Gallery of programmer interfaces

JonathanMEdwards@gmail.com
@jonathoda
http://alarmingdevelopment.org

These images bear witness to the passionate work of so many people striving to improve programming. Thank you.

Not covered: spreadsheets, data analysis/viz.

Slideshow version at:
https://docs.google.com/presentation/d/1MD-CgzODFWzdpnYXr8bEgysfDmb8PDV6iCAjH5JIval/pub?start=false&loop=false&delayms=3000

See also

https://www.cs.cmu.edu/~caitlin/papers/NoviceProgSurvey.pdf

http://blog.interfacevision.com/design/design-visual-progarmming-languages-snapshots/

https://github.com/ivanreese/visual-programming-codex

https://github.com/shaunlebron/history-of-lisp-parens

https://futureofcoding.org/catalog/

In the beginning

IBM 402
Plugboard
1948

Teletype
ASR-33
1963

IBM 026
Keypunch
1949

DEC
VT05
1970

Symbolics Lisp Machine 1980

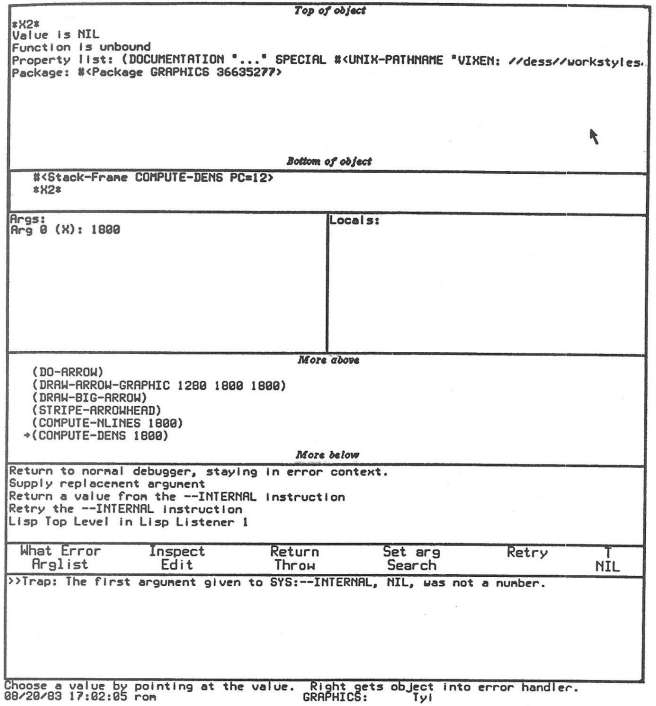
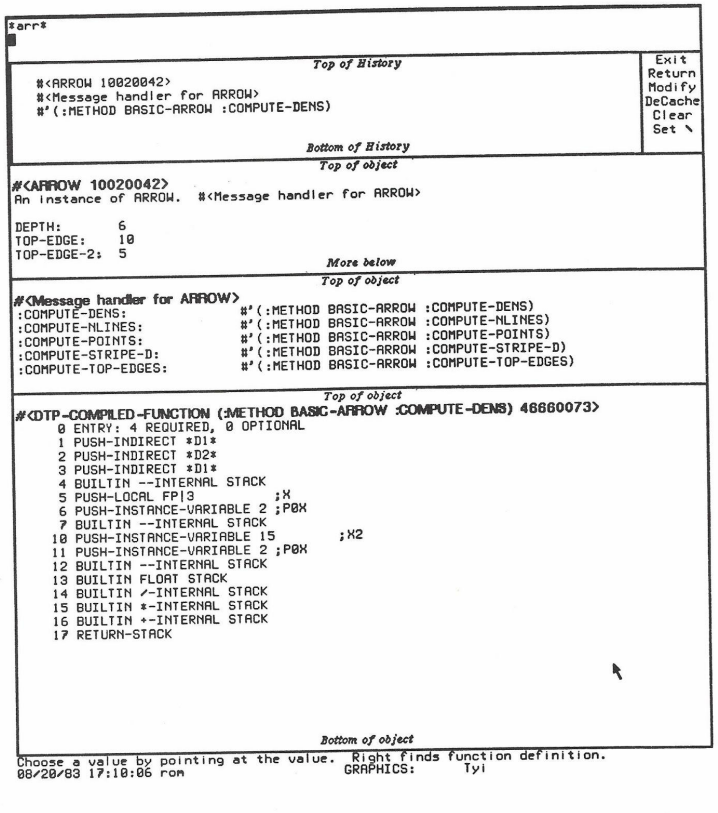
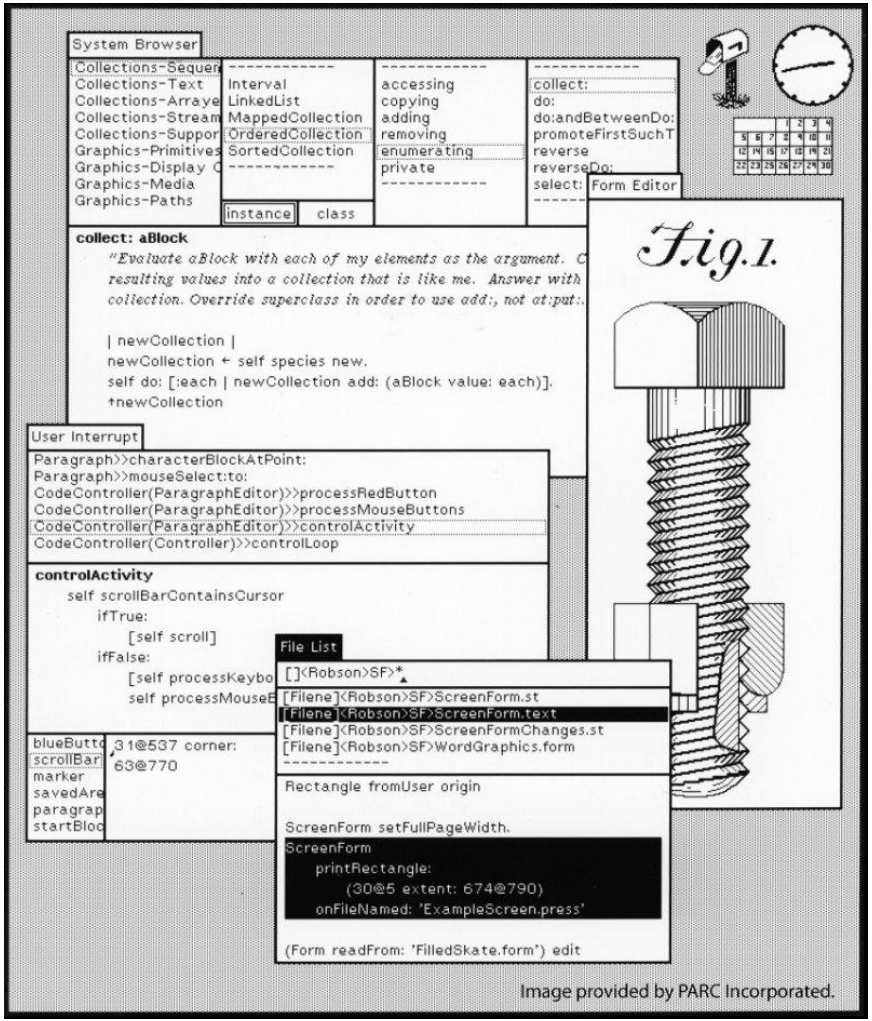


Figure 8. The Window Debugger: inspecting the variable *x2*.



Smalltalk-80



Self 1987

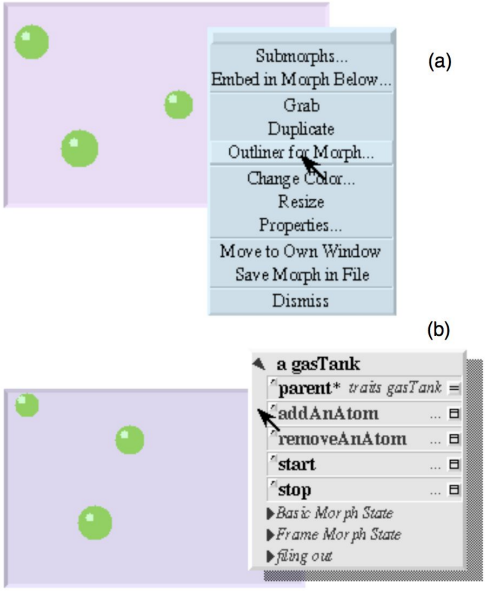


Figure 4. In a Self window, the user pops up the meta menu on the ideal gas simulation (a). Selecting "outliner" gives the Self-level representation to the user, which can be carried and placed as needed (b). (The italic items at the bottom of the outliner represent slot categories that may be expanded to view the slots. Unlike slots, categories have no language level semantics and are essentially a user interface convenience.)

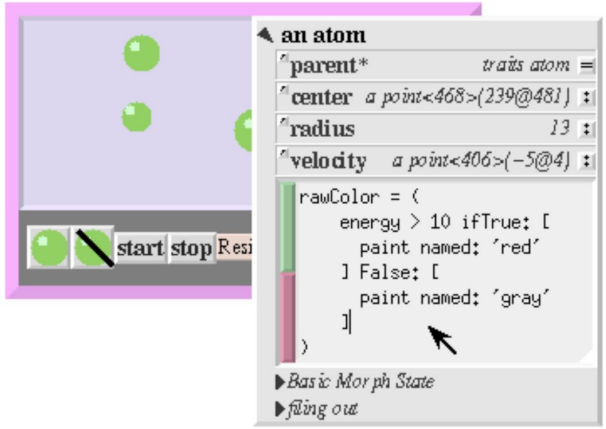


Figure 9. The user has selected one atom on which to experiment. The user changes the "rawColor" slot from a computed to a stored value by editing directly in the atom's outliner.

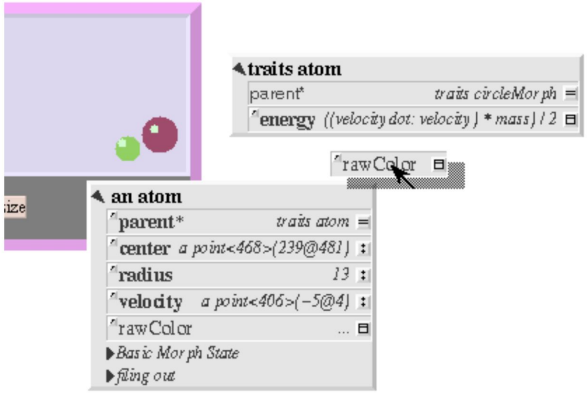
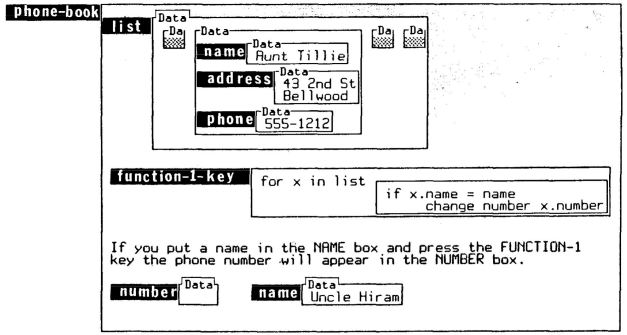
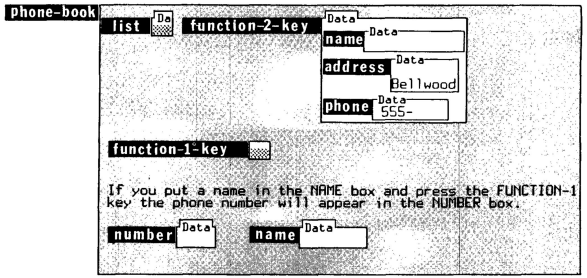


Figure 10. The user copies the modified rawColor slot as a first step in getting the computed method of coloring more widely shared. Because slots can be moved about readily, restructuring changes are relatively light weight, enhancing the sense of flexibility.

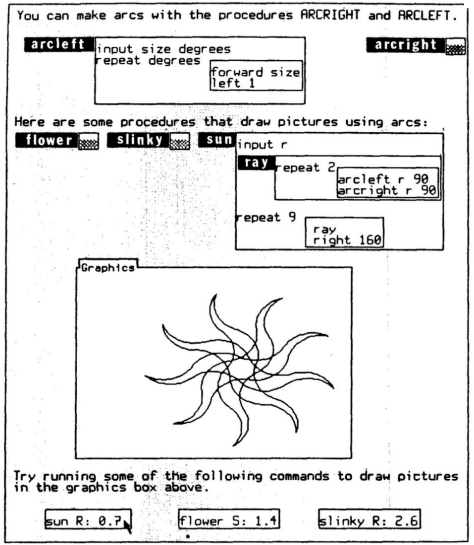
Boxer 1986



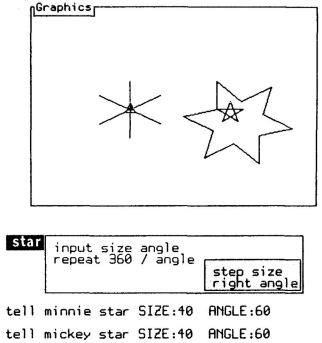
This box is a utility for finding phone numbers. Its local database is a box called LIST that contains subboxes with a standard format. The procedure named FUNCTION-1-KEY will be run whenever the FUNCTION-1 key is pressed.



