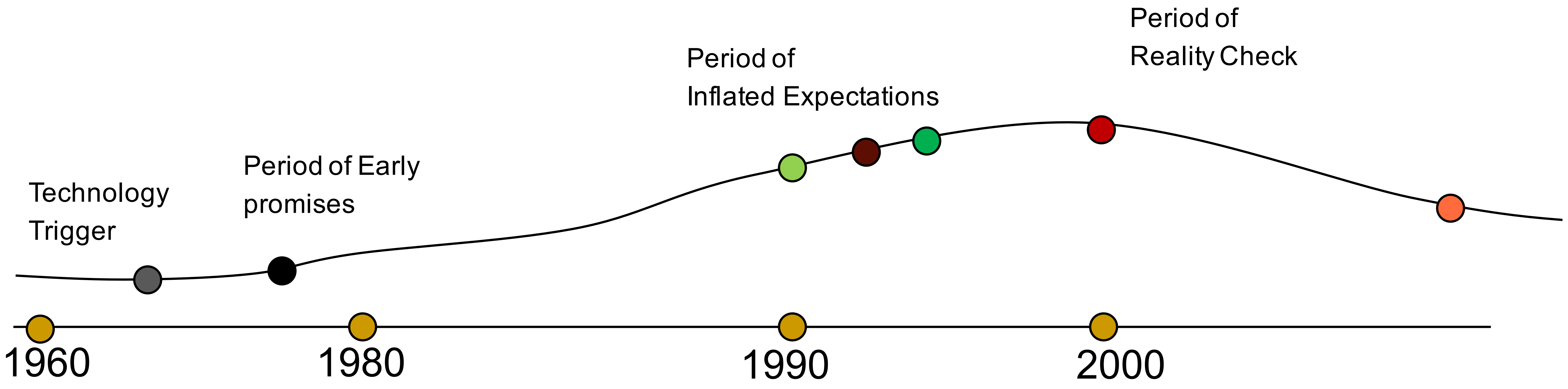
Techniques/Goals
- 3D Rendering
- Visual Hierarchy
- Procedures
- Control Structures
- Programmable Graphics
- Animations
- Video Imagery Exploitation
- General purpose, declarative language
- Audio, video and image processing
- Graphical models from behavioral models
- Learning and Cognitive abilities in vision processes
- Handling Scalability, typing, and imperative design
- Collaborative Software Development

2000

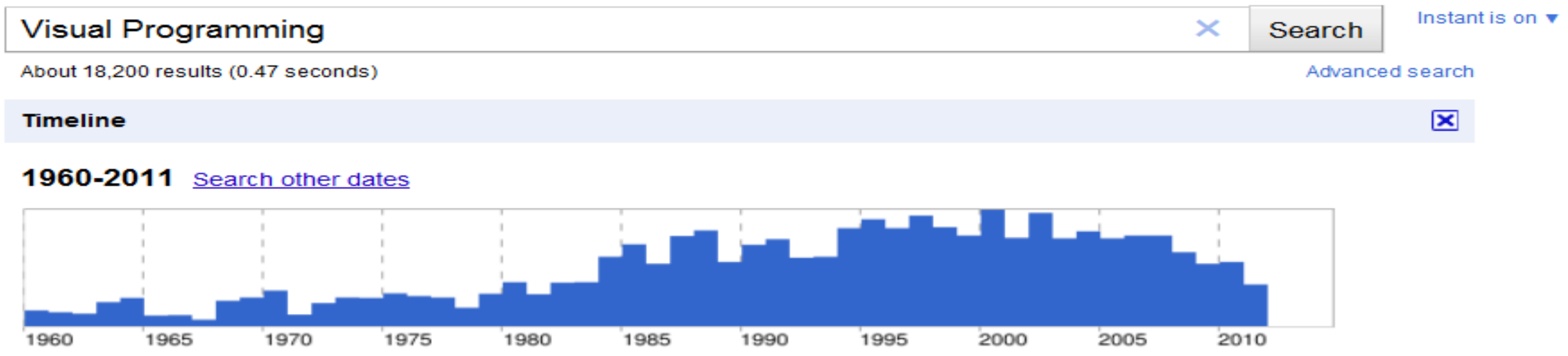
- LOFI/HIPI
- FOXQ
- VMQL
- GXL
- Euler View
- Yahoo Pipes
- Popfly

Techniques/Goals
- Child Learning
- Xquery by FORMS
- Spreadsheet Analysis
- Visual Model Query
- Layouts
- Specification and Interchange
- Mashups
- Web-based design
- Programming for end-users (2003) / non-Professionals

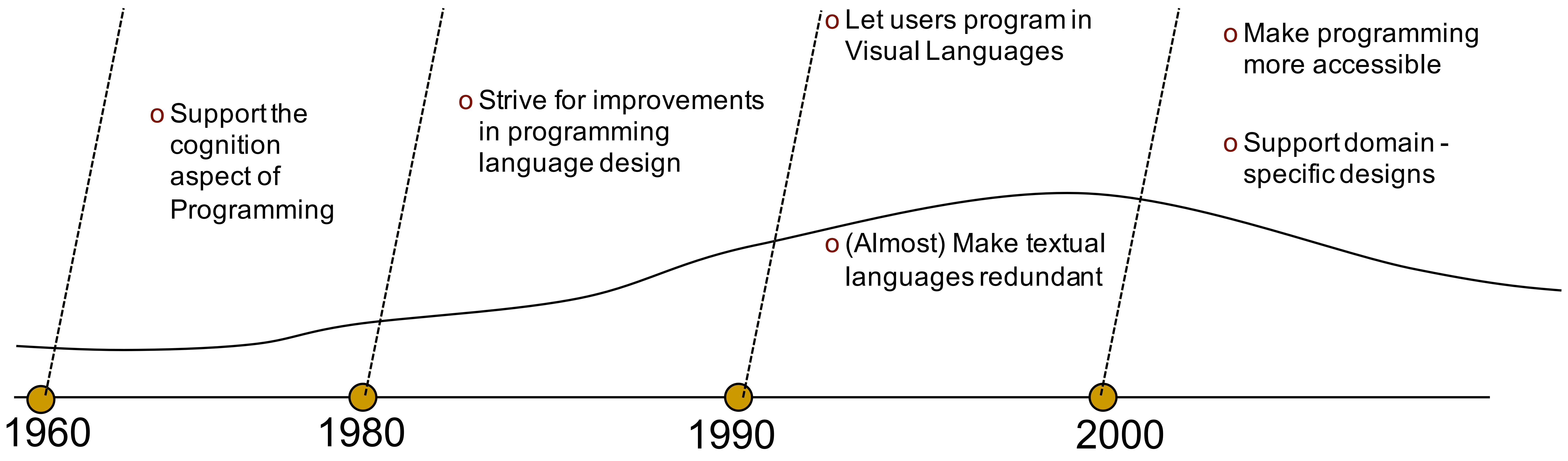
History of VPLs



- [**Ellis**, 1969] : GRAIL
- [**Smith**, 1975] : Pygmalion
- [**Myers, 1990**] : Taxonomies for VPL
- [**Repenning, 1992**] : Agent Sheet
- [**Burnett, 1994**] : Broad Classifications for VPL Research
- [**Kirsten N. Whitley, 1997**]: User Studies (for/against VPLs)
- [**MacLaurin, 2009**] : KODU



History of VPLs



Taxonomy of visual programming languages

Specification Technique:	Systems:
Textual Languages:	Pascal, Ada, Fortran, Lisp, Ada, etc. Tinker, Smallstar
Flowcharts:	Grail, Pict, FPL, IBGE, OPAL
Flowchart derivatives:	GAL, PIGS, SchemaCode, PLAY
Petri nets:	MOPS-2, VERDI
Data flow graphs:	Graphical Program Editor, PROGRAPH, Graphical Thinglab, Music System, HI-VISUAL, LabVIEW, Fabrik, InterCONS
Directed graphs:	AMBIT/G/L, State Transition UIMS, Bauer's Traces
Graph derivatives:	HiGraphs, Miro, StateMaster
Matrices:	ALEX, MPL
Jigsaw puzzle pieces:	Proc-BLOX
Forms:	Query by Example, FORMAL
Iconic Sentences:	SIL-ICON
Spreadsheets*:	VisiCalc, Lotus 1-2-3, Action Graphics, ‘‘Forms’’
Demonstrational*:	Pygmalion, Rehearsal World, Peridot
None*:	I/O Pairs, Editing by Example

Brad A. Myers. "Taxonomies of Visual Programming and Program Visualization," Journal of Visual Languages and Computing. vol. 1, no. 1. March, 1990. pp. 97-123.

Dataflow Program Representations

- Represent computation as a network
- Nodes correspond to components
- Edges correspond to data flow between components

Fabrik 1988

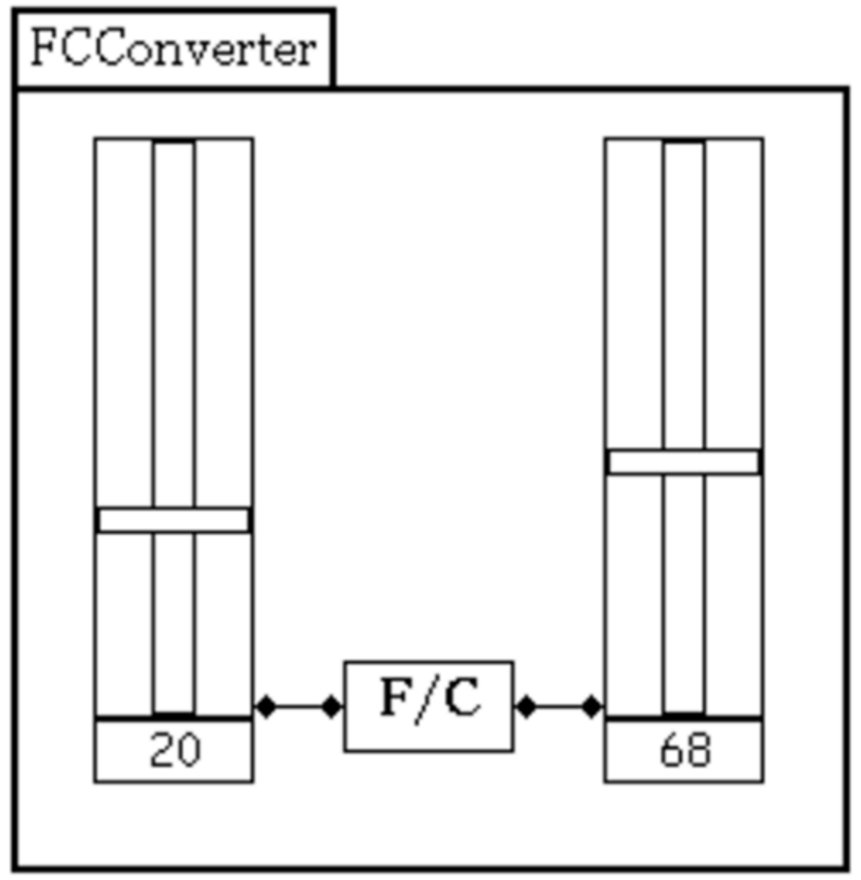


Figure 2a. *Bidirectional diagram using two Slider controls to achieve a Fahrenheit-to-Centigrade converter.*