When the FUNCTION-2 key is pressed, Boxer will insert at the cursor a template for a record with address and phone fields partially filled in that can be added to the LIST. In order to make this extension, the user needed only to type the template and label it with the name FUNCTION-2-KEY. (The template shown here is especially useful if you happen to know many people who live in Bellwood whose phone numbers begin with 555.)



Graphics appear inside regions of the screen called graphics boxes. The box labeled SUN defines the procedure that drew the design in the graphics box above. Definitions (boxes with name tabs on them) can be invoked anywhere within the box in which they appear. This scoping rule provides a general block-structuring capability that has been used here to include RAY as an internal subprocedure of SUN. The bottom line of the screen is a menu from which the user can select commands to be run. Here, as indicated by the small arrow (mouse cursor), the user has selected SUN with an input of 0.7. The letter R followed by a colon is a prompt generated by the system to indicate that the SUN procedure requires one argument, named R. Such prompts are generated automatically whenever users type the name of a defined procedure and press a help key. In general, any text on the screen can be modified or run.



This graphics box contains two sprites—Minnie (the small triangle) and Mickey (the small five-pointed star). Each sprite has been told to run the procedure called STAR. The designs drawn are different, because each sprite has its own definition of the STEP procedure when executing STAR.

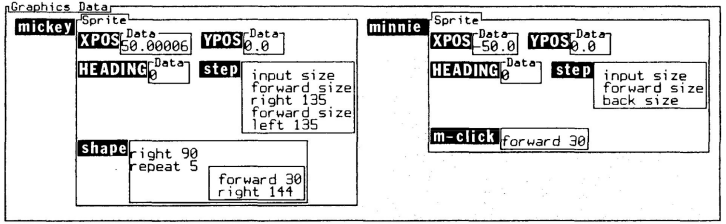
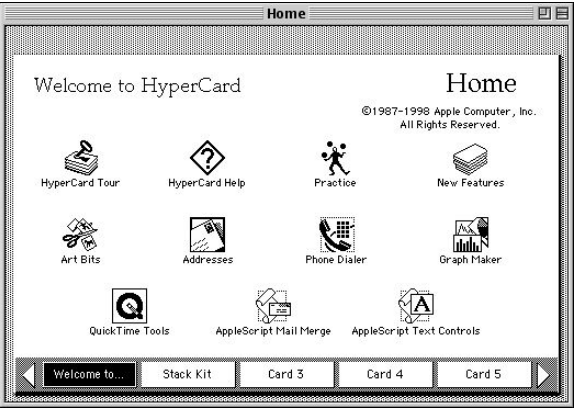
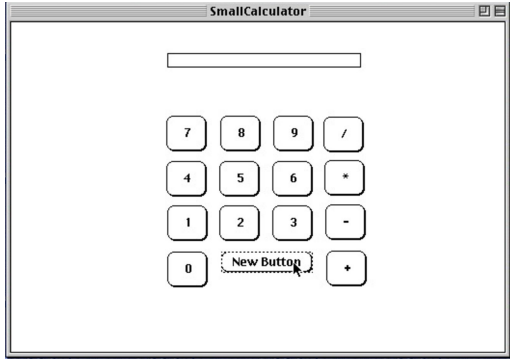


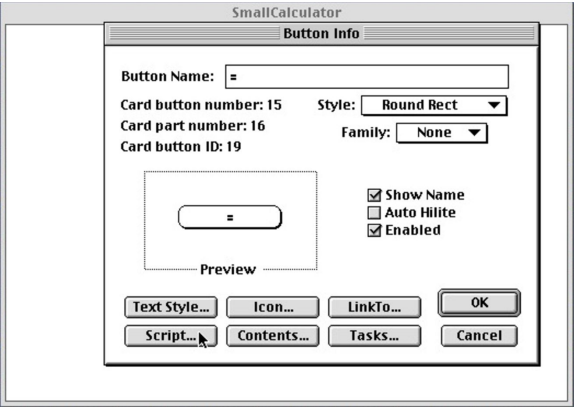
FIGURE 5a. A Graphics Box

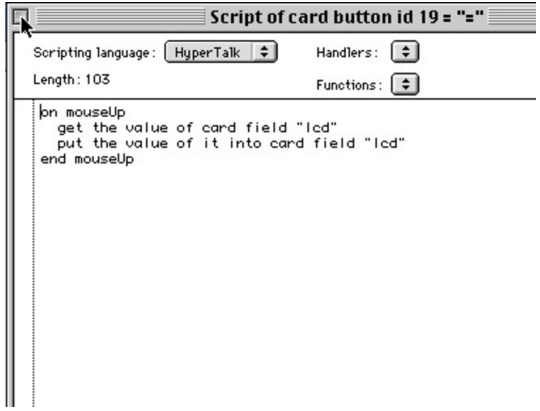
Here is the graphics box of Figure 5a, shown in data form so that all of its computational structure is visible, changeable, and extendable in ordinary Boxer textual format. The XPOS, YPOS, and HEADING variables show the position and heading of each sprite. In addition, Mickey has a SHAPE procedure that makes him appear as a five-pointed star; Minnie has no SHAPE procedure and hence appears in the default shape, a triangle.

HyperCard 1987









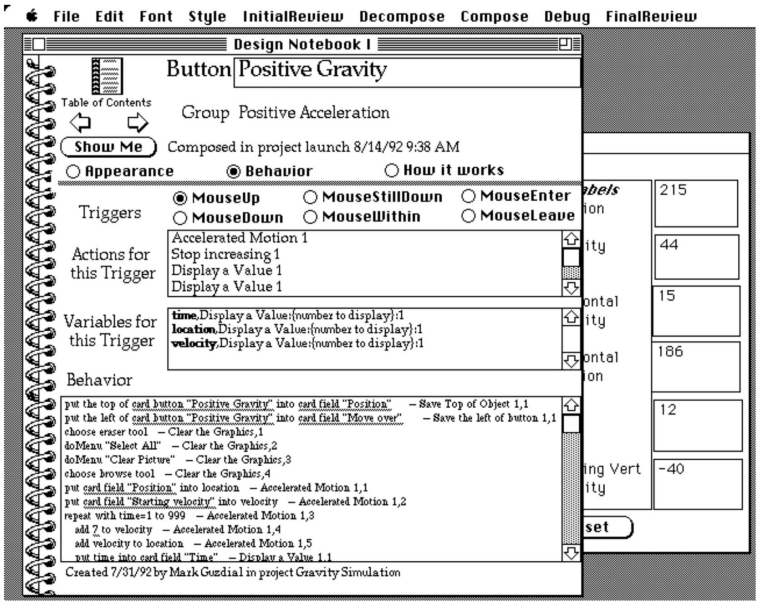


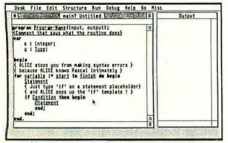
Fig. 23. An editor for the Positive Gravity Button. When the mouse goes up, Emile will execute 4 actions: Accelerated Motion 1, Stop Increasing 1, and Display a Value 1 (2 times). At the bottom of the screen, we can see the code that Emile will execute. Underlined text corresponds to parameters (or slots)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

Alice Pascal 1987

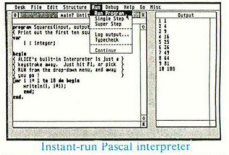
We were going to tell you all about

ALICE™
The Personal Pascal

but we'd rather show you . . .


You program from "templates"


Multiple window editing


Instant-run Pascal interpreter

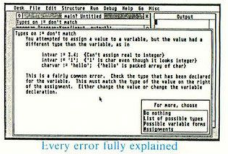

Over 700 help screens

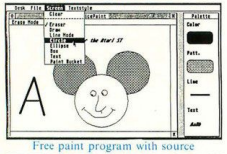
Programming Made Easy

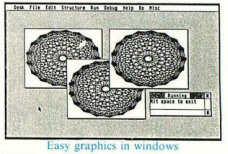
If you're into programming, or if you *want* to be, you won't find a better system for your Atari ST than ALICE: The Personal Pascal. Already popular on the IBM-PC, ALICE now brings easy, interactive programming to the Atari ST.

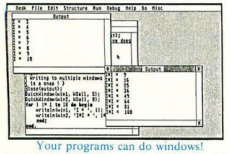
ALICE is, without question, the best way around to learn about computers and programming. Beginners can even order our ALICE Based textbook for only $19.95 with ALICE.

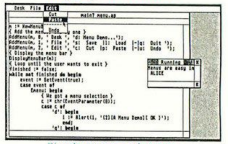
Even if you're an expert, ALICE makes it easier to write, test and especially *debug* programs. ALICE's extensive Pascal improvements include a GEM interface even beginners can use and most of the extensions of Turbo Pascal. The surrounding examples just give a glimpse at how easy it is to put programs together using ALICE.

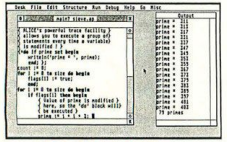

Every error fully explained

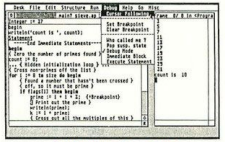

Free paint program with source

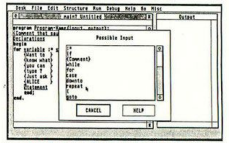

Easy graphics in windows


Your programs can do windows!


Simple menus and events


Powerful variable trace


Breakpoints, Single Step


Ask for possible input at any time

"If I needed to learn Pascal all over again, or were going to teach a course in the language, I can't imagine using any program other than ALICE."

— Adam Green, **Infoworld**

"ALICE may be the most advanced programming environment currently available for the PC."

— Michael Covington, **PC World**

"If you enjoy programming languages, this comes pretty darn close to being as much fun as a video game."

— Gene Wilburn, **Computing Canada**

Visit your dealer or send us $79.95 (USD) plus $5 for shipping and handling. VISA and MasterCard orders may phone collect. Dealer inquiries invited.

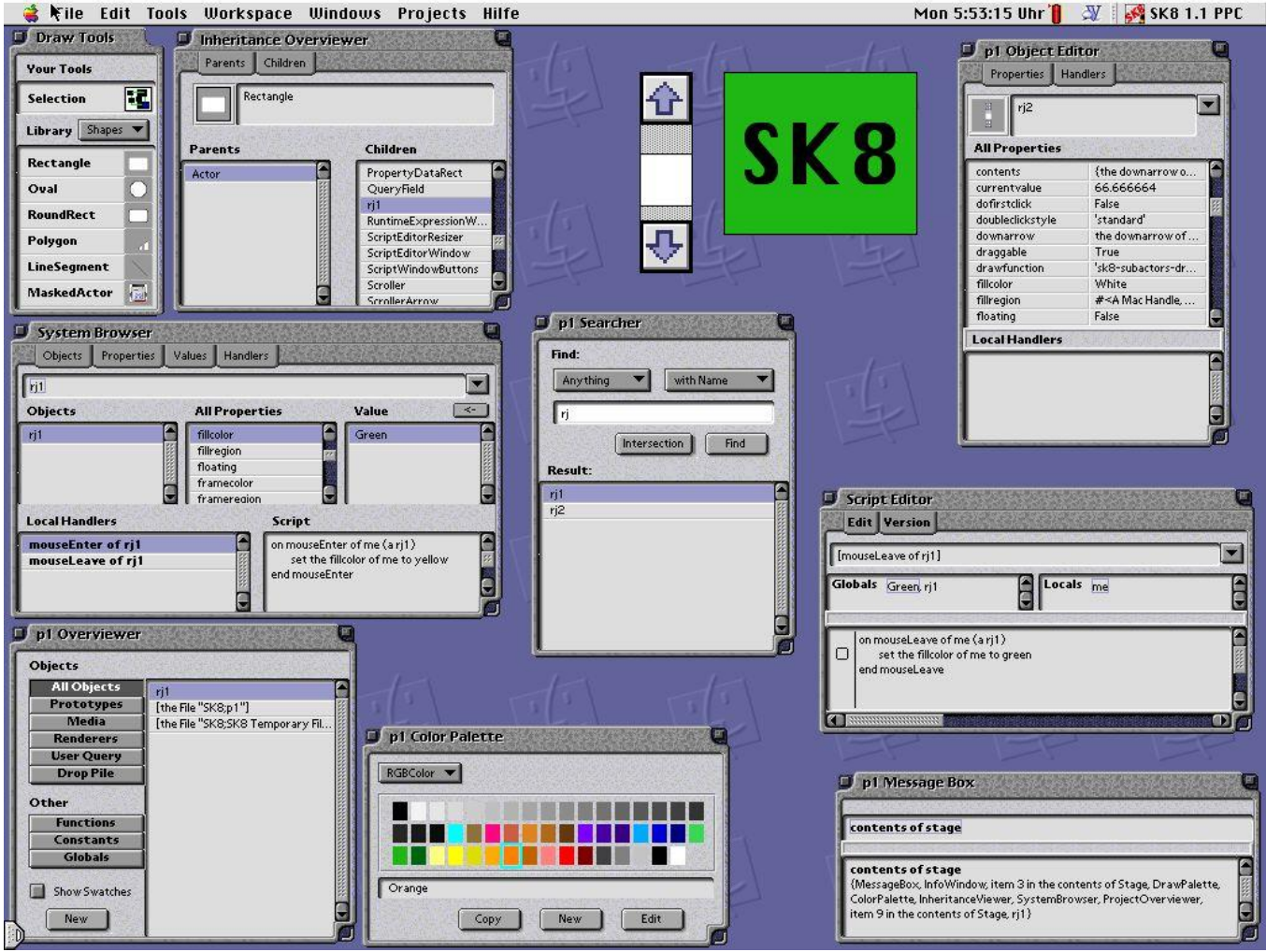
Looking Glass Software Limited
124 King St. N. Waterloo, Ontario
N2J 2X8
519/884-7473

Looking Glass Software

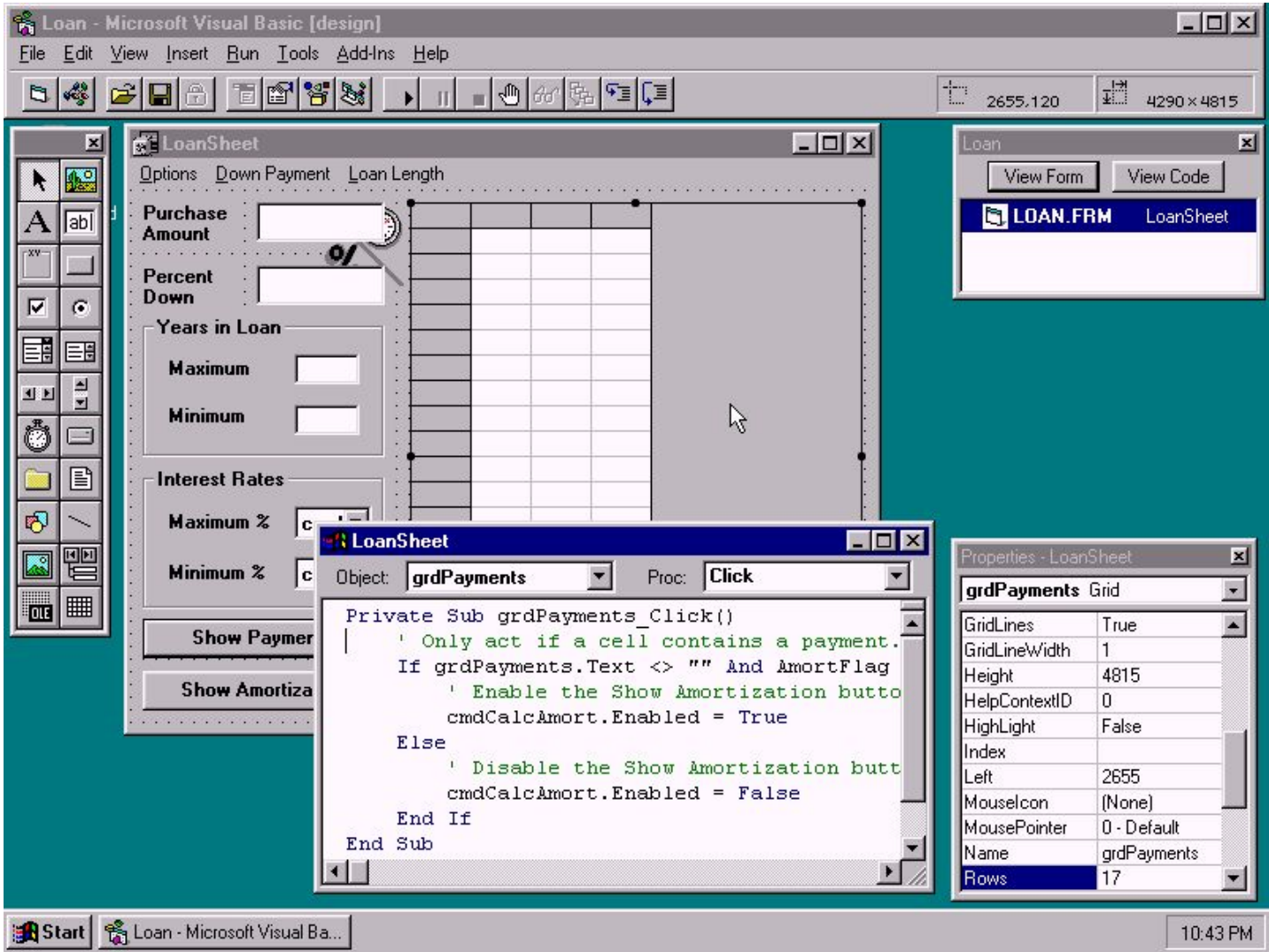
Trademarks: Turbo Pascal — Borland International, IBM-PC — IBM, Atari ST — Atari Corp., GEM — Digital Research Inc. ALICE refers to ALICE: The Personal Pascal, a trademark of Graham Software Corp., used with permission.

CIRCLE 027 ON READERS SERVICE CARD

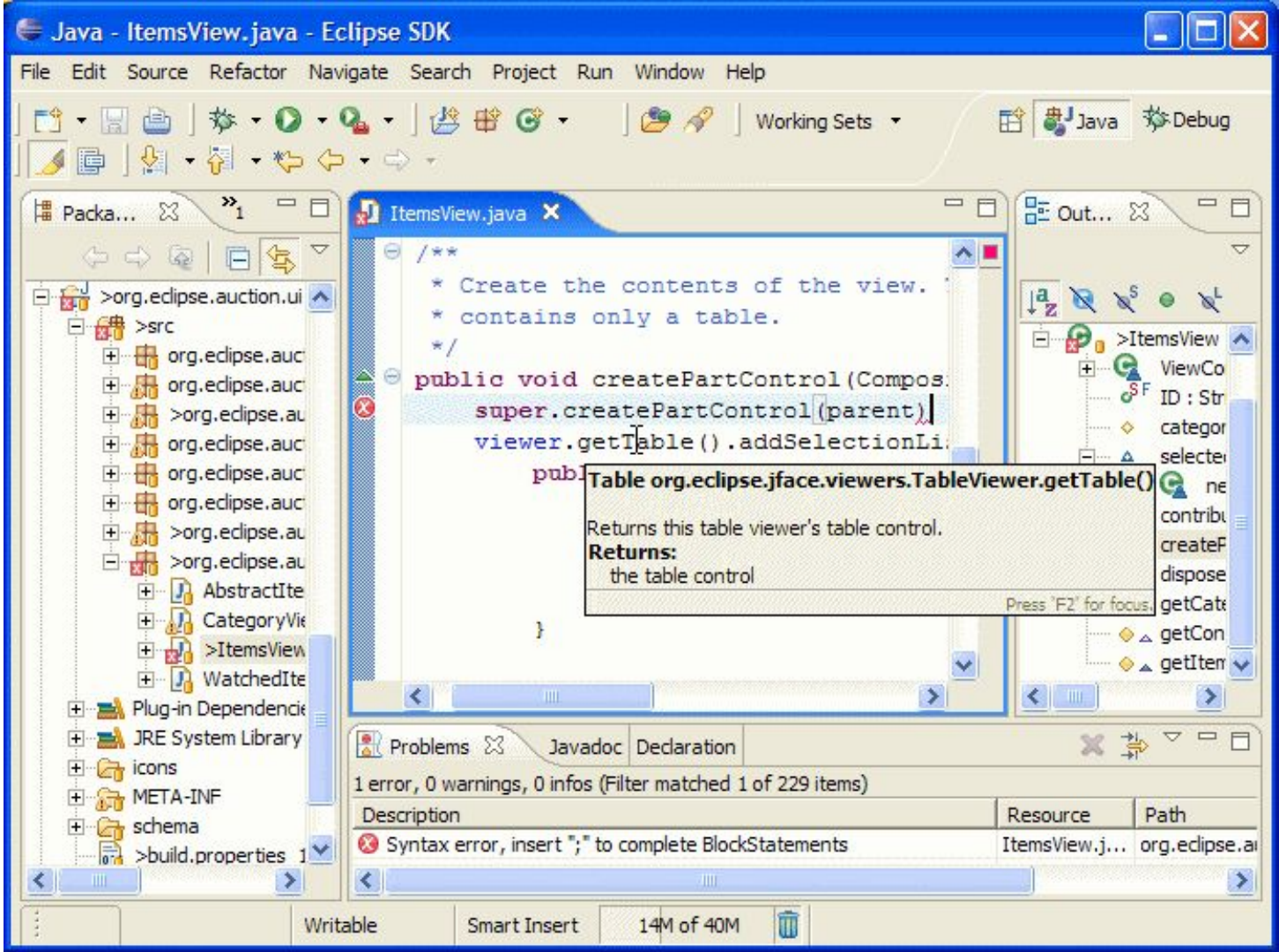
SK8 1994


File Edit Tools Workspace Windows Projects Hilfe
Mon 5:53:15 Uhr
SK8 1.1 PPC
Draw Tools
Your Tools
Selection
Library
Shapes
Rectangle
Oval
RoundRect
Polygon
LineSegment
MaskedActor
Inheritance Overviewer
Parents
Children
Rectangle
Parents
Actor
Children
PropertyDataRect
QueryField
rj1
RuntimeExpressionW...
ScriptEditorResizer
ScriptEditorWindow
ScriptWindowButtons
Scroller
SK8
p1 Object Editor
Properties
Handlers
rj2
All Properties
contents {the downarrow o...
currentvalue 66.666664
dofirstclick False
doubleclickstyle 'standard'
downarrow the downarrow of ...
draggable True
drawfunction 'sk8-subactors-dr...
fillcolor White
fillregion #<A Mac Handle, ...
floating False
Local Handlers
System Browser
Objects
Properties
Values
Handlers
rj1
Objects
All Properties
Value
rj1
fillcolor
fillregion
floating
framecolor
frameregion
Green
Local Handlers
Script
mouseEnter of rj1
mouseLeave of rj1
on mouseEnter of me (a rj1)
set the fillcolor of me to yellow
end mouseEnter
p1 Searcher
Find:
Anything
with Name
rj
Intersection
Find
Result:
rj1
rj2
Script Editor
Edit
Version
[mouseLeave of rj1]
Globals
Green, rj1
Locals
me
on mouseLeave of me (a rj1)
set the fillcolor of me to green
end mouseLeave
p1 Overviewer
Objects
All Objects
Prototypes
Media
Renderers
User Query
Drop Pile
Other
Functions
Constants
Globals
rj1
[the File "SK8:p1"]
[the File "SK8:SK8 Temporary Fil...
Show Swatches
New
p1 Color Palette
RGBColor
Orange
Copy
New
Edit
p1 Message Box
contents of stage
contents of stage
{MessageBox, InfoWindow, item 3 in the contents of Stage, DrawPalette, ColorPalette, InheritanceViewer, SystemBrowser, ProjectOverviewer, item 9 in the contents of Stage, rj1}