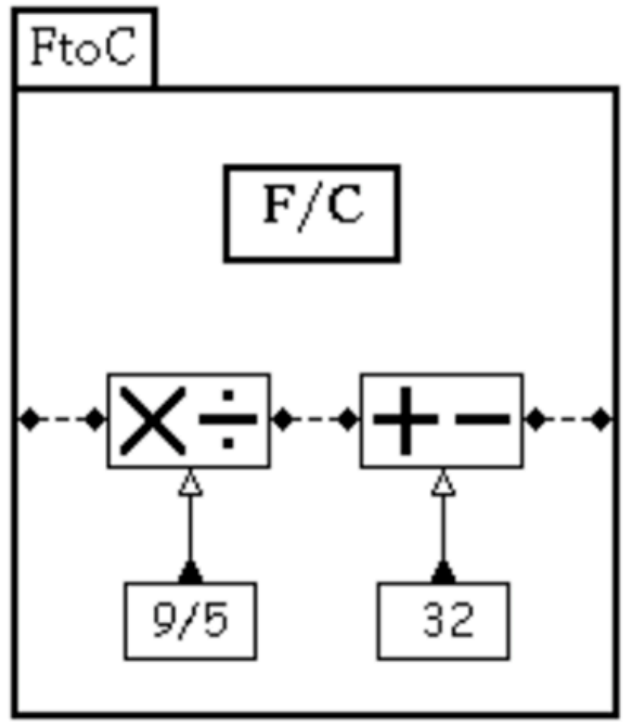


Figure 2b. *Internal diagram for the F/C component used in the diagram at left.*

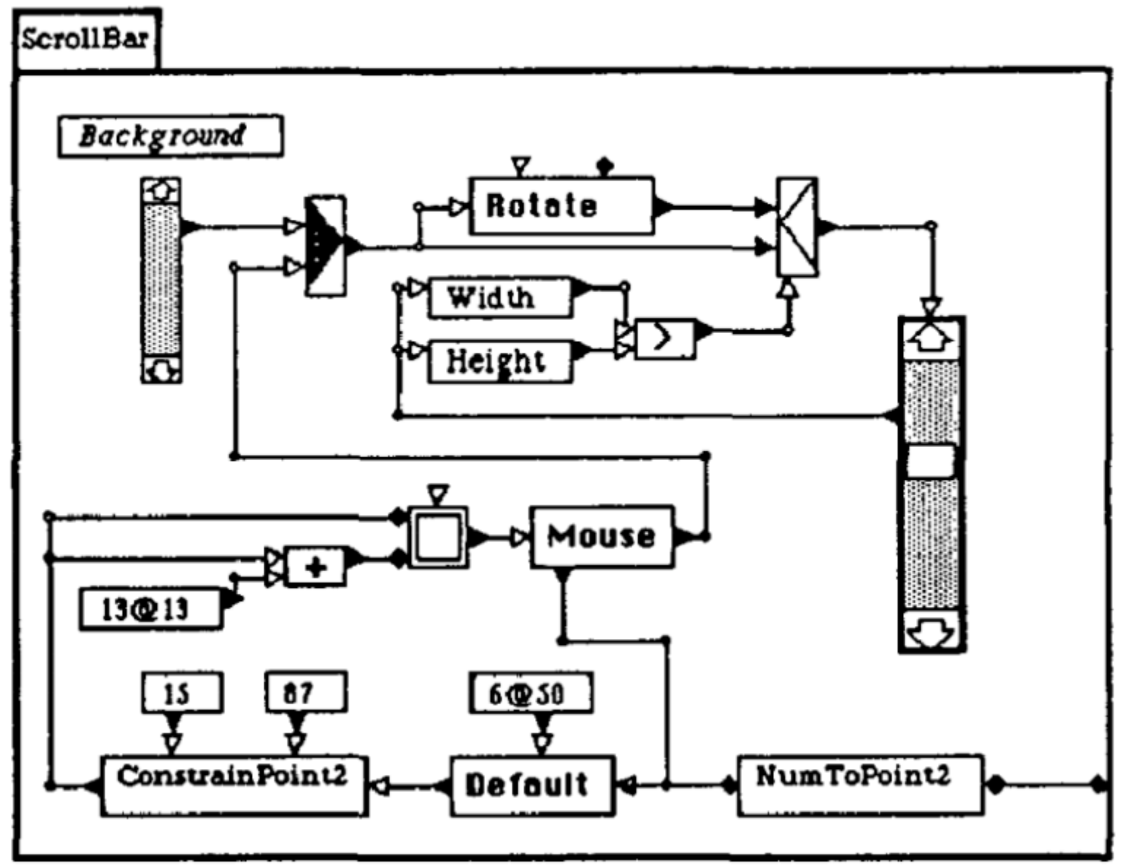


Figure 4a. A simple scrollbar diagram. Logic is provided for rotating when the image is wider than high.

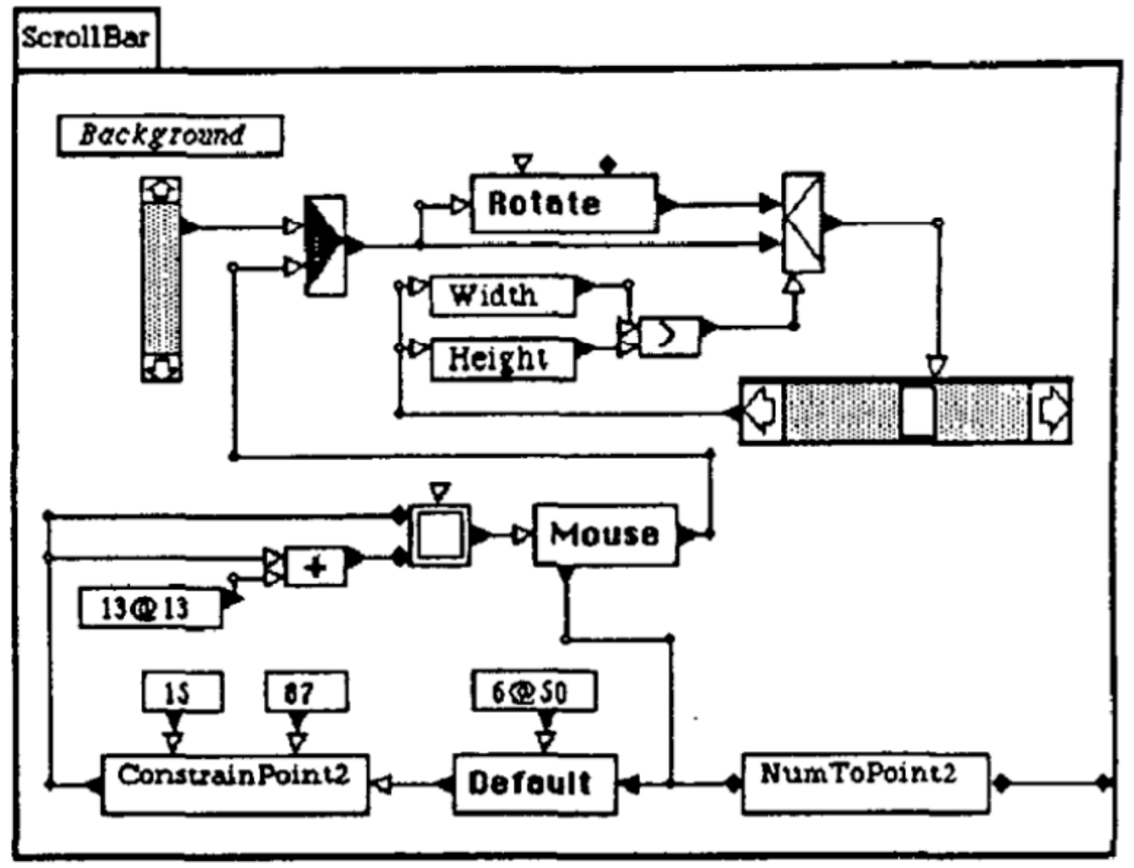


Figure 4b. Demonstrating the rotation logic. All display and mouse-tracking operations are properly scaled and rotated.

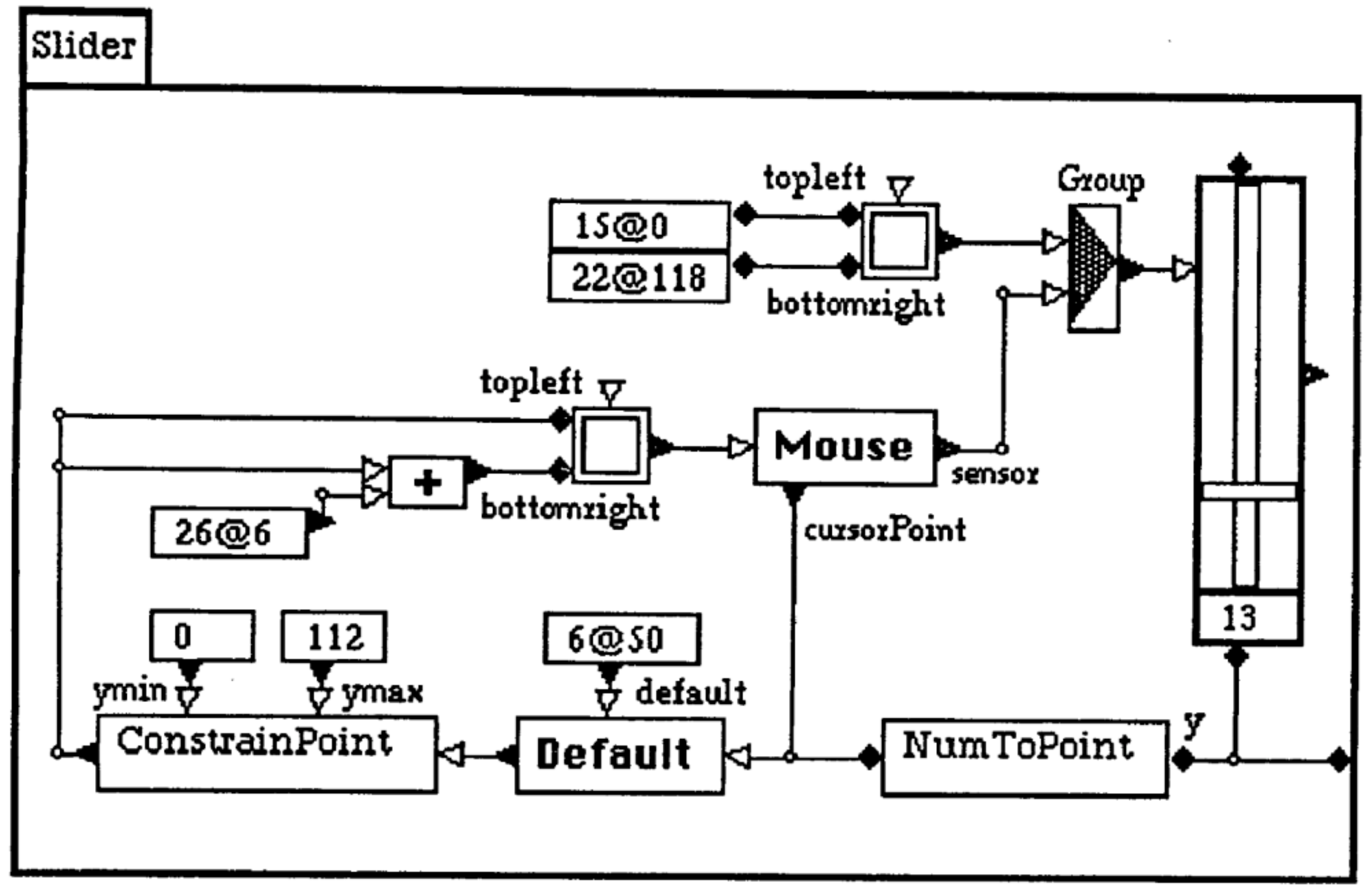


Figure 3. A Fabrik diagram computes the image for the slider in figure 2. The Mouse component sensitizes the slider image to support input as well as output.

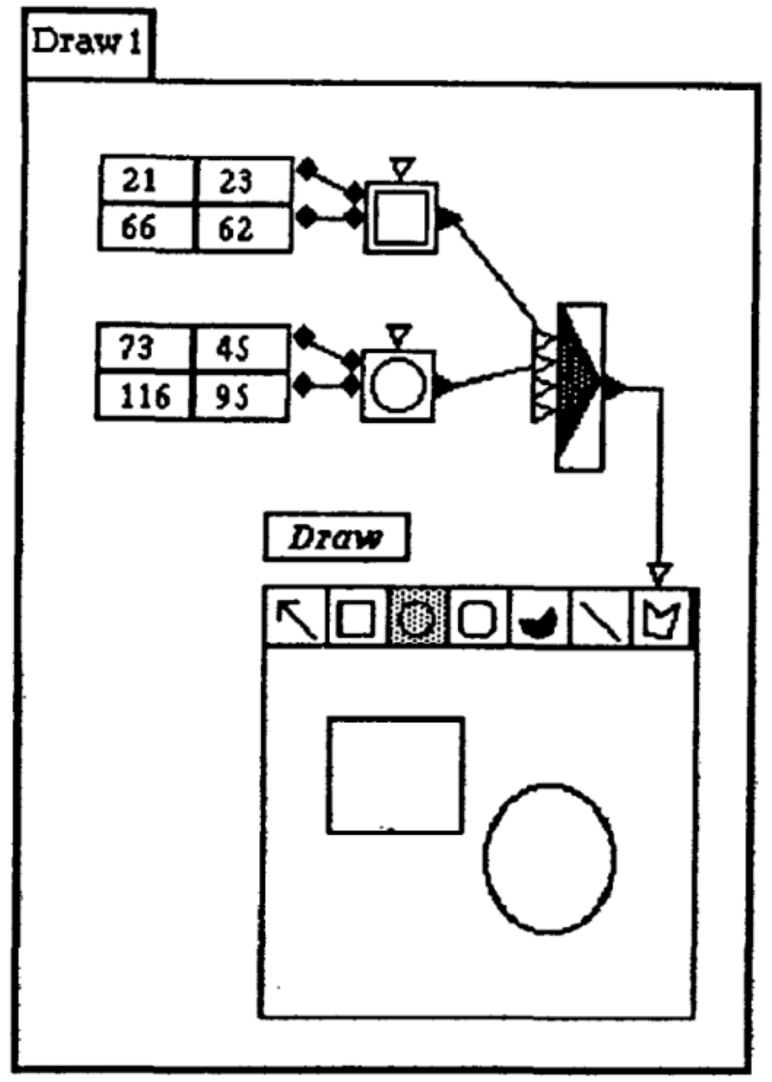


Figure 5a. The Draw component automatically lays out diagrams as the user creates a drawing.