Visual Basic 1991



Eclipse
2001



Squeak/eToys
1996

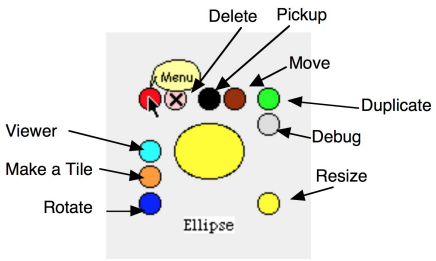
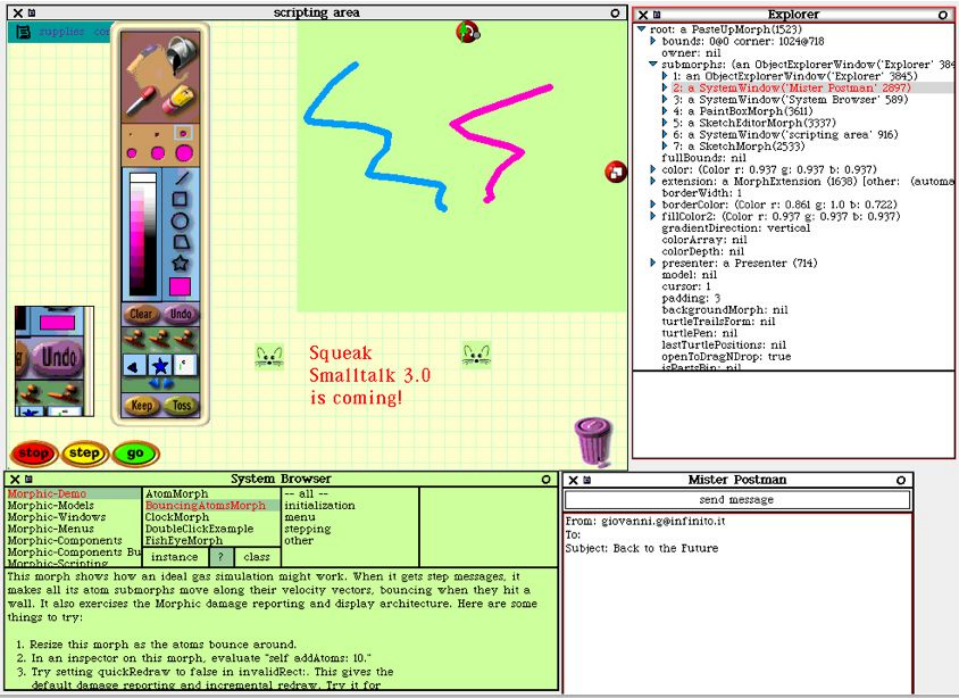
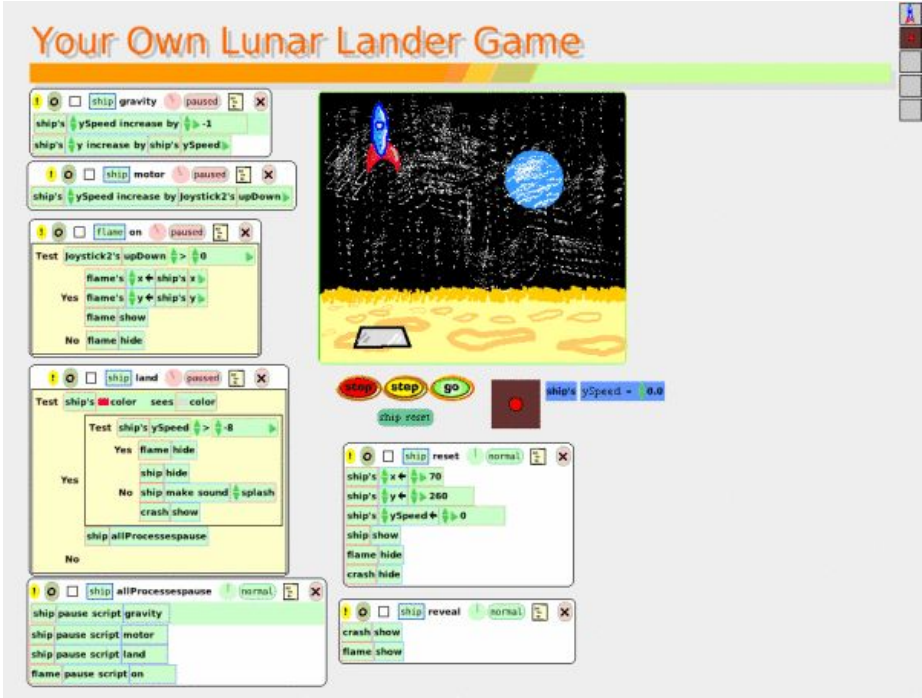
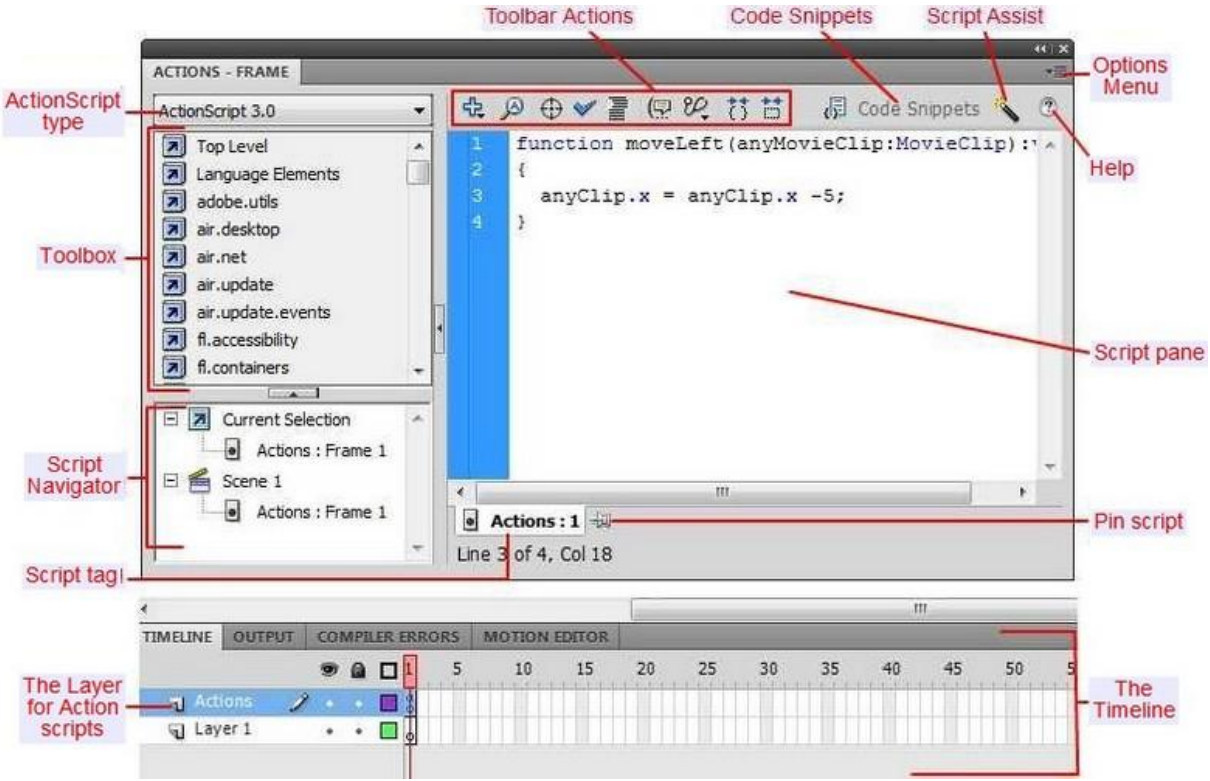


Figure 8: Standard Morphic Halos

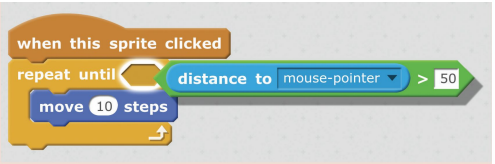


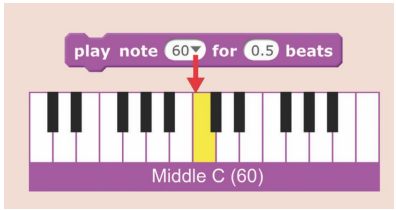


Flash MX
2003

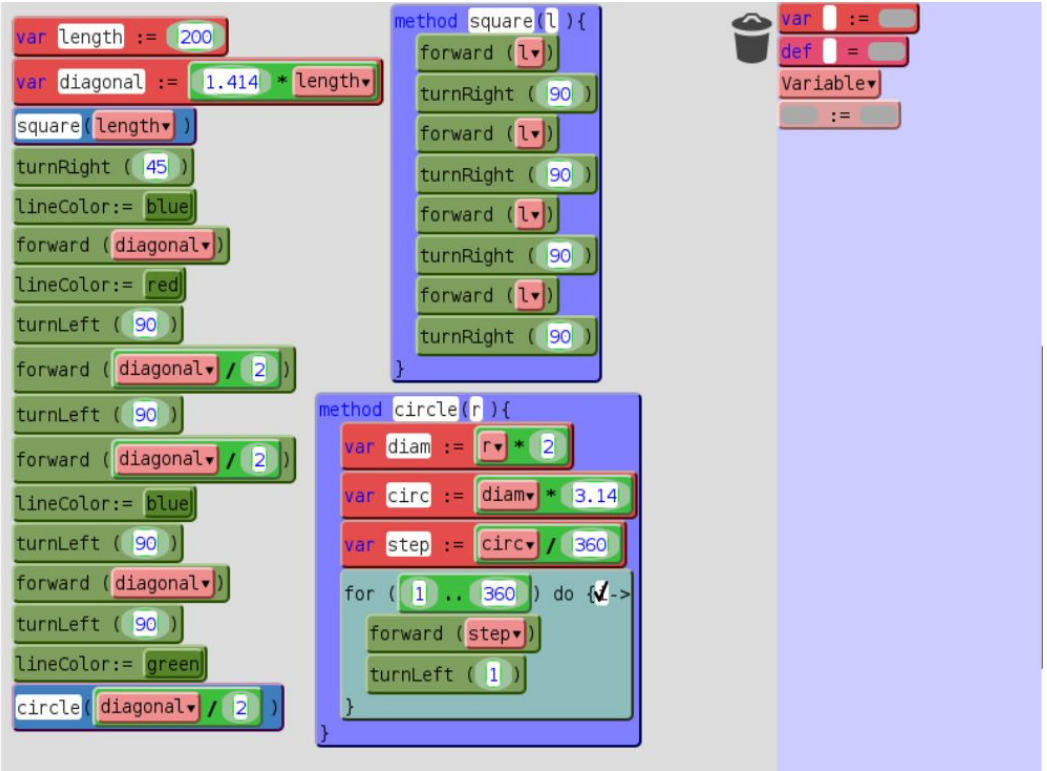


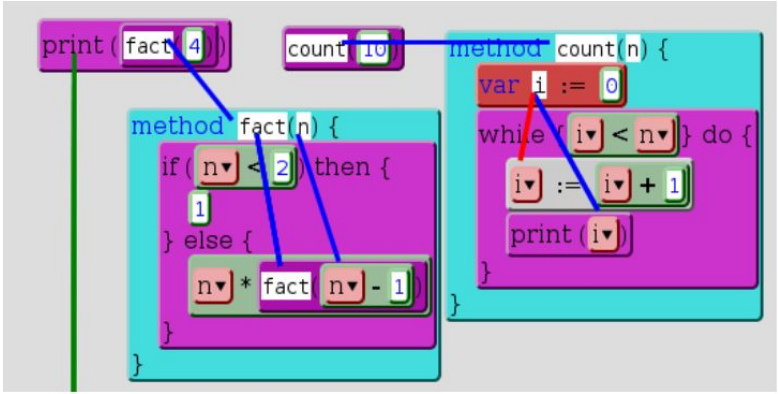
Scratch 2005

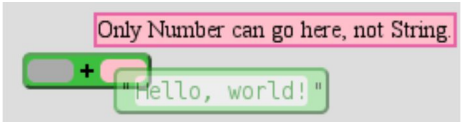




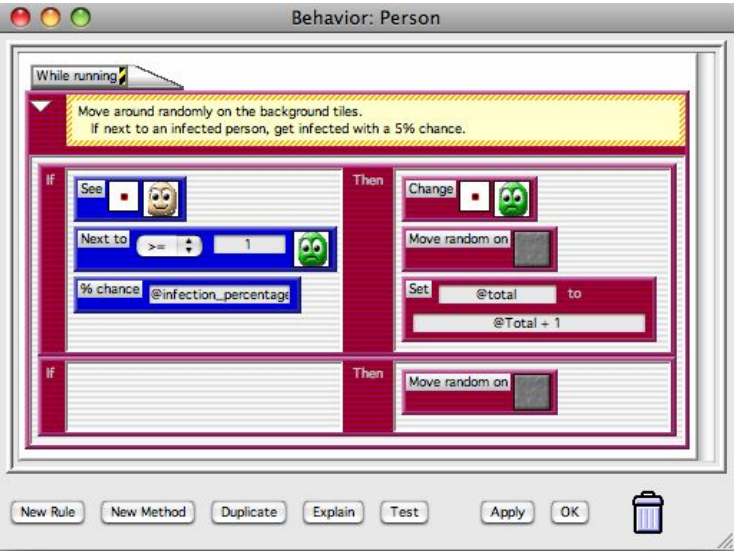
Tiled Grace 2014





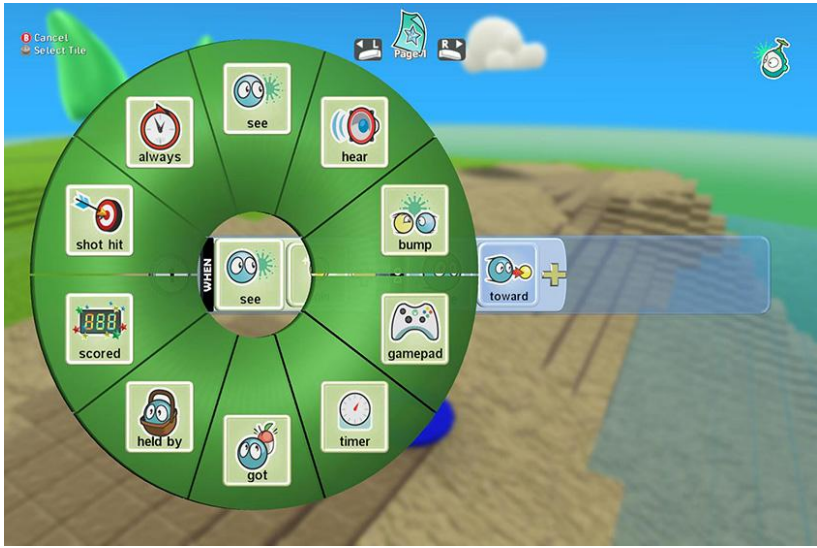


AgentSheets 1991, AgentCubes 2006

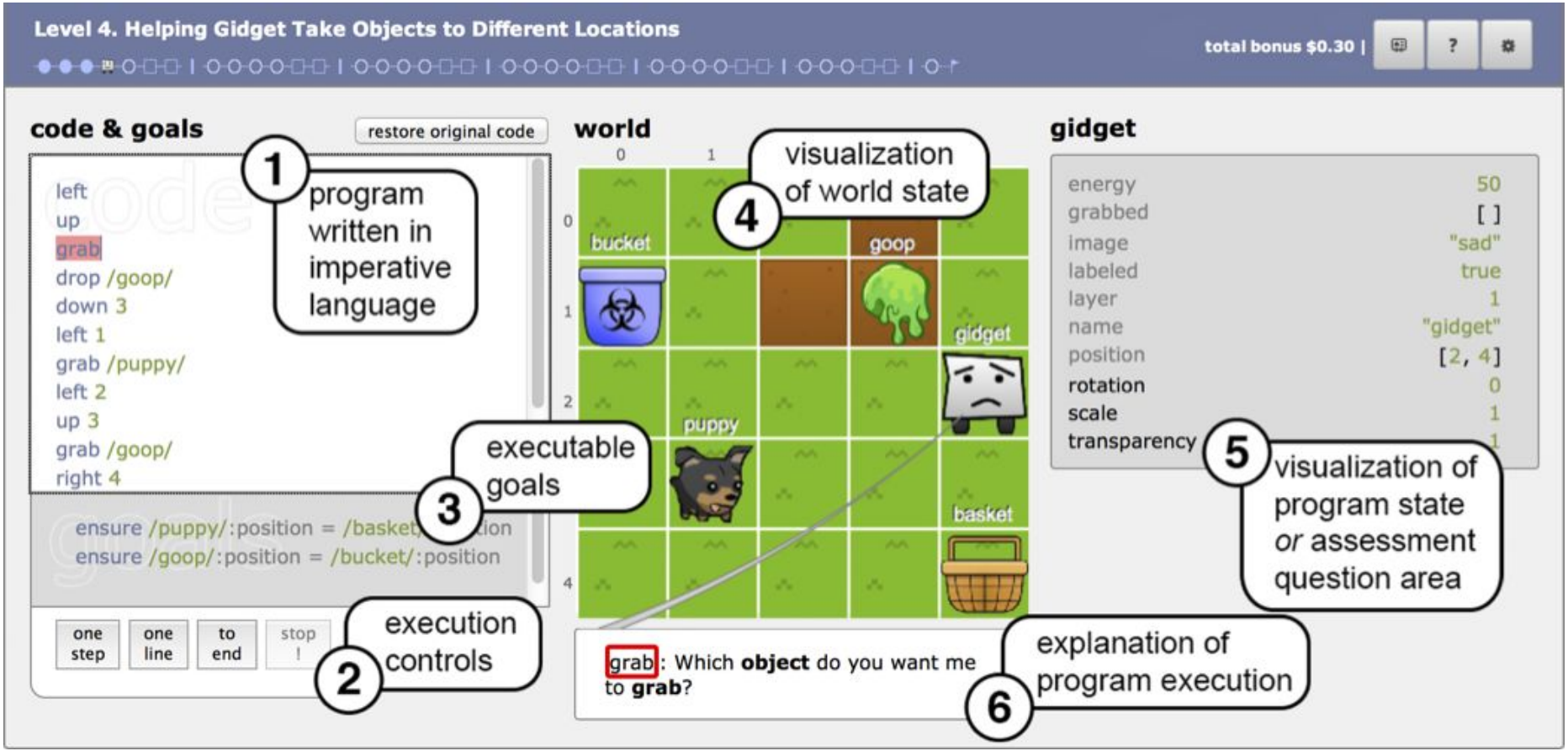


Kodu 2009

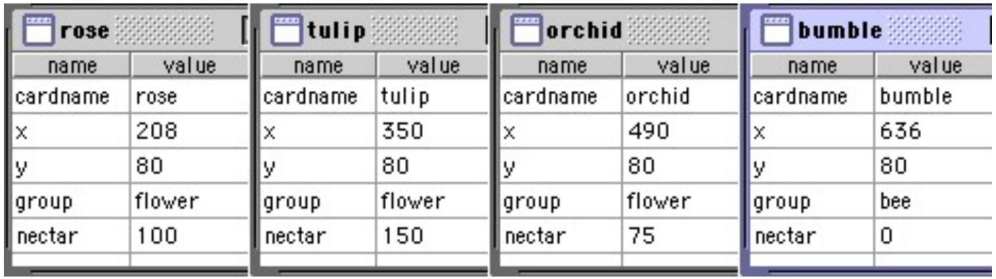

Kodu Game Lab (1.2.88.0, General)
Add/Change Tile
Delete Tile
Enter Add/Change Tile
Tab Next Page
Shift Tab Prev Page
Delete Delete Tile
Esc Return
Ctrl-P Print Kode
WHEN
DO
keyboard
Arrows
move
bump
apple
red
eat
bump
apple
blue
say


Cancel
Select Tile
Page 1
always
see
hear
shot hit
bump
WHEN
see
toward
scored
gamepad
held by
got
timer

Gidget 2014


Level 4. Helping Gidget Take Objects to Different Locations
total bonus $0.30 |
code & goals
restore original code
left
up
grab
drop /goop/
down 3
left 1
grab /puppy/
left 2
up 3
grab /goop/
right 4
ensure /puppy/:position = /basket/:position
ensure /goop/:position = /bucket/:position
1
program written in imperative language
3
executable goals
2
execution controls
one step
one line
to end
stop !
world
visualization of world state
4
bucket
goop
gidget
puppy
basket
grab: Which object do you want me to grab?
6
explanation of program execution
gidget
energy 50
grabbed []
image "sad"
labeled true
layer 1
name "gidget"
position [2, 4]
rotation 0
scale 1
transparency 1
5
visualization of program state or assessment question area

HANDS 2002



rose

name	value
cardname	rose
x	208
y	80
group	flower
nectar	100

tulip

name	value
cardname	tulip
x	350
y	80
group	flower
nectar	150

orchid

name	value
cardname	orchid
x	490
y	80
group	flower
nectar	75

bumble

name	value
cardname	bumble
x	636
y	80
group	bee
nectar	0

Figure 3. When the system evaluates the query *all flowers* it returns *orchid, rose, tulip*.

```
with all flowers
    set the item's nectar to random from 50 to 100
end with

with all flowers calling each the flower
    set the flower's nectar to random from 50 to 100
end with
```

- **all flowers** evaluates to **orchid, rose, tulip**
- **all bees** evaluates to **bumble**
- **all snakes** evaluates to **the empty list**
- **all (flower and (nectar < 100))** evaluates to **orchid**
- **set the nectar of all flowers to 0**
- **FirstItem of all flowers** evaluates to **orchid**
- **GreatestItem in nectar of all flowers** evaluates to **150**
- **Sorted nectar of all flowers** evaluates to **75,100,150**
- **CardWithLeast nectar of all flowers** evaluates to **orchid**
- **Sum nectar of all flowers** evaluates to **325**

```
• if f=tulip then                    // if-then-else style
    set f's nectar to 0
    otherwise set f's nectar to 100
end if
• if f=...                           // case statement style
        tulip then set f's nectar to 0
        orchid then set f's nectar to 10
        otherwise set f's nectar to 20
end if
• if                                 // cond style
        f's nectar > 30 then set f's nectar to 0
        f's nectar < 30 then set f's nectar to 10
        otherwise set f's nectar to 20
end if
```

Barista 2006

Figure 10. On top, a "match form" representation of a Boolean expression [15]; on bottom is the more editable view that is shown when the caret is in the expression.

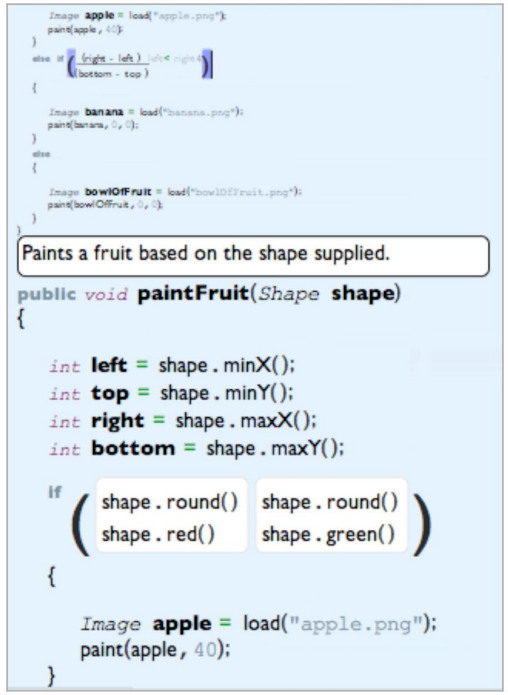


Figure 11. A focus + context interaction that allows users to partially collapse blocks of Java code.

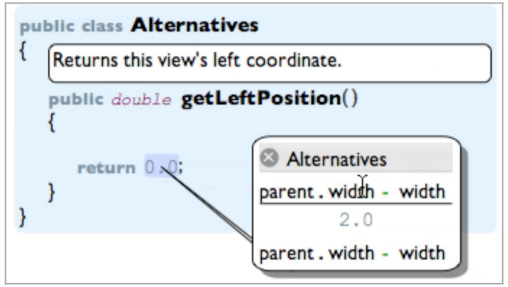
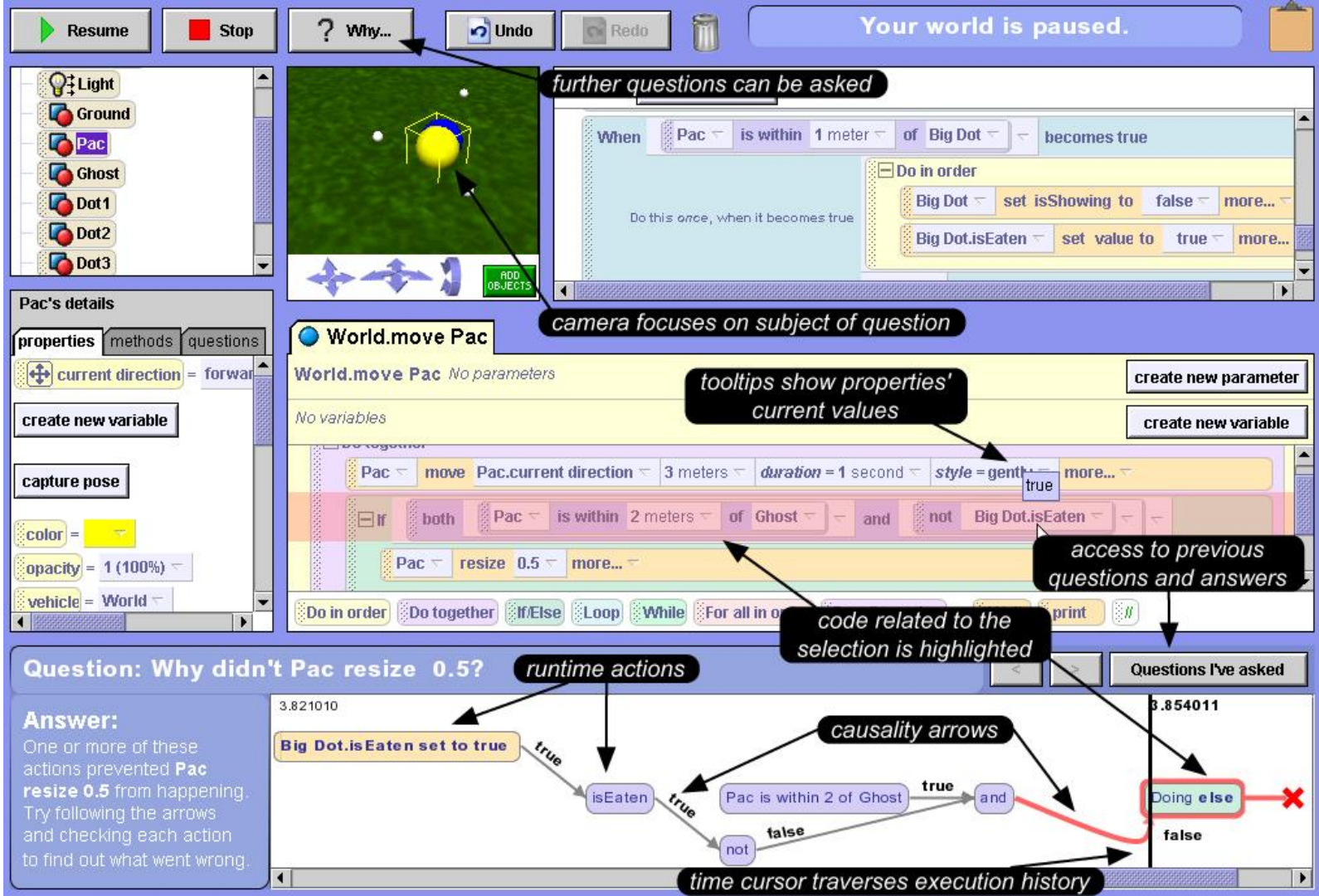


Figure 12. A tool for managing alternative expressions, helping users track and explore different solutions.

[Whyline](#)
2009



InterState 2014

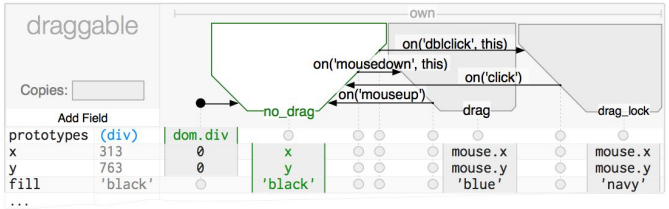


Figure 1: A basic InterState object, named `draggable`, which implements draggable and drag lock behaviors. Properties that control `draggable`'s display are represented as rows (e.g. `x`, `y`, and `fill`). States and transitions are represented as columns (e.g. `no_drag` and `drag`). An entry in a property's row for a particular state specifies a constraint that controls that property's value in that state. Here, while `draggable` is in the `drag` state, `x` and `y` will be constrained to `mouse.x` and `mouse.y` respectively, meaning `draggable` will follow the mouse.