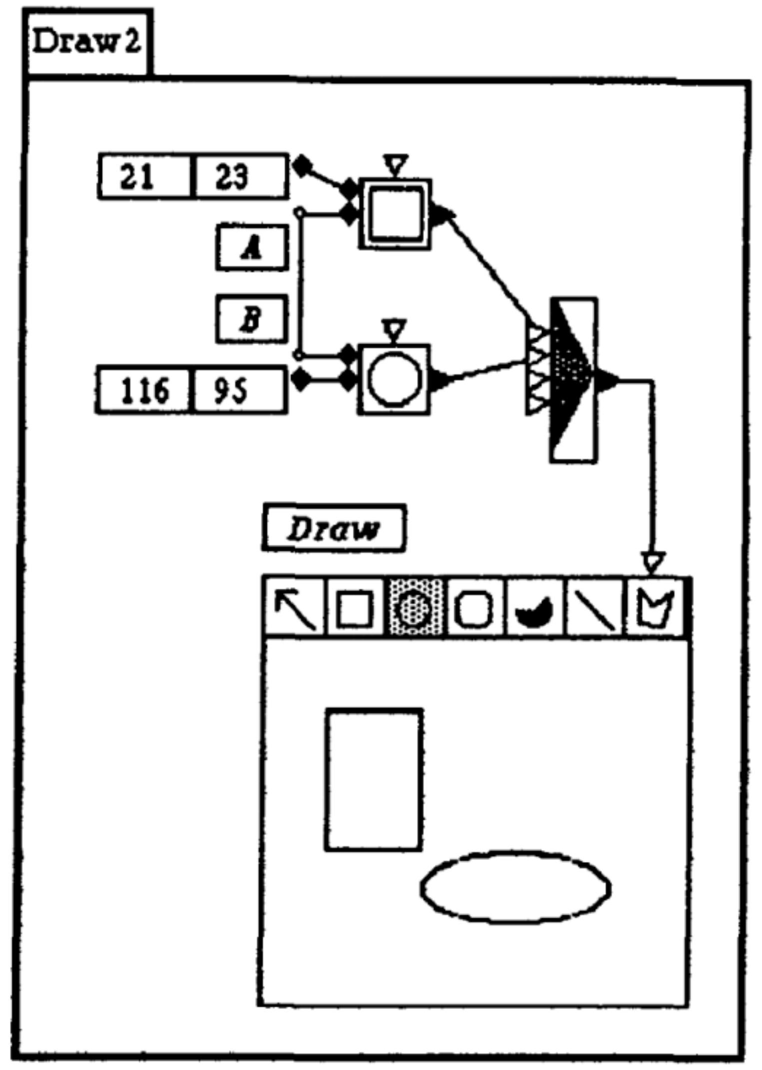


Figure 5b. By editing the generative diagram, the top-left of the oval is tied to the bottom-right of the rectangle.

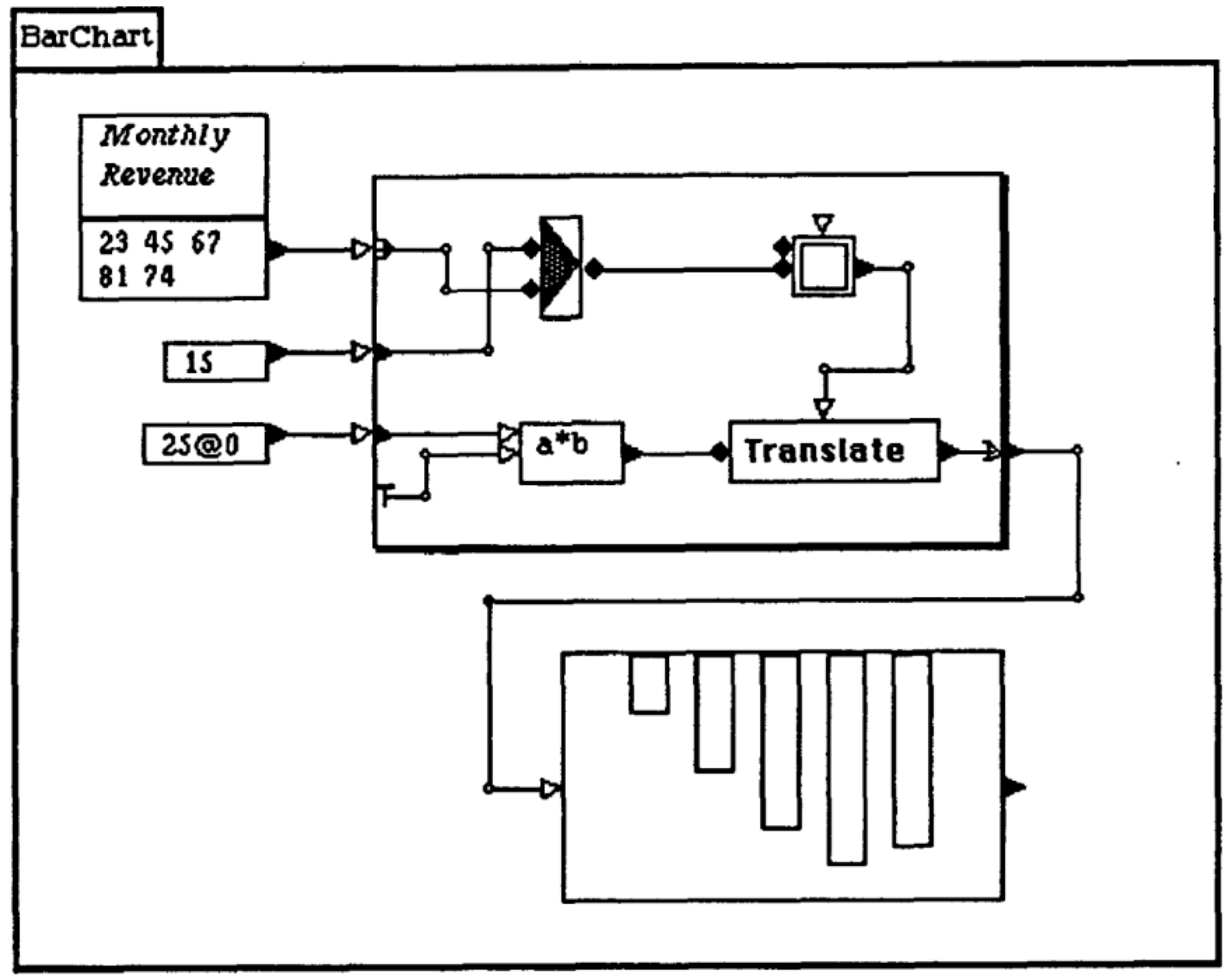
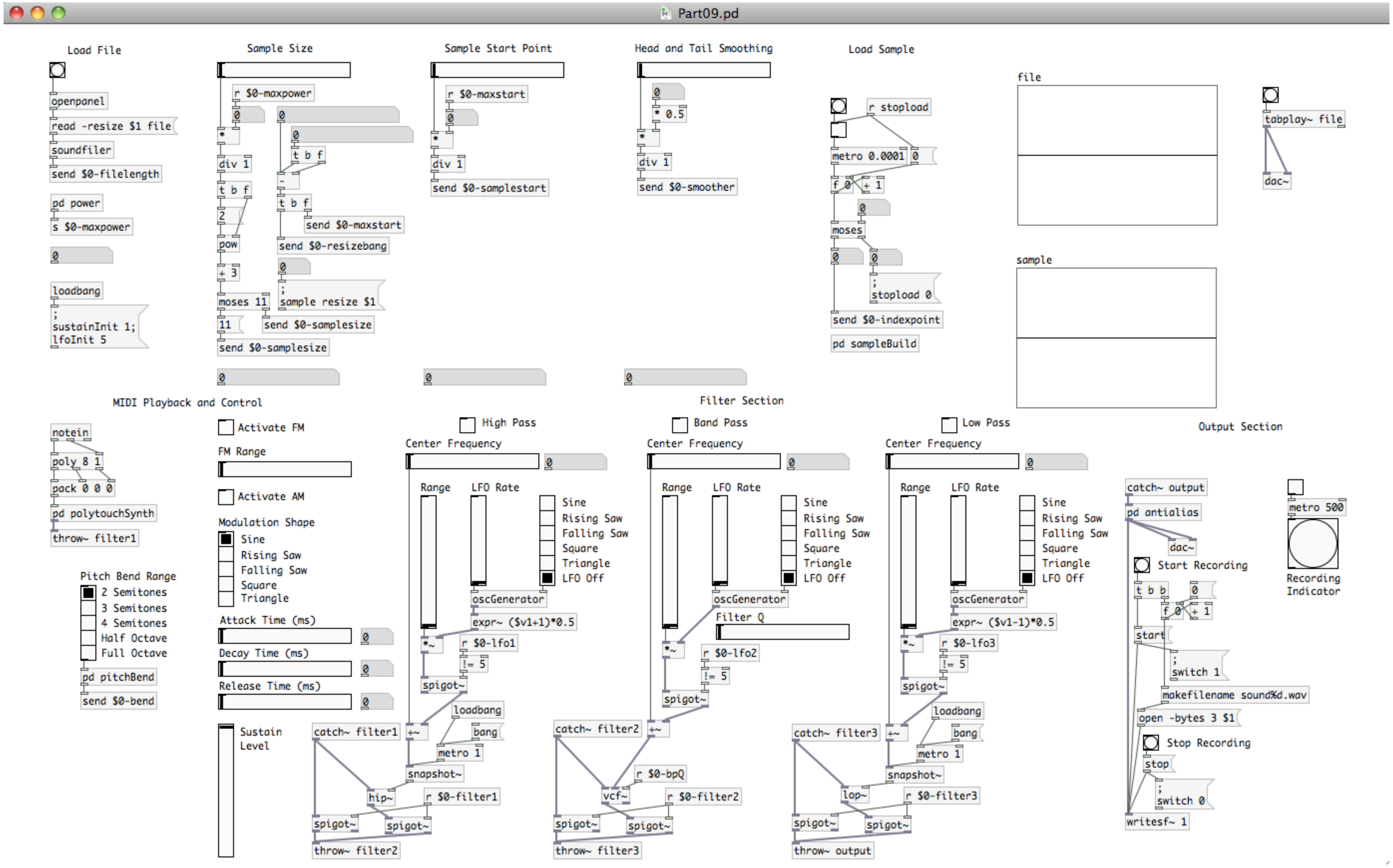


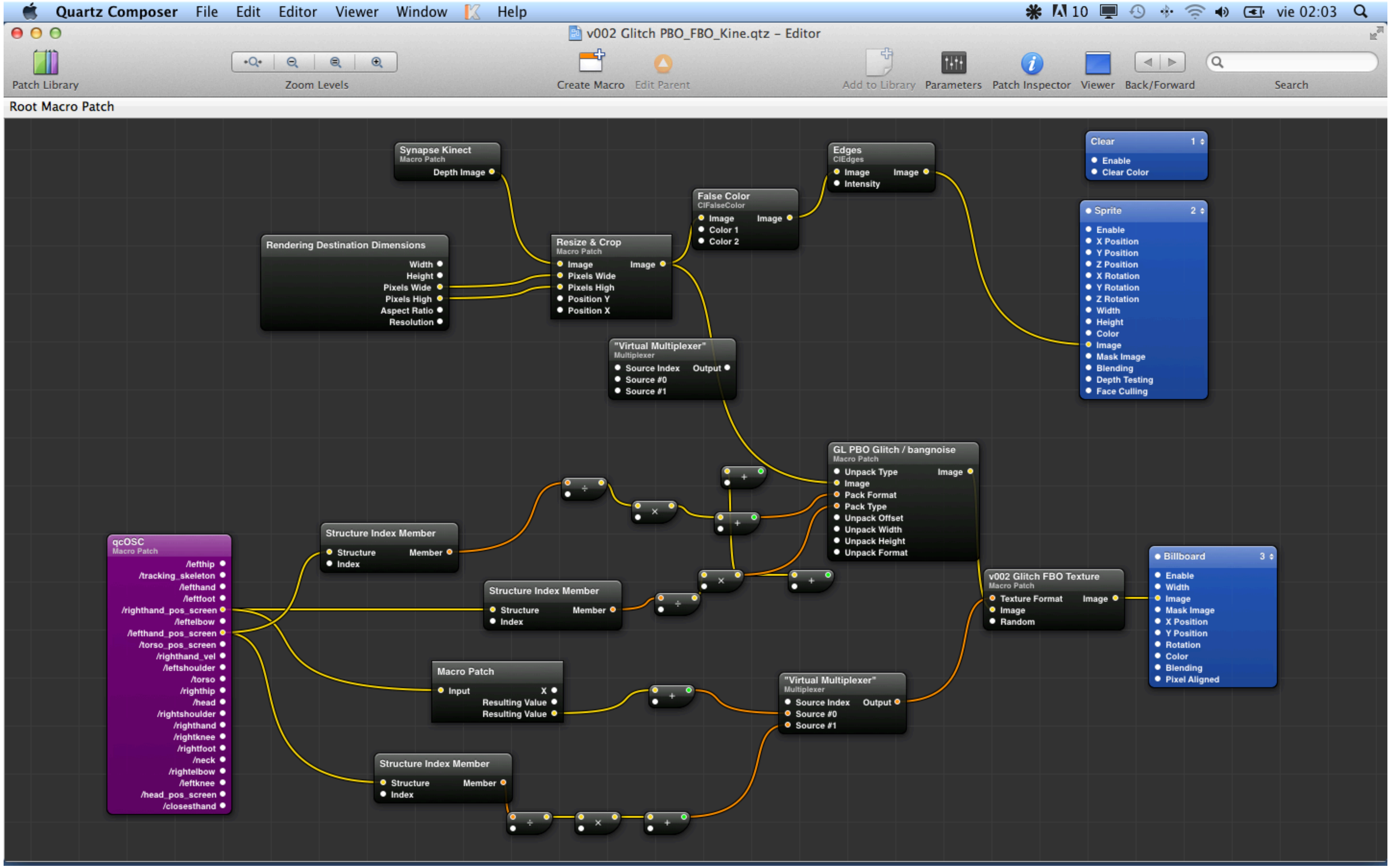
Figure 6. Streaming gateways provide iterative capability. Here numbers are converted to rectangles, resulting in a simple bar chart.

Pure Data (1996)

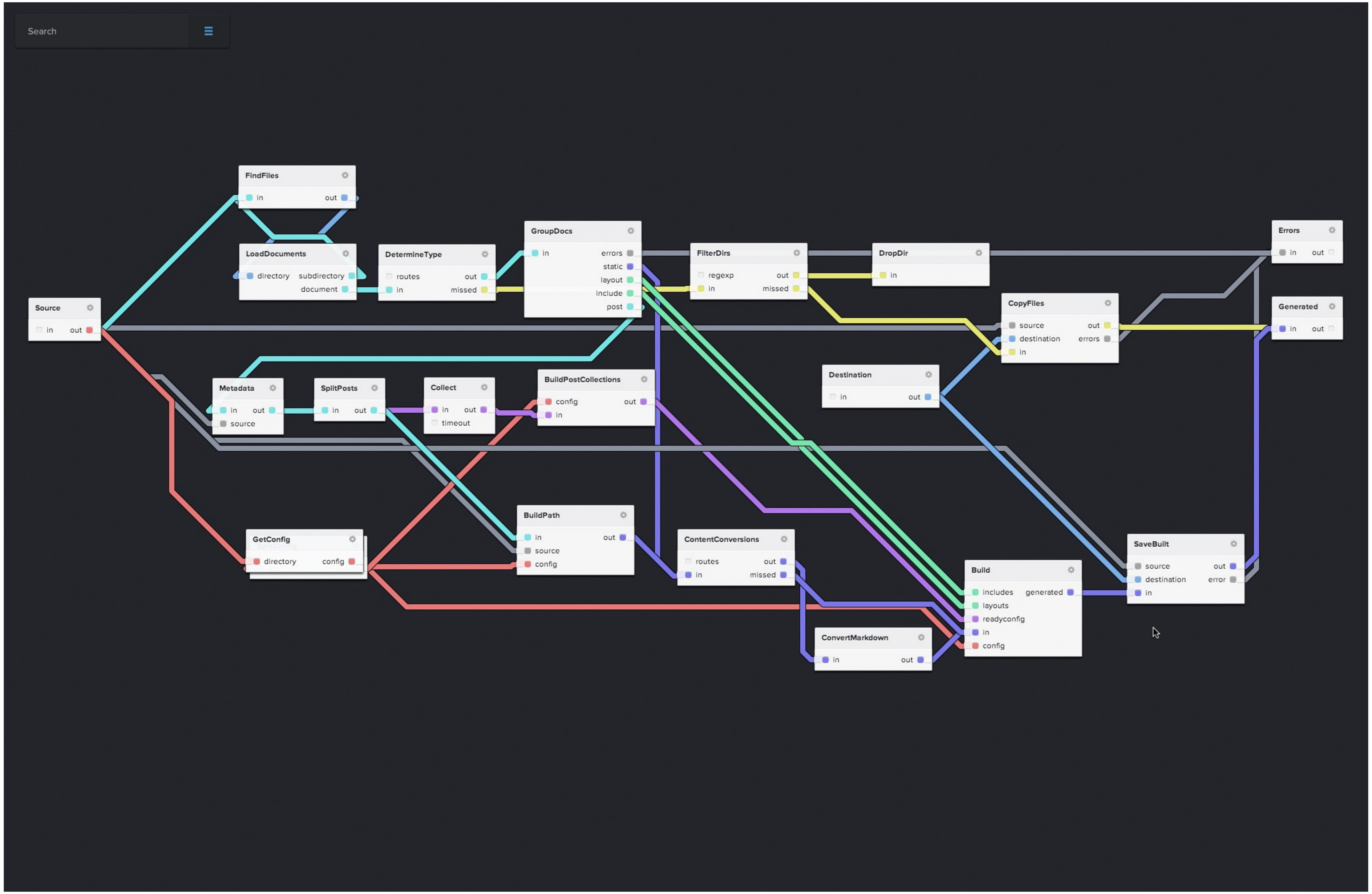



Collapse All
Fetch Feed
URL
http://darkufo.blogspot.com/feeds/
Create RSS
Title:
Description:
Link:
PubDate:
Author:
GUID:
media:content
media:thumbnail

Quartz Composer (2005)



NoFlo (2014)

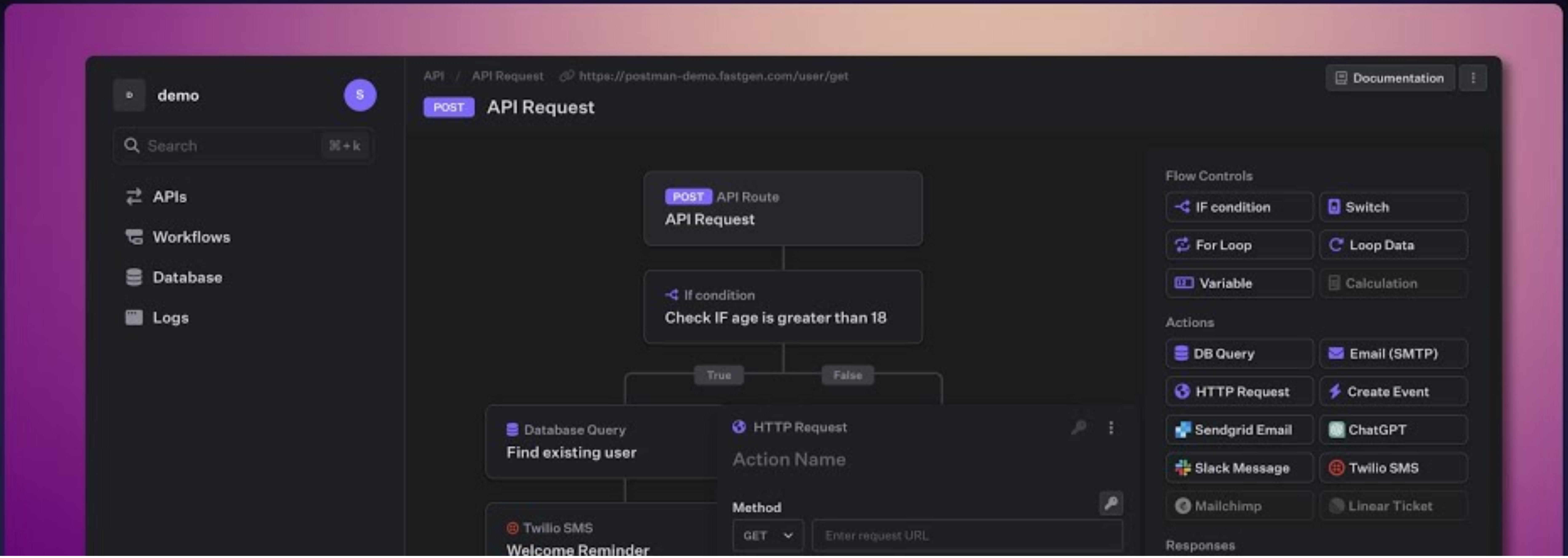


Search
FindFiles in out
LoadDocuments directory subdirectory document
DetermineType routes out in missed
GroupDocs in errors static layout include post
FilterDirs regexp out in missed
DropDir in
Errors in out
Generated in out
CopyFiles source out destination errors in
Source in out
Destination in out
Metadata in out source
SplitPosts in out
Collect in out timeout
BuildPostCollections config out in
BuildPath in out source config
GetConfig directory config
ContentConversions routes out in missed
Build includes generated layouts readyconfig in config
SaveBuilt source out destination error in
ConvertMarkdown in out


BP_HelloWorld > EventGraph
Event Begin Play
Set Text
Target is Text Render Component
Target
Value
Hello World
BLUEPRINTS
GETTING STARTED

Fastgen in 5 minutes

David
Founder



Form Representations

- Program consists of a form, with a network of interconnected cells
- Developers define cell through combination of pointing, typing, gesturing
- Cells may define constraints describing relationships between cells

Forms/3

- Based on constraints between cells
- Supports graphics, animation, recursion
- Concreteness: resulting box is immediately seen
- Directness: demonstrates elements directly
- Level 4 liveness: immediate visual feedback