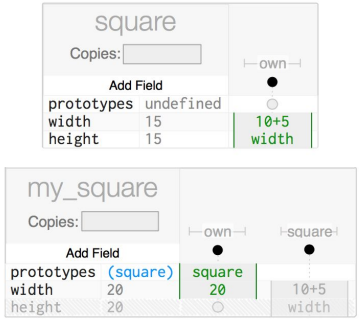


Figure 3: InterState uses a prototype-instance inheritance model with multiple inheritance. Prototypes are simply specified in the `prototypes` property. Here, `my_square` inherits from `square`. Because `my_square` does not define a value for `height`, it inherits the definition of `square.height`, as indicated by the greyed out text in the columns on the right. Note that `my_square` inherits the *definition* of `height`, not the value. Thus, its `width` property evaluates to a different value (`20`) than it does in `square` (`15`).

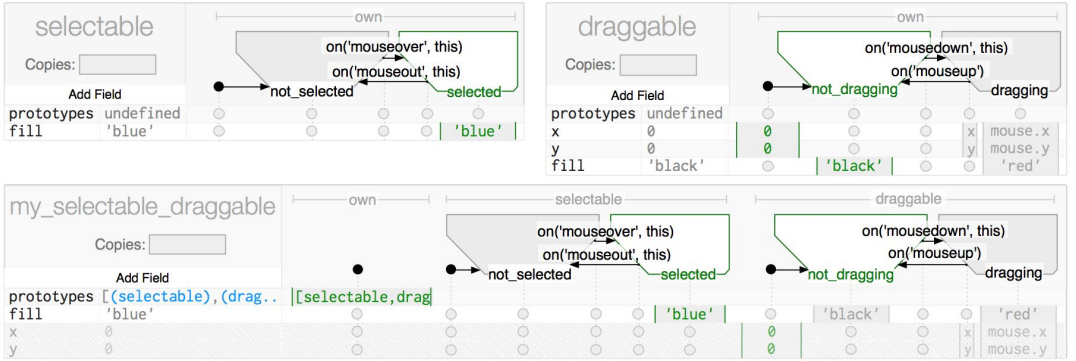


Figure 4: An object that inherits from both `draggable` and `selectable` behaviors. Note that the definitions for the `color` property are inherited from `draggable` (`'red'`) and `selectable` (`'blue'`).

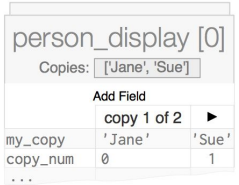


Figure 5: An object with multiple copies; `copies` is set to `['Jane', 'Sue']`. Every copy has two properties: `my_copy`, which is set to that copy's item (here, either `'Jane'` or `'Sue'`) and `copy_num`, which is set to that copy's index. Here, we are looking at the first copy.

Figure 6: Syntax and runtime errors are highlighted in the editor but do not prevent the program from running.

Euklas 2014

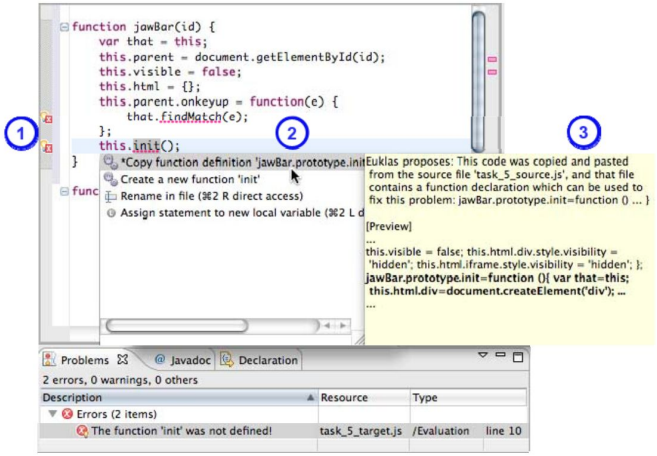


Figure 1. Euklas supports peoples' integration attempts as it enhances Eclipse's JavaScript editor by (#1) highlighting errors in the source code, (#2) providing quick fixes including the use of the context in the origin of any copied code, and (#3) including explanations for copy-and-paste errors based on the code of the example snippet that was used.

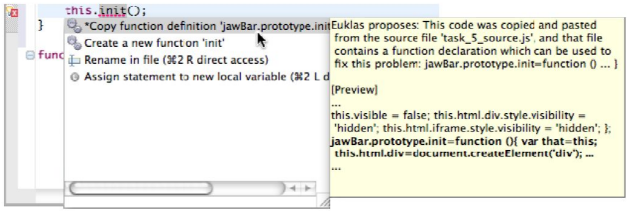


Figure 3. Euklas proposes the top two quick fixes for the undefined function init(). More information about the first (selected) proposed fix is shown in the beige pop-up at the right. The pop-up shows a short proposal explaining what Euklas intends to do, followed by a preview of how the code will look after the selected fix has been applied. Euklas augments Eclipse's quick fix feature which is well-known by many programmers.

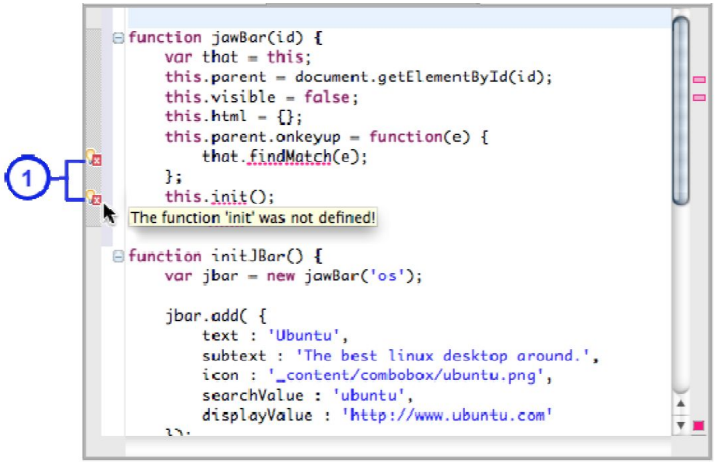


Figure 2. Euklas marks errors in the function jawBar() after Jamie pastes it into the target file.

Linked Editing

2004

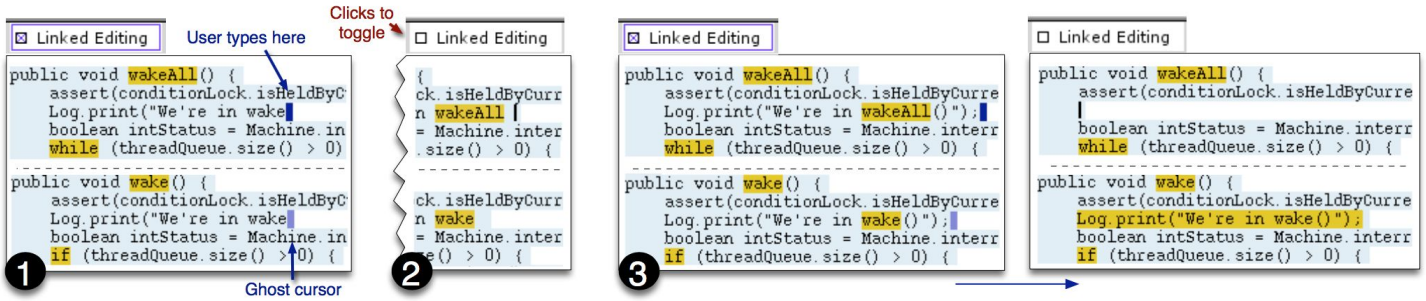


Figure 2. (1) Adding a line to two clones. (2) Modifying one instance. (3) Deleting line in one instance.

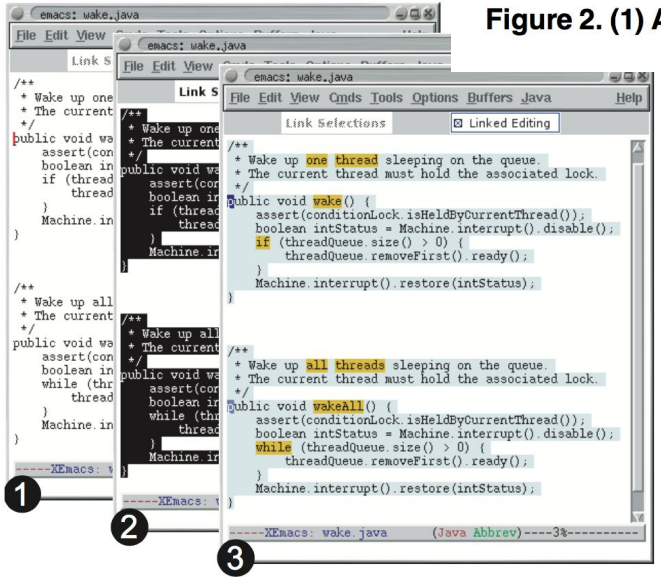


Figure 1. (1) Before linking two similar Java methods in Codelink. (2) After selecting the methods. (3) After linking the methods.

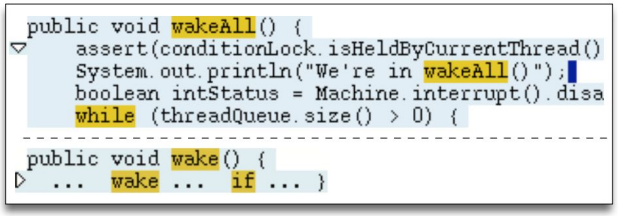
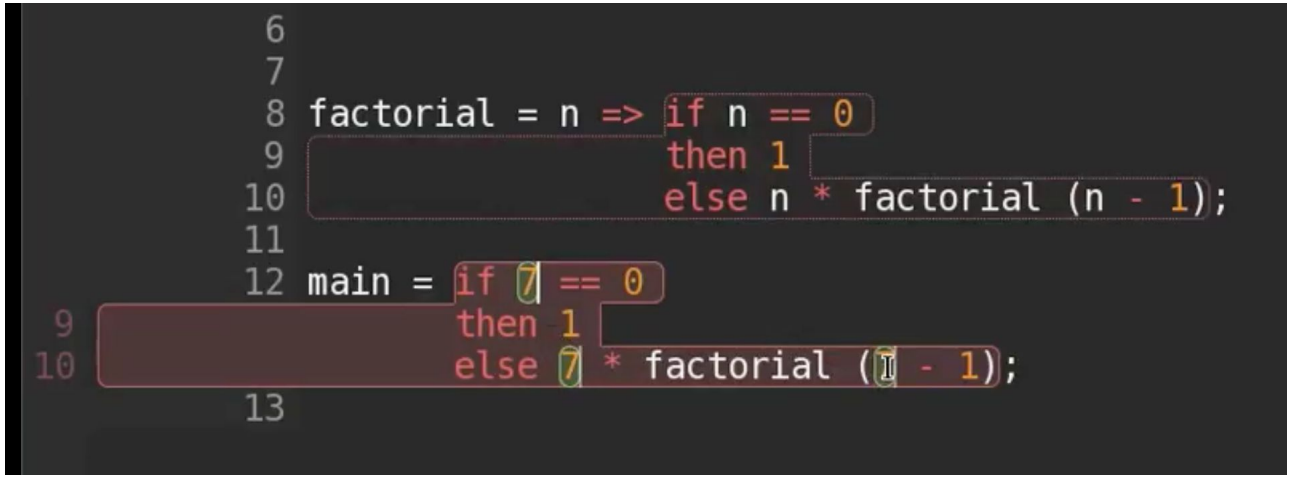
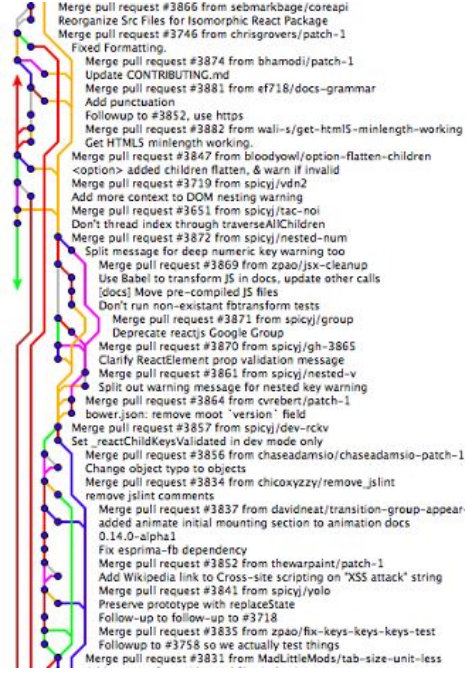


Figure 3. An elided clone looks similar to a function definition and use

Code portals 2016



versions & diffs




```
876
877       default:
878         return undefined;
879     }
880   }
881
882   /** visit slots and values top-down outside of derived values */
883   visitStatic(visitor: (x: SlotOrValue) => void) {
884     visitor(this);
885     if (this.metadata) {
886       assert(this.metadata.inSlot === this);
887       assert(this.metadata.version === this.version);
888       this.metadata.visitStatic(visitor);
889     }
890     if (this.derived) {
891       // don't visit value to allow recursion
892       if (this.source) {
893         // bind source
894         this.bind();
895       }
896       return;
897     }
898     assert(this.value.inSlot === this);
899     assert(this.value.version === this.version);
900     this.value.visitStatic(visitor);
901   }
902
903   equals(other: SlotOrValue): boolean {
904     // FIXME: skip formulas?
905     return other instanceof Slot && this.value.equals(other.value);
```

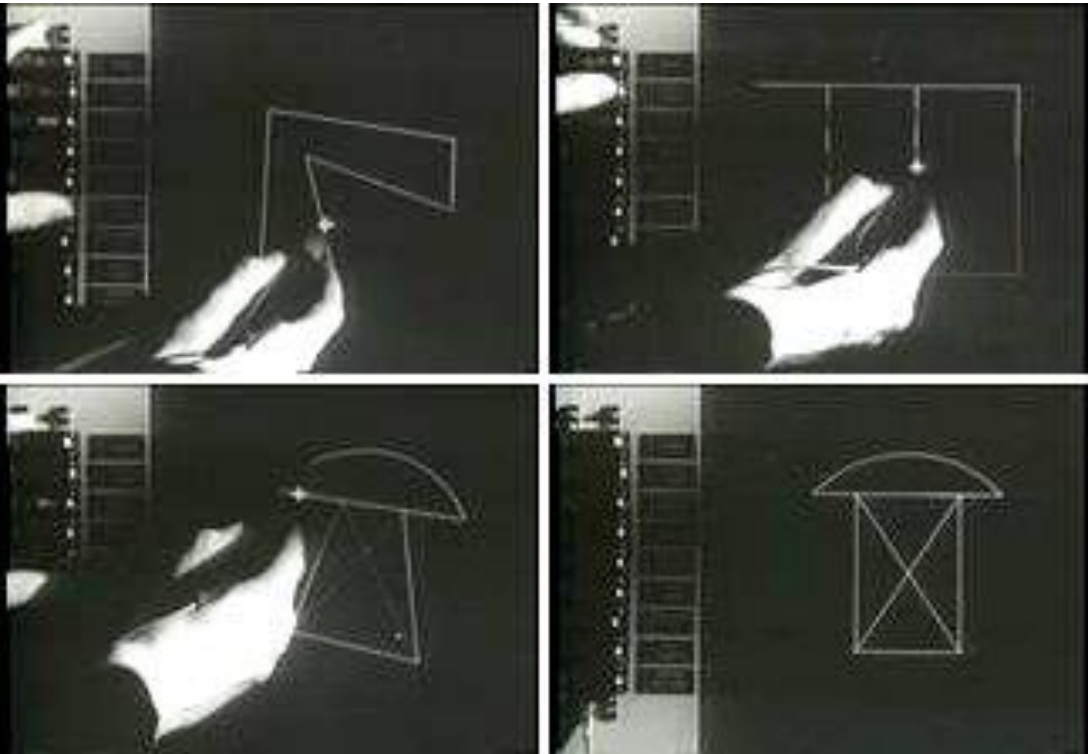
```
1004
1005      default:
1006        return undefined;
1007    }
1008  }
1009
1010  // analyze slots and values top-down outside of derived values */
1011  analyze() {
1012    if (this.metadata) {
1013      assert(this.metadata.inSlot === this);
1014      assert(this.metadata.version === this.version);
1015      this.metadata.analyze();
1016    }
1017    if (this.derived) {
1018      if (this.source) {
1019        // bind source
1020        this.bind();
1021        // but don't instantiate
1022        return;
1023      }
1024      // instantiate calls and conditionals to analyse them
1025      this.value;
1026      return;
1027    }
1028    assert(this.value.inSlot === this);
1029    assert(this.value.version === this.version);
1030    this.value.analyze();
1031  }
1032
1033  equals(other: SlotOrValue): boolean {
1034    // FIXME: skip formulas?
1035    return other instanceof Slot && this.value.equals(other.value);
```

```
872     switch (name) {
873       case MetaID.prototypeField.name:
874         let value = this.value;
875         return value instanceof List ? value.prototypeField :
    undefined;
876
877       default:
878         return undefined;
879     }
880   }
881
882   /** visit slots and values top-down outside of derived values
    */
883   visitStatic(visitor: (x: SlotOrValue) => void) {
884     visitor(this);
885     if (this.metadata) {
886       assert(this.metadata.inSlot === this);
887       assert(this.metadata.version === this.version);
888       this.metadata.visitStatic(visitor);
889     }
890     if (this.derived) {
891       // don't visit value to allow recursion
892       if (this.source) {
893         // bind source
894         this.bind();
895       }
896       return;
897     }
898     assert(this.value.inSlot === this);
899     assert(this.value.version === this.version);
900     this.value.visitStatic(visitor);
901   }
902
903   equals(other: SlotOrValue): boolean {
904     // FIXME: skip formulas?
```

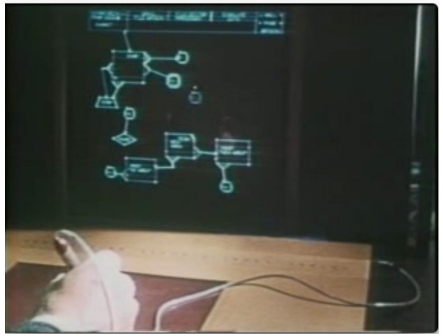
```
1000    switch (name) {
1001      case MetaID.prototypeField.name:
1002        let value = this.value;
1003        return value instanceof List ? value.prototypeField :
    undefined;
1004
1005      default:
1006        return undefined;
1007    }
1008  }
1009
1010  // analyze slots and values top-down outside of derived values
    */
1011  analyze() {
1012    if (this.metadata) {
1013      assert(this.metadata.inSlot === this);
1014      assert(this.metadata.version === this.version);
1015      this.metadata.analyze();
1016    }
1017    if (this.derived) {
1018      if (this.source) {
1019        // bind source
1020        this.bind();
1021        // but don't instantiate
1022        return;
1023      }
1024      // instantiate calls and conditionals to analyse them
1025      this.value;
1026      return;
1027    }
1028    assert(this.value.inSlot === this);
1029    assert(this.value.version === this.version);
1030    this.value.analyze();
1031  }
1032
```

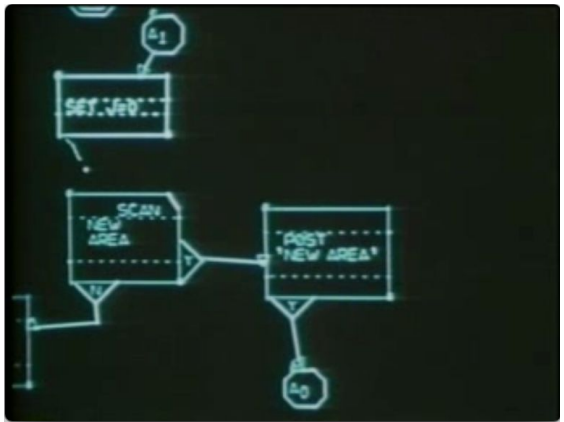
```
878         return undefined;
879     }
880   }
881
A 882   /** visit slots and values top-down outside of derived values */
883   visitStatic(visitor: (x: SlotOrValue) => void) {
  B 1010  // analyze slots and values top-down outside of derived values */
1011    analyze() {
A 884     visitor(this);
885     if (this.metadata) {
886       assert(this.metadata.inSlot === this);
887       assert(this.metadata.version === this.version);
A 888     this.metadata.visitStatic(visitor);
  B 1015    this.metadata.analyze();
889     }
890     if (this.derived) {
A 891     // don't visit value to allow recursion
892       if (this.source) {
893         // bind source
894         this.bind();
  B 1021    // but don't instantiate
1022      return;
895       }
  B 1024    // instantiate calls and conditionals to analyse them
1025      this.value;
896       return;
897     }
898     assert(this.value.inSlot === this);
899     assert(this.value.version === this.version);
A 900     this.value.visitStatic(visitor);
  B 1030    this.value.analyze();
901   }
902
903   equals(other: SlotOrValue): boolean {
```

Sketchpad
1963



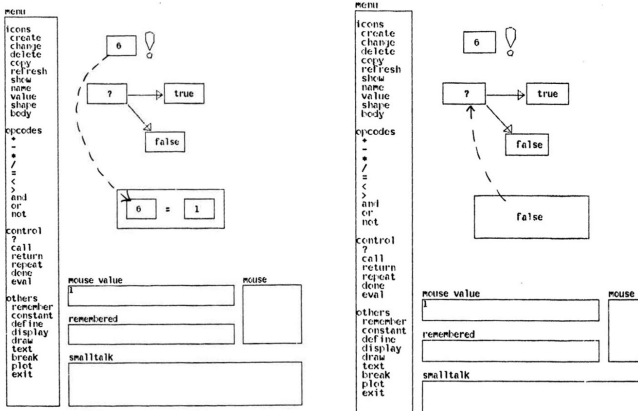
GRAIL
1969





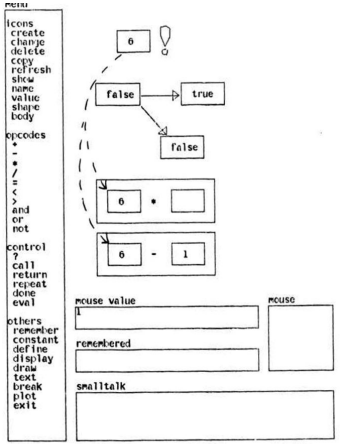
PYGMALION

1975

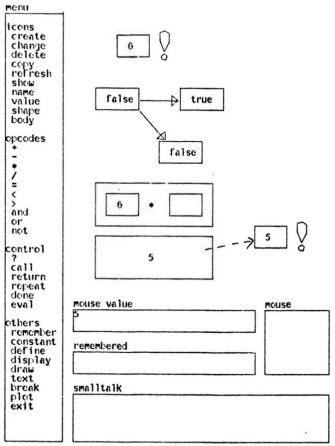


The argument to FACTORIAL is tested against the constant 1

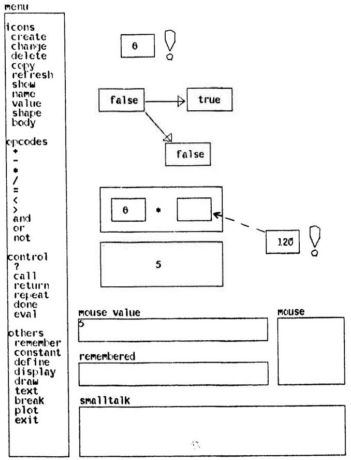
6 ≠ 1



The "false" icon causes 6 ≈ FACTORIAL(6 - 1) to be computed



Preparing for a recursive call on FACTORIAL



FACTORIAL(5) = 120 (The intermediate steps have not been shown.)