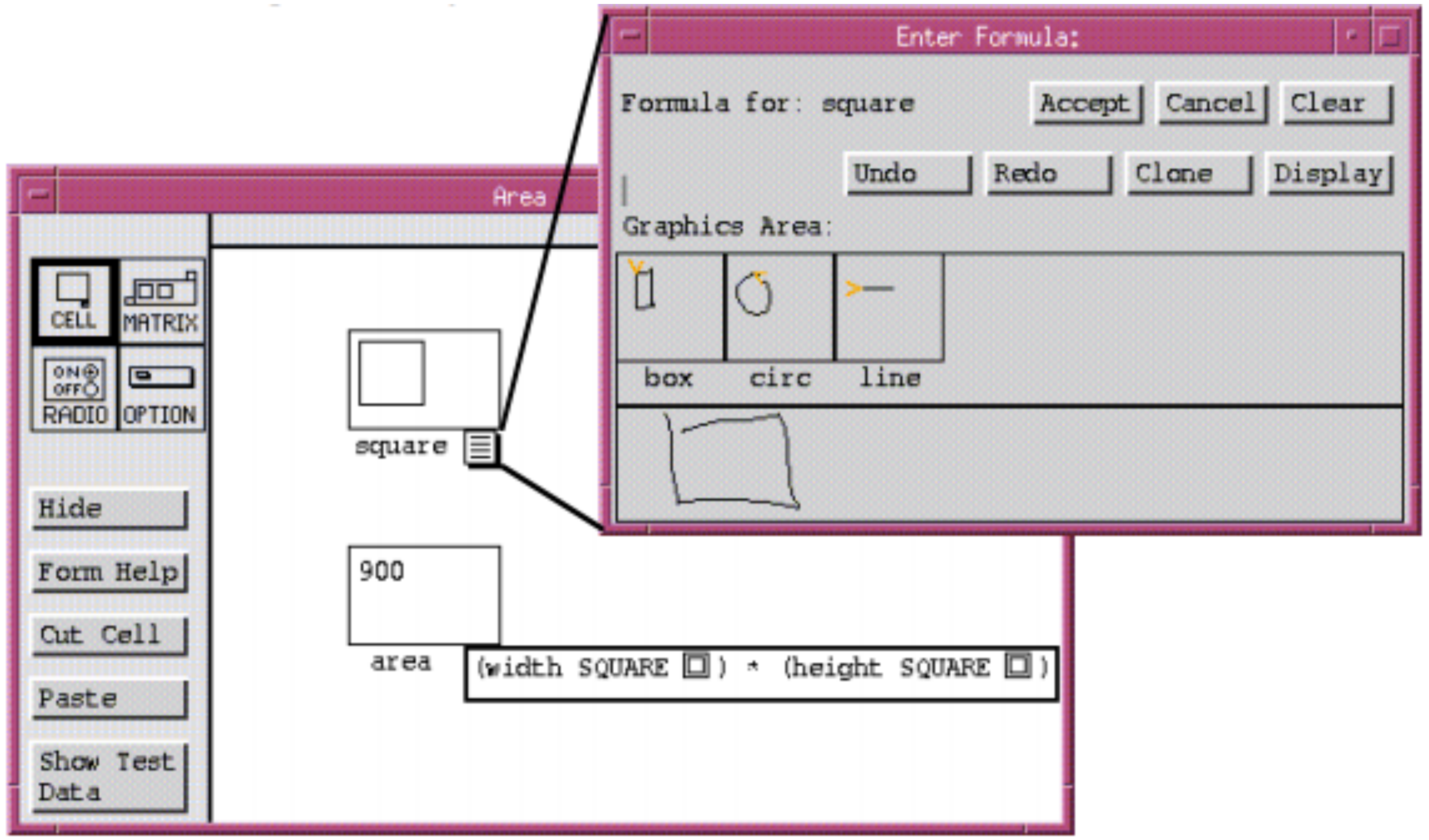
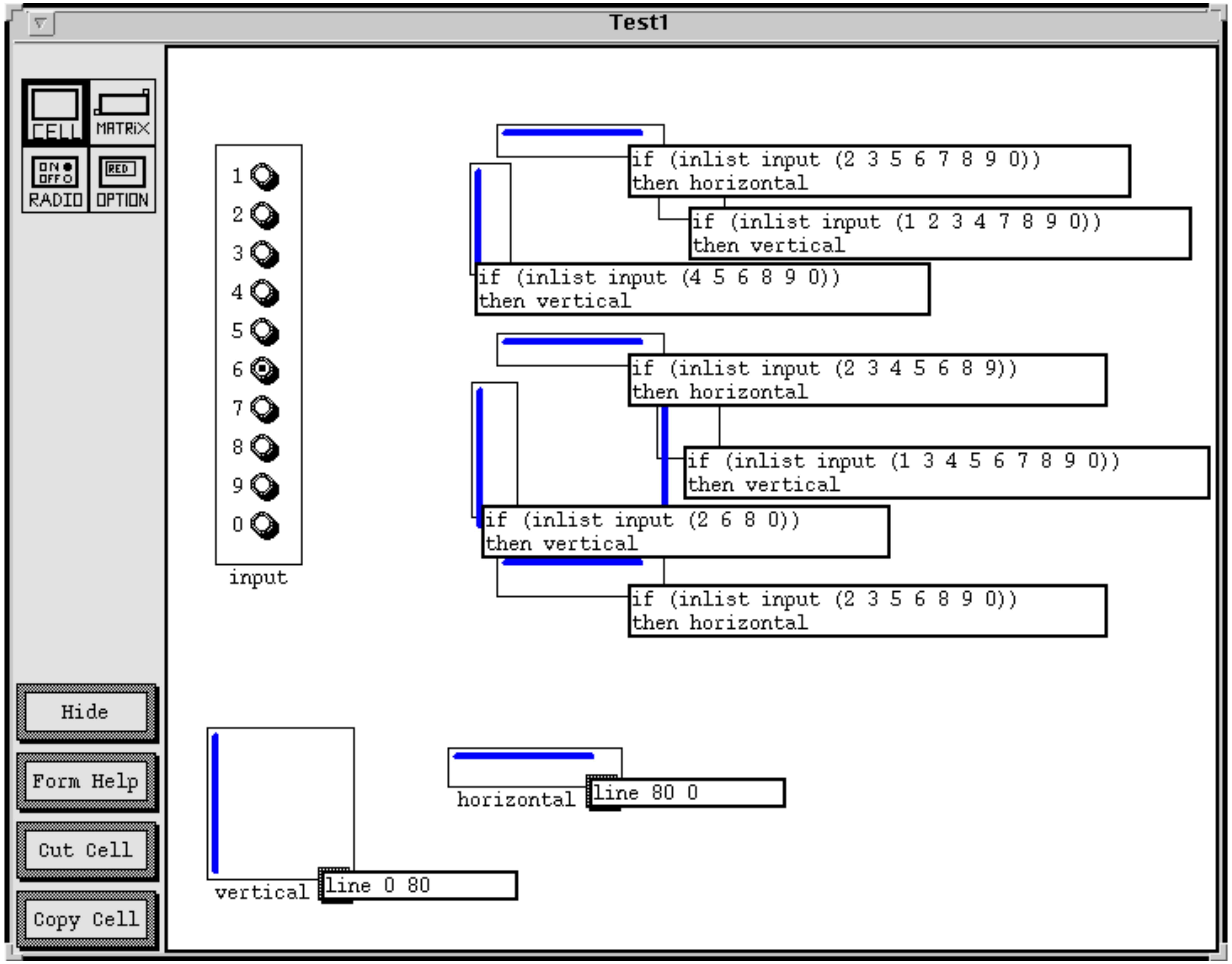


Figure 2: Defining the area of a square using spreadsheet-like cells and formulas in Forms/3. Graphical types are supported as first-class values, and the programmer can enter cell square's formula either by sketching a square box or by typing textual specifications (e.g., "box 30 30").

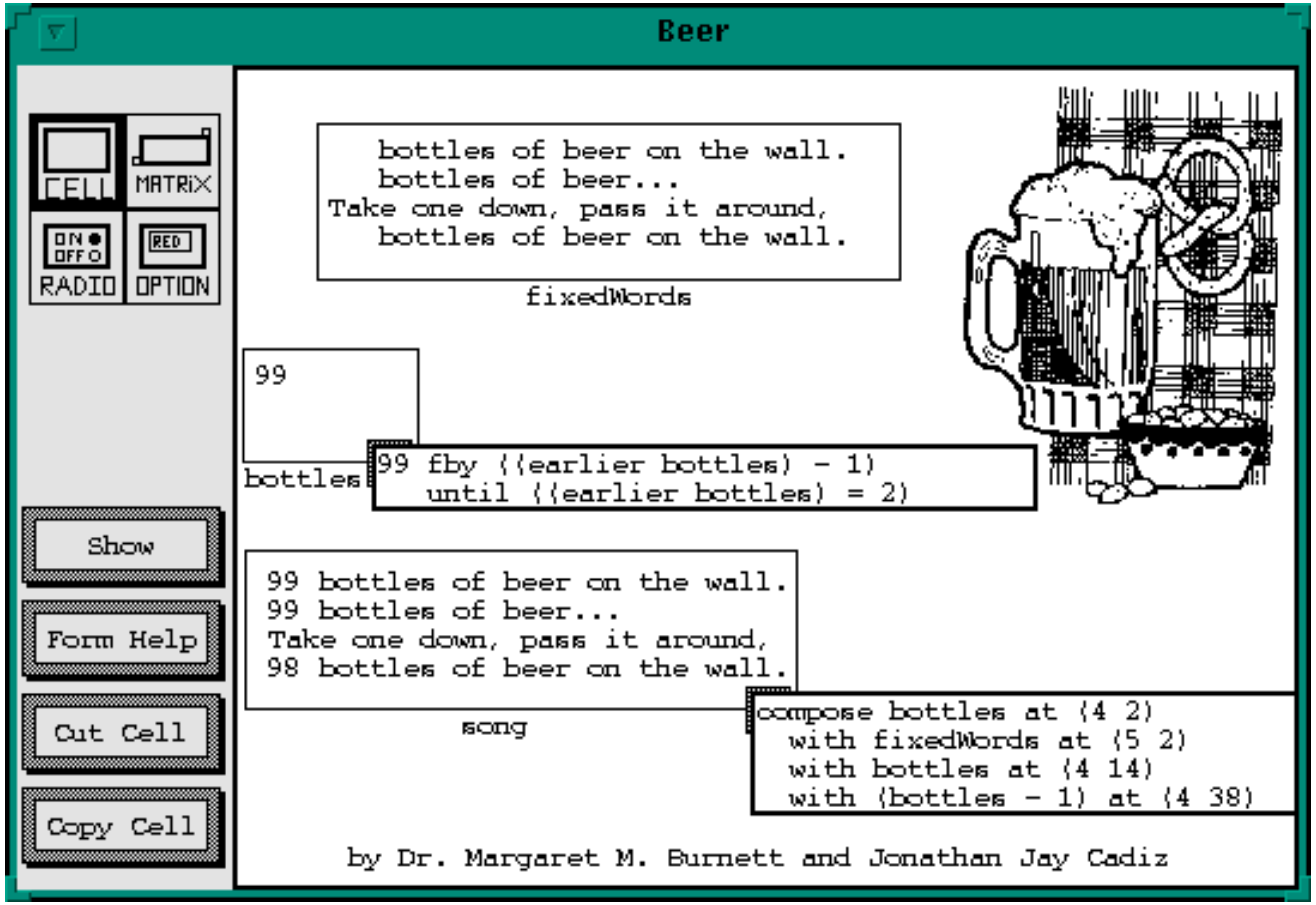
Margaret M. Burnett, "Visual Programming" In the Encyclopedia of Electrical and Electronics Engineering (John G. Webster, ed.), 1999

Forms/3 Example



http://web.engr.oregonstate.edu/~burnett/Forms3/LED.html

Forms/3 Example



Discussion

- Given potential advantages, why isn't all programming now done in a visual programming language?

Cognitive Dimensions of Notations

- Analytical technique for assessing usability of notation through a set of heuristics
  - Also terminology for describing usability problems

Abstraction gradient	What are the minimum and maximum levels of abstraction? Can fragments be encapsulated?
Closeness of mapping	What 'programming games' need to be learned?
Consistency	When some of the language has been learnt, how much of the rest can be inferred?
Diffuseness	How many symbols or graphic entities are required to express a meaning?
Error-proneness	Does the design of the notation induce 'careless mistakes'?
Hard mental operations	Are there places where the user needs to resort to fingers or penciled annotation to keep track of what's happening?
Hidden dependencies	Is every dependency overtly indicated in both directions? Is the indication perceptual or only symbolic?
Premature commitment	Do programmers have to make decisions before they have the information they need?
Progressive evaluation	Can a partially-complete program be executed to obtain feedback on "How am I doing"?
Role-expressiveness	Can the reader see how each component of a program relates to the whole?
Secondary notation	Can programmers use layout, color, or other cues to convey extra meaning, above and beyond the 'official' semantics of the language?
Viscosity	How much effort is required to perform a single change?
Visibility	Is every part of the code simultaneously visible (assuming a large enough display), or is it at least possible to compare any two parts side-by-side at will? If the code is dispersed, is it at least possible to know in what order to read it?

T. Green and M. Petre, Usability analysis of visual programming environments: a 'cognitive dimensions' framework. Journal of Visual Languages and Computing 7(2): 131-174, June

Diffuseness / Terseness

- How many symbols or graphic elements is required to express a meaning?

- Simple rocket simulation program
- Basic: 22 LOC, 140 words (fits on screen)
- LabView: 45 icons, 59 wires (fits on screen)
- Prograph: 52 icons, 79 connectors, 11 screens

T. Green and M. Petre, Usability analysis of visual programming environments: a ‘cognitive dimensions’ framework. Journal of Visual Languages and Computing 7(2): 131-174, June 1996

Error-proneness

- Does the design of the notation induce slips?

- Compared to textual language, VPLs
  - Do not need delimiters & separators
  - Fewer identifiers are needed, easier to reference
  - Constructs inserted automatically (e.g., loops)

Viscosity

- How much effort is required to make a simple change?

- Edit Rocket program to take account of air resistance
- Basic: 63.3 s
- LabView: 508.3 s
- Prograph: 193.6 s

- VPLs required many wires to be rebuilt, layout to be tweaked

Visibility

- Is every (relevant) part of the code simultaneously visible?

- LabView does not show both branches of conditional at same time (!)
  - Particular problem for nested conditionals
- Prograph has poor support for deep nesting of routines

VPLs Discussion

- Often offers a representation that makes specific tasks easy
  - e.g., tracking data flow
  - Often involves structured editor targeted to specific domain, which may not support full range of programs
- But may make other tasks harder
- Often limited focus on scalability

- May be possible to get benefits of task-specific representations without drawbacks through task specific **editor** rather than language

GUI Builders

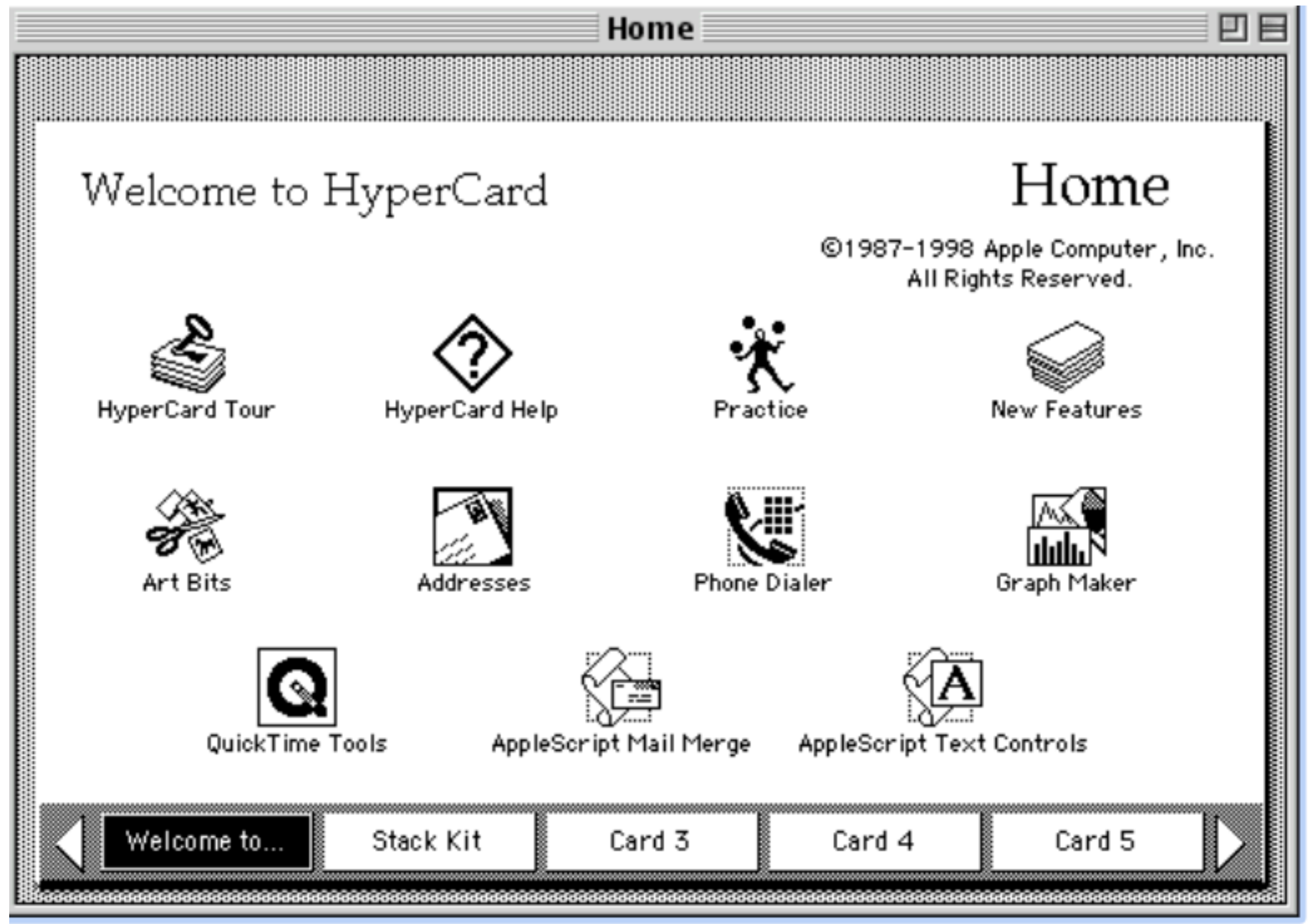
GUI Builders

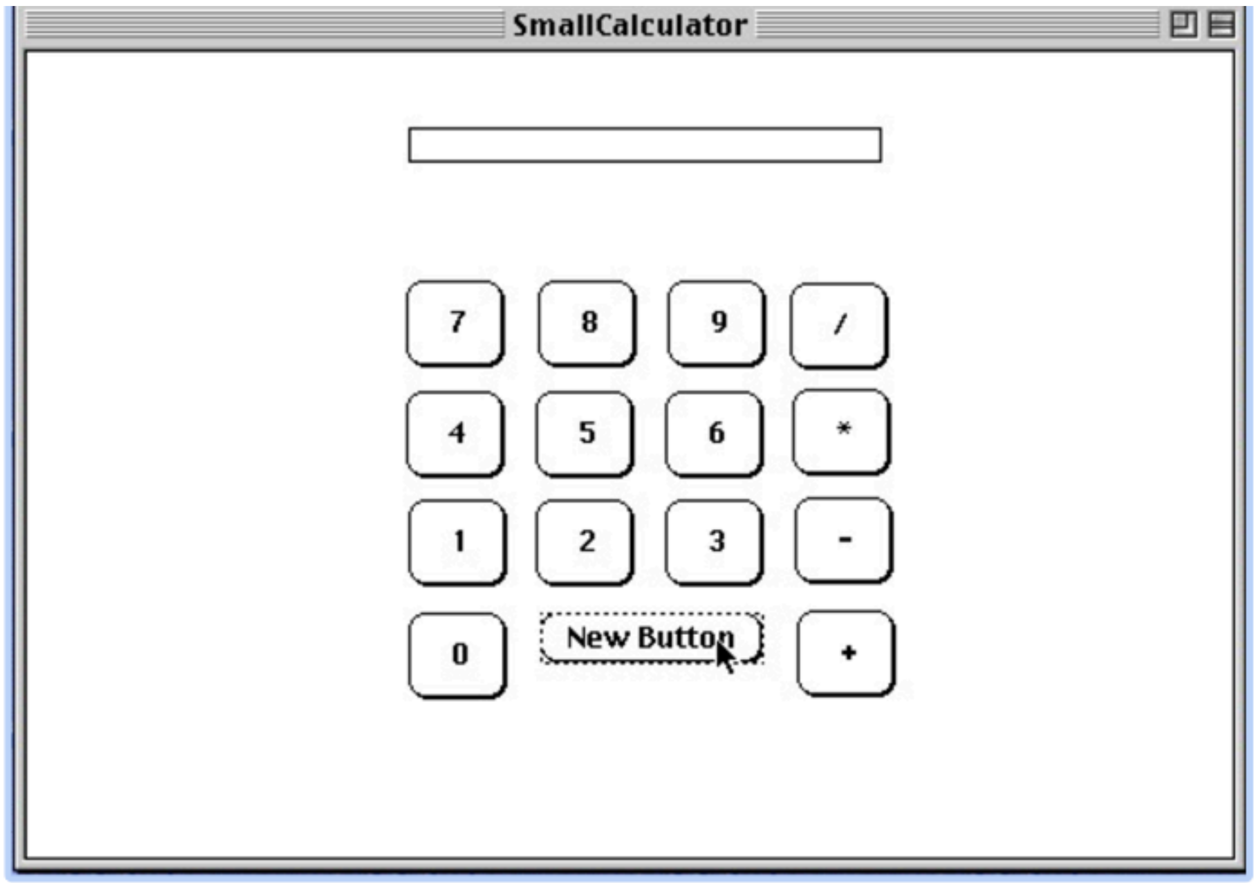
- User interfaces are visual
  - Why not edit interfaces visually?
  - Enables direct manipulation - drag and drop to create element, change properties to style, immediately see new feedback

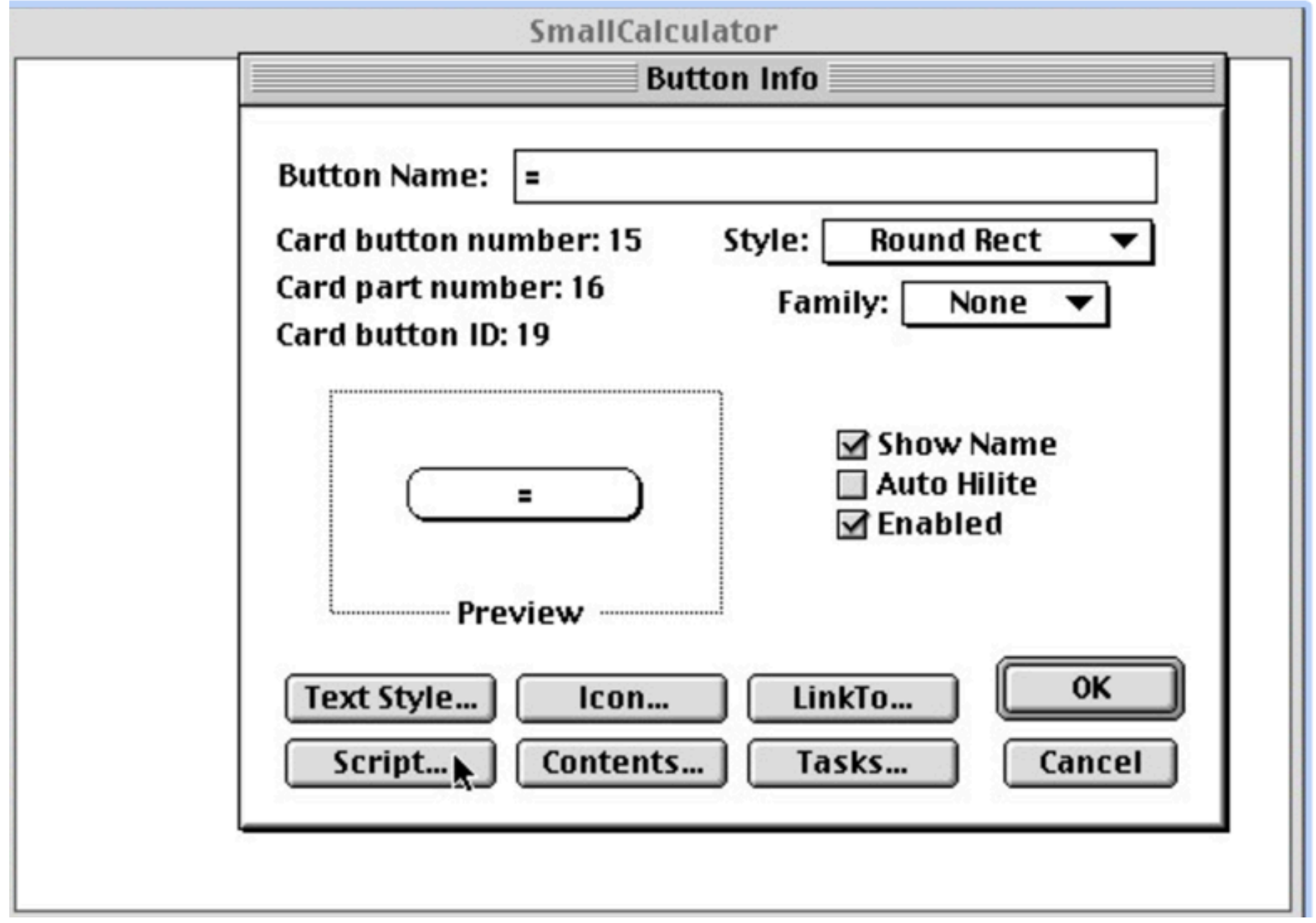
HyperCard (1987)

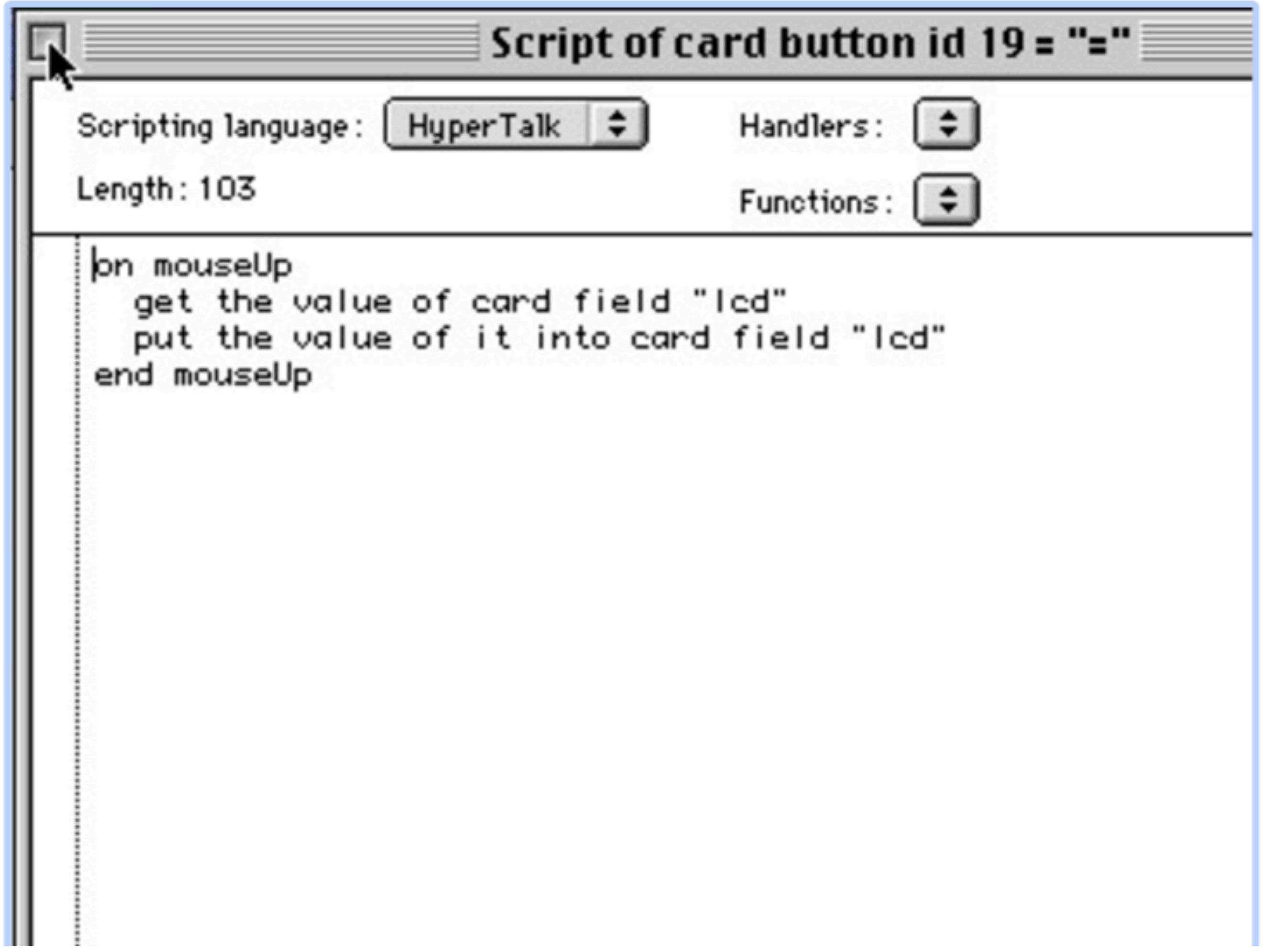
https://www.loom.com/share/6c57bdb7e4d0488eb2f9f8949b028ef8?t=491

HyperCard (1987)









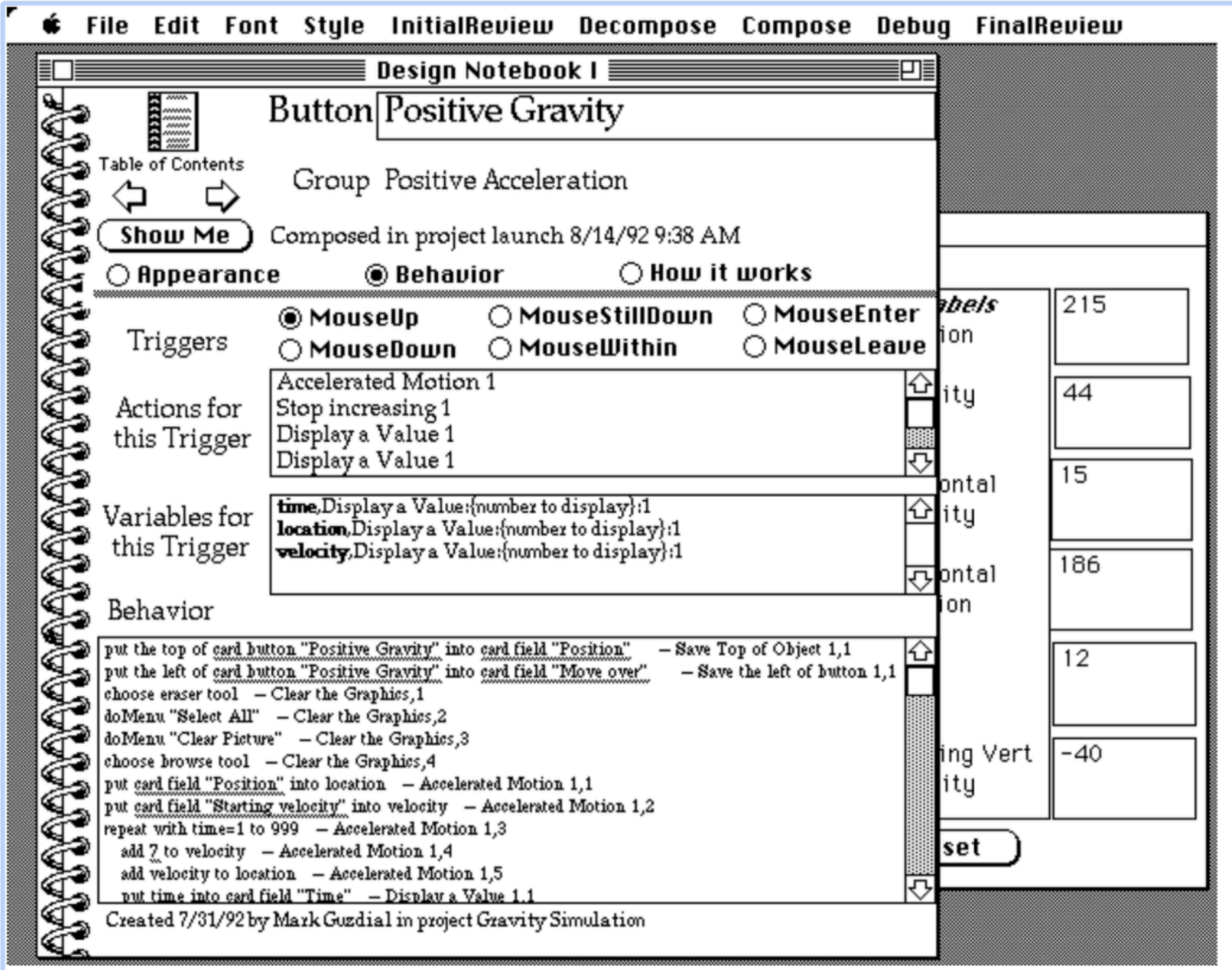
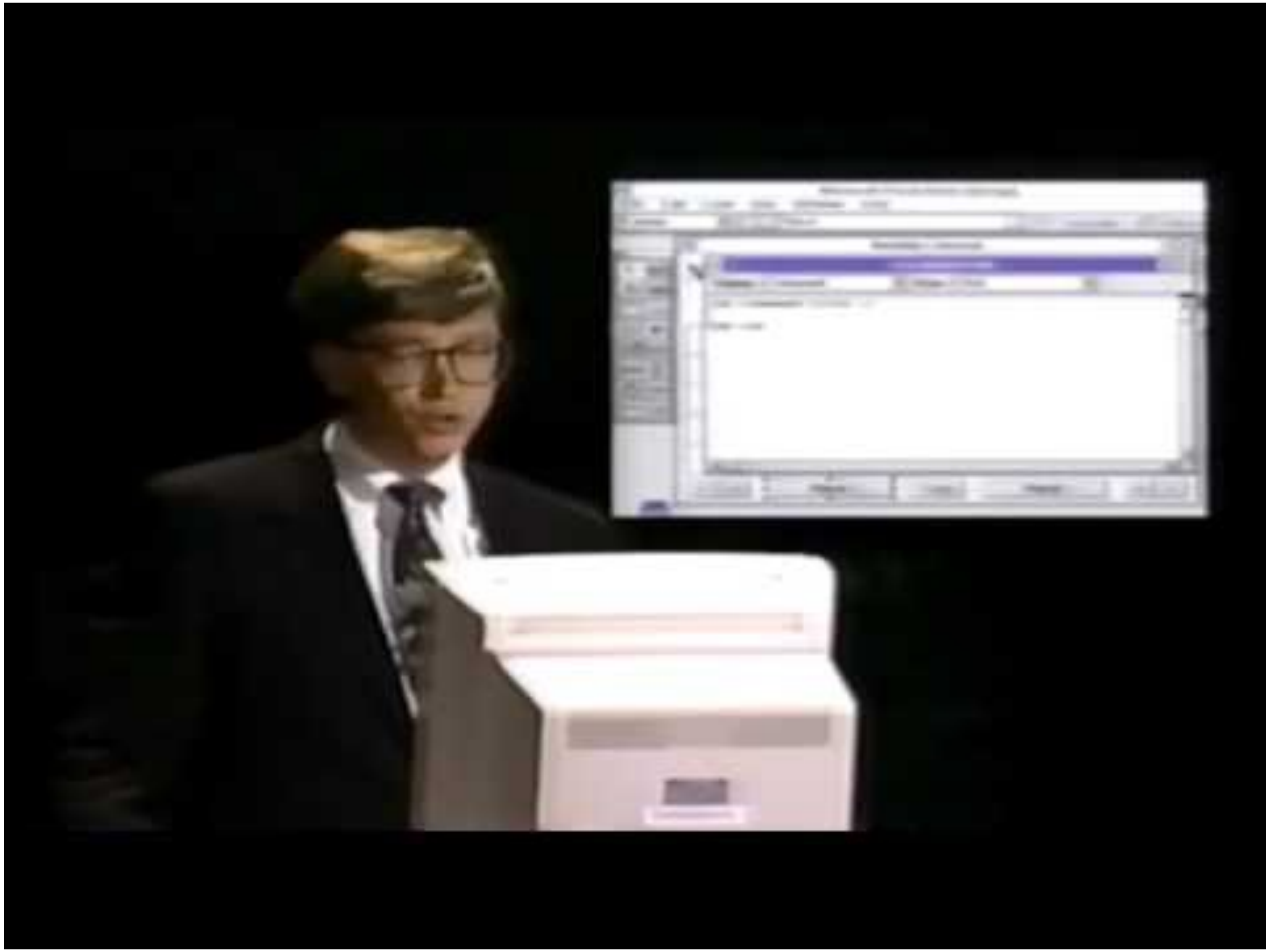
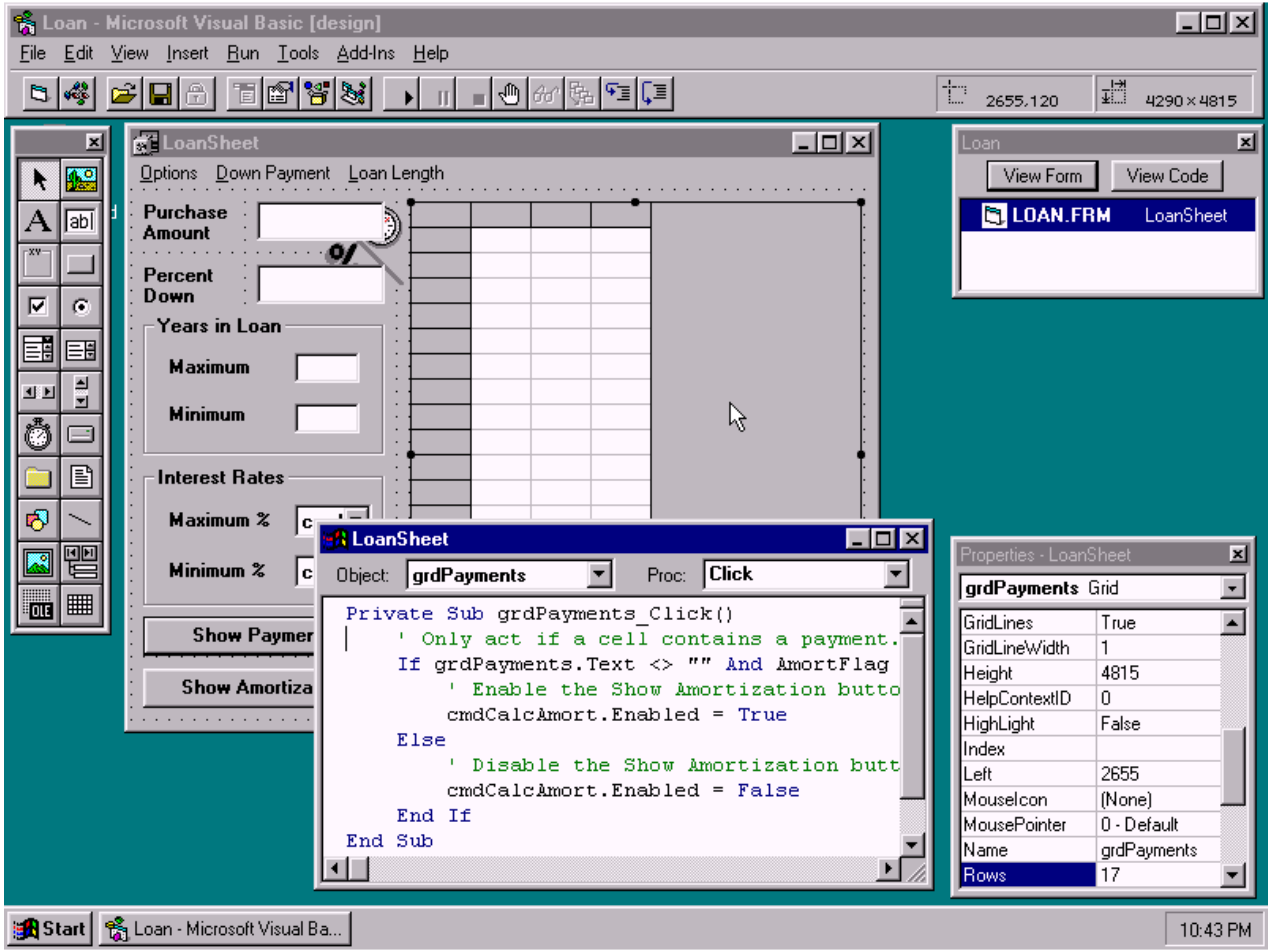


Fig. 23. An editor for the Positive Gravity Button. When the mouse goes up, Emile will execute 4 actions: Accelerated Motion 1, Stop Increasing 1, and Display a Value 1 (2 times). At the bottom of the screen, we can see the code that Emile will execute. Underlined text corresponds to parameters (or slots) that the user can fill in using menu options and dialog boxes.

Visual Basic (1991)



Elements of a GUI Builder



Profile-clone-sass
omer/topbar-connections
Commit (2)
Sync Updated 1 min ago
Top Bar
Top Bar
75%
1352 × 452
Elements
+ Add
Window
Canvas
TopBar
auto (967px)
auto (287px)
TopBar
Connect
John Doe
john-doe
Member since 1984
USA
john@doe.com
966 × 287
Properties
TopBar (no classes)
JohnDoe
ApiUser
bio *
string
Some bio information
email *
string
john@doe.com
location *
string
USA
name *
string
John Doe
organization *
string
(empty string)
organizations *
ApiOrganization[]
Add items
0
twitter *
null | string
(empty string)
url *
string
(empty string)
username *
string
john-doe
Code
Log
Top Bar
bio: 'Some bio information',
username: 'john-doe',
location: 'USA',
name: 'John Doe',
email: 'john@doe.com',
followers: 19,
following: 20,
twitter: '',
url: '',
organization: '',
organizations: [],
techs: {
technologies: ['js', 'node', 'react', 'python'],
},
};
top-bar.board.tsx M
src > _wcs > boards > top-bar > top-bar.board.tsx > JohnDoe > organi
import { createBoard } from '@wixc3/react-board';
import { TopBar } from '../../../components/pages/to
import type { ApiUser } from '../../../shared/model/
const JohnDoe: ApiUser = {
avatar: 'https://i.pravatar.cc/64?u=john-doe',
bio: 'Some bio information',
username: 'john-doe',
location: 'USA',
name: 'John Doe',
email: 'john@doe.com',
followers: 19,
following: 20,
twitter: '',
url: '',
organization: 'Code Ducks',
organizations: [],
techs: {
technologies: ['js', 'node', 'react', 'pytho
},
};
export default createBoard({
name: 'Top Bar',
Board: () => <TopBar user={JohnDoe} />,
environmentProps: {
windowHeight: 452,
windowWidth: 1352,
},
});
omer/topbar-connections*
1 Spell
Prettier

Challenges

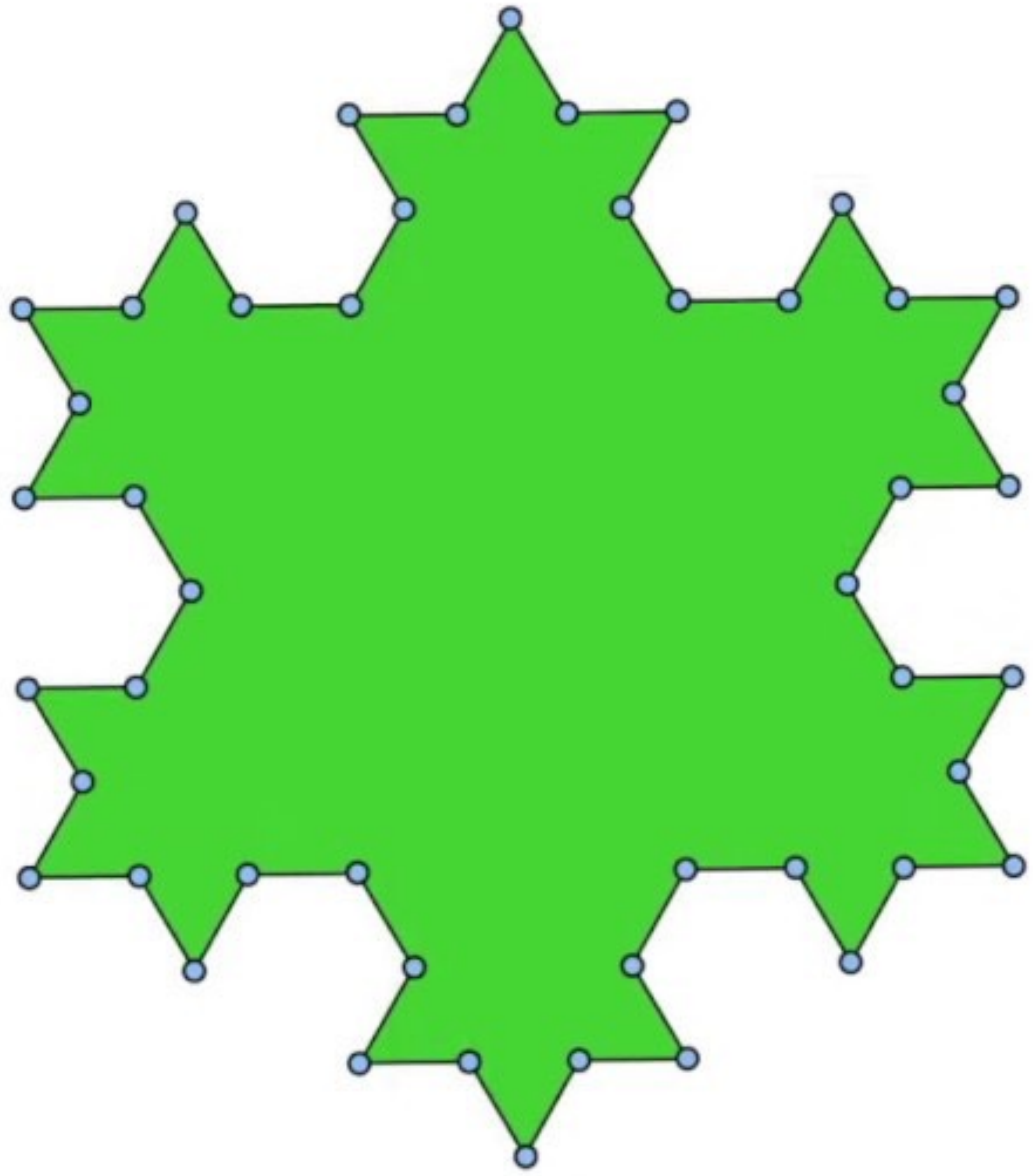
- Web apps dynamically generate HTML based on data
  - Lists of elements, conditional behavior
- Keeping code and GUI builder **in sync**
  - What happens if code includes something that can't be represented in GUI builder properties (e.g., an animation)
- --> Need to **abstract** an immutable concrete value set in a property editor into a relationship that is dynamically computed from other variables
  - Potential for ambiguity, where there's multiple ways to arrive at a specific concrete value

Current file: Untitled *

Undo | Redo | Clean Up | Run

```
equiTriPt [x3, y3] [x2, y2] =
  [ (x3 + x2 + sqrt 3! * (y3 - y2))/ 2!, (y3 + y2 - sqrt 3! * (x3 - x2

oneThirdPt [x2, y2] [x, y] =
  [ x / 1.5!+ x2 / 3!, y / 1.5! + y2 / 3!]

point = [69, 308]

point2 = [642, 301]

makeKochPts depth pt1 pt2 =
  let oneThirdPt2 = oneThirdPt pt1 pt2 in
  let oneThirdPt3 = oneThirdPt pt2 pt1 in
  let equiTriPt2 = equiTriPt oneThirdPt2 oneThirdPt3 in
  if depth < 2 then
    [pt1, oneThirdPt3, equiTriPt2, oneThirdPt2]
  else
    let makeKochPts2 = makeKochPts (depth - 1) pt1 oneThirdPt3 in
    let makeKochPts3 = makeKochPts (depth - 1) oneThirdPt3 equiTriPt2
    let makeKochPts4 = makeKochPts (depth - 1) equiTriPt2 oneThirdPt2
    let makeKochPts5 = makeKochPts (depth - 1) oneThirdPt2 pt2 in
      concat [makeKochPts2, makeKochPts3, makeKochPts4, makeKochPts5]

depth = 2{1-5}

topPts = makeKochPts depth point point2

botPt = equiTriPt point point2

rightPts = makeKochPts depth point2 botPt

leftPts = makeKochPts depth botPt point

snowflakePts = concat [topPts, rightPts, leftPts]

polygon1 =
  let pts = snowflakePts in
  let [color, strokeColor, strokeWidth] = [114, 360, 2] in
    polygon color strokeColor strokeWidth pts

svg (concat [
  [polygon1]
```

Context: Program



3x

Cursor
Point or Offset
Polygon

User-Defined Tools

equiTriPt
oneThirdPt
makeKochPts

Standard Library Tools

vec2DPlus
vec2DLength
circle
ring
ellipse
rect
square
line
rectByCenter
squareByCenter
nPointsOnCircle
nPointsOnSegment
nPointsSepBy
nHorizontalPointsSepBy
nVerticalPointsSepBy
pointsBetweenSepBy
midpoint


InterState Runtime
EDIT
0:05/3:19
Volume:
All songs
New Playlist Name
Name
Artist
Duration
Run (Terrifying Night)
Admiralbob77
3:21
Entranced Beauty the Beauty
Carosone
3:39
Close to Mike Jazz Mix (ft. old Dog)
CiggiBurns
3:19
Tattoo (Candy Flowers Edit)
CoffeeEurope
3:30
Mad Dirty Naked
CSoul
3:22
FeedForward (DuckBack Mix)
Duckett
2:52
Wake up Kiss
Hansatom
3:11
CSoul Oriental Express
JeffSpeed68
3:36
Surrender
One Project
3:52
Made from Corn
Only Meth
3:44
InterState Editor
Undo
Redo
Files
sketch
Add Field
song_info
button
player
svg
dom
physics
on
(native
find
(native
emit
(native

10 min break

Tech Talk: Microsoft Power Automate

In-Class Activity

- In groups of 2, try out Microsoft Power Automate
  - https://powerautomate.microsoft.com/en-us/#home-signup
  - Setup the free trial for the Web version of Power Automate
  - Read docs to understand how to build a Power Automate App.
  - Pick a data processing problem based on examples in docs
  - Build a Power Automate App and try it out.