Thinglab 1979

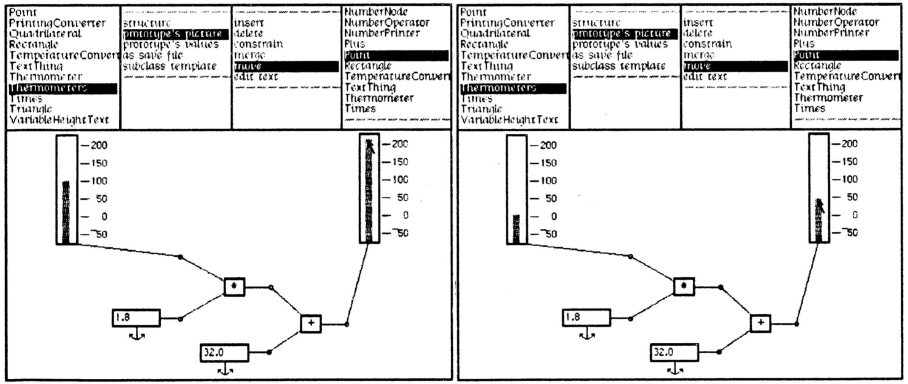


Figure 2.18 - The temperature converter with thermometers for input and output

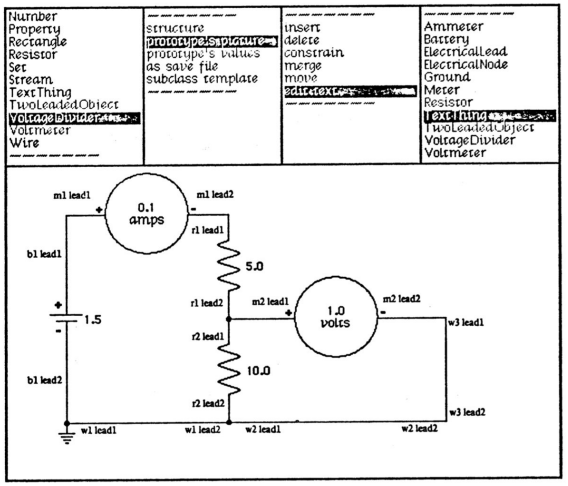


Figure 5.2 - A voltage divider

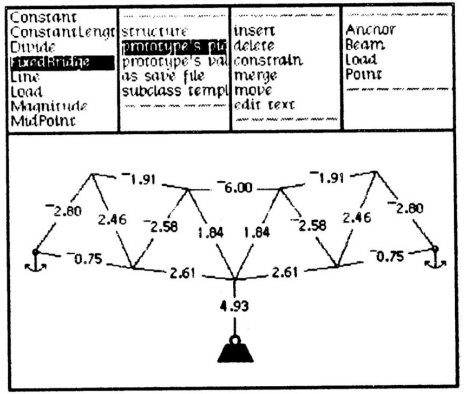
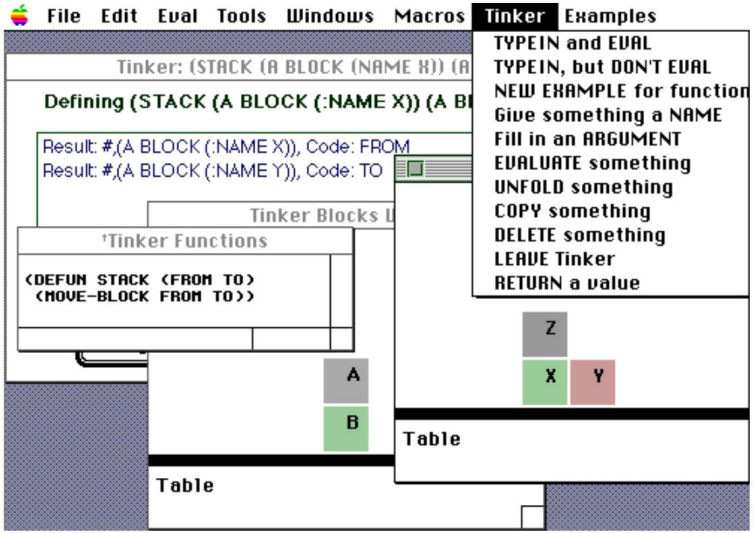
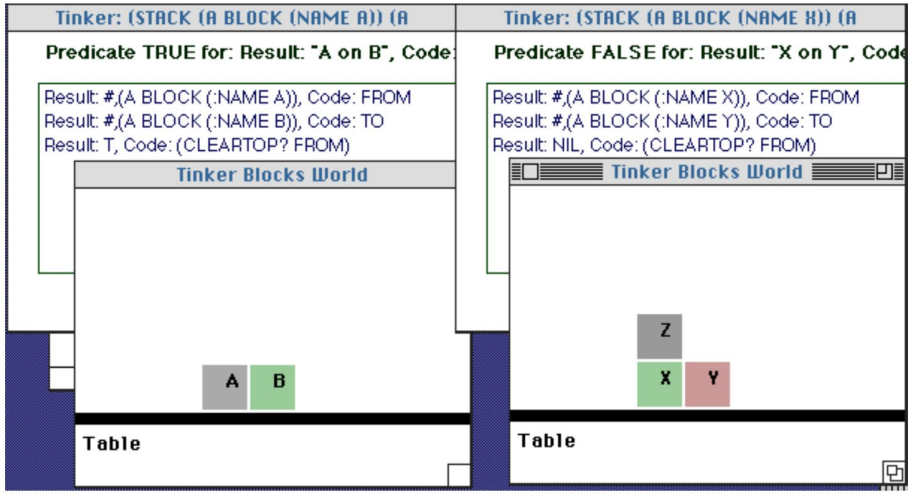
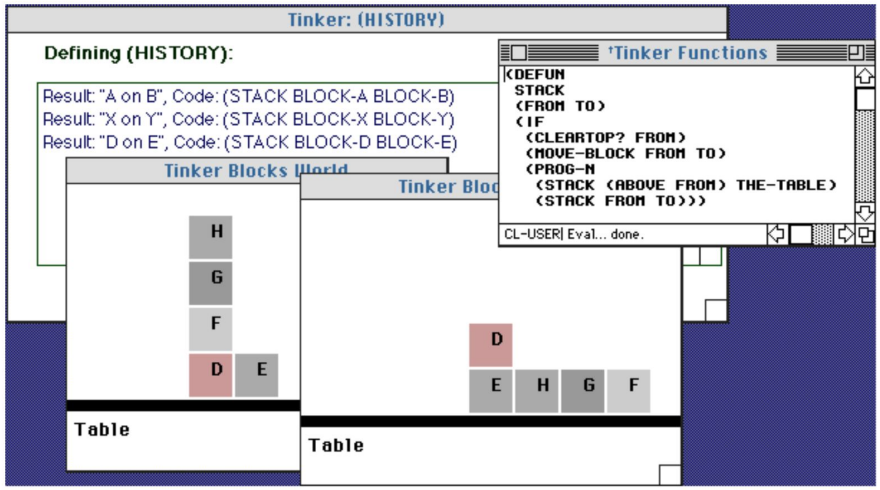


Figure 2.31 - A bridge under load

Tinker 1981

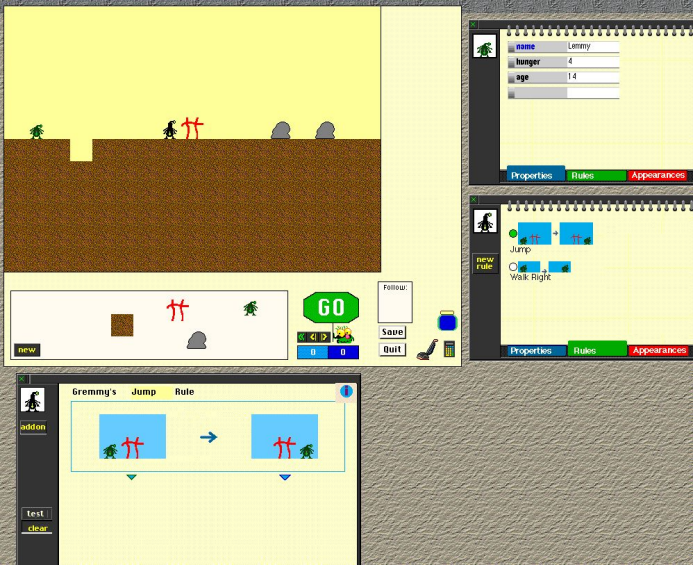

File Edit Eval Tools Windows Macros Tinker Examples
TYPEIN and EVAL
TYPEIN, but DON'T EVAL
NEW EXAMPLE for function
Give something a NAME
Fill in an ARGUMENT
EVALUATE something
UNFOLD something
COPY something
DELETE something
LEAVE Tinker
RETURN a value
Tinker: (STACK (A BLOCK (NAME X)) (A
Defining (STACK (A BLOCK (:NAME X)) (A BL
Result: #,(A BLOCK (:NAME X)), Code: FROM
Result: #,(A BLOCK (:NAME Y)), Code: TO
Tinker Blocks
'Tinker Functions
(DEFUN STACK (FROM TO)
(MOVE-BLOCK FROM TO))
Z
X
Y
A
B
Table
Table

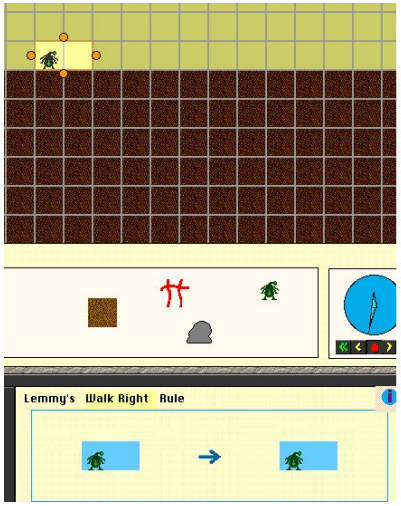

Tinker: (STACK (A BLOCK (NAME A)) (A
Predicate TRUE for: Result: "A on B", Code:
Result: #,(A BLOCK (:NAME A)), Code: FROM
Result: #,(A BLOCK (:NAME B)), Code: TO
Result: T, Code: (CLEARTOP? FROM)
Tinker Blocks World
A
B
Table
Tinker: (STACK (A BLOCK (NAME X)) (A
Predicate FALSE for: Result: "X on Y", Code
Result: #,(A BLOCK (:NAME X)), Code: FROM
Result: #,(A BLOCK (:NAME Y)), Code: TO
Result: NIL, Code: (CLEARTOP? FROM)
Tinker Blocks World
Z
X
Y
Table

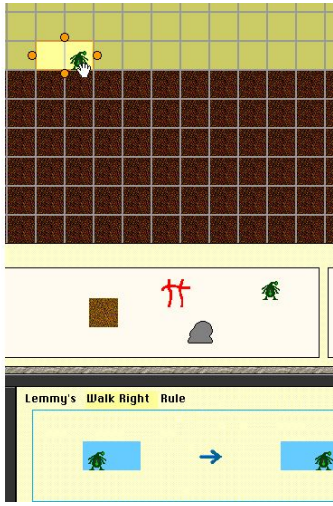

Tinker: (HISTORY)
Defining (HISTORY):
Result: "A on B", Code: (STACK BLOCK-A BLOCK-B)
Result: "X on Y", Code: (STACK BLOCK-X BLOCK-Y)
Result: "D on E", Code: (STACK BLOCK-D BLOCK-E)
'Tinker Functions
(DEFUN
STACK
(FROM TO)
(IF
(CLEARTOP? FROM)
(MOVE-BLOCK FROM TO)
(PROG-N
(STACK (ABOVE FROM) THE-TABLE)
(STACK FROM TO)))
CL-USER| Eval... done.
Tinker Blocks World
H
G
F
D
E
Table
Tinker Bloc
D
E
H
G
F
Table

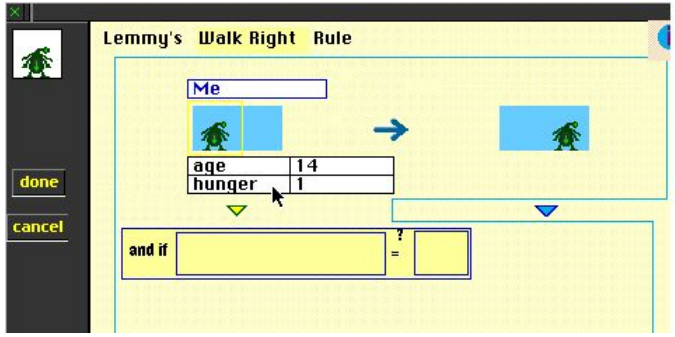
KidSim

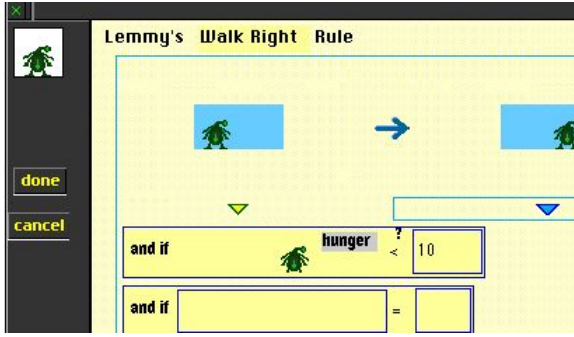
1995


name Lemmy
hunger 4
age 14
Properties
Rules
Appearances
Jump
Walk Right
new rule
GO
Follow
Save
Quit
new
Gremmy's Jump Rule
addon
test
clear


Lemmy's Walk Right Rule


Lemmy's Walk Right Rule


Lemmy's Walk Right Rule
Me
age 14
hunger 1
done
cancel
and if


Lemmy's Walk Right Rule
done
cancel
and if
hunger
10
and if

ToonTalk 1996

Figure 5 -- A bird delivering a box with a nest to its nest

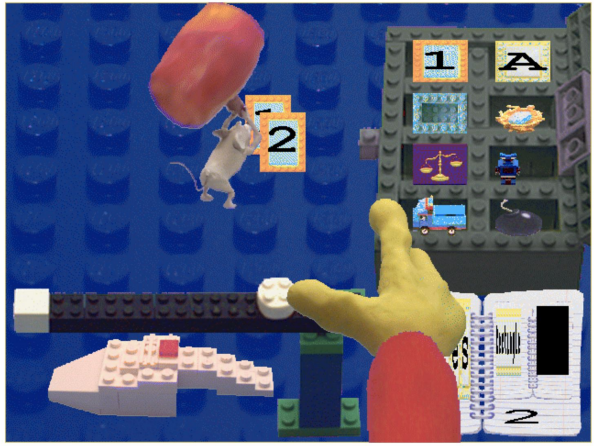


Figure 6 -- A mouse performing 1+2

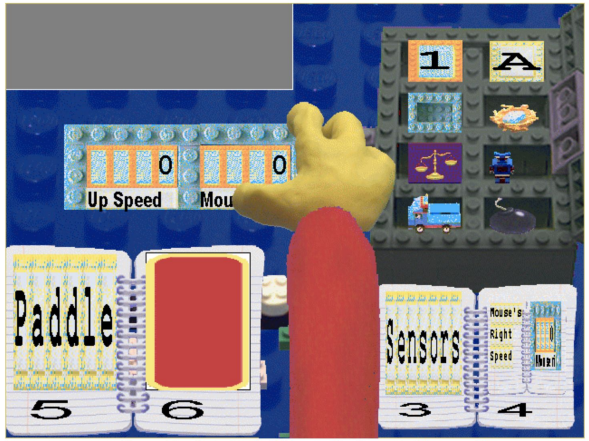


Figure 11 -- Just after building a cubby for controlling a paddle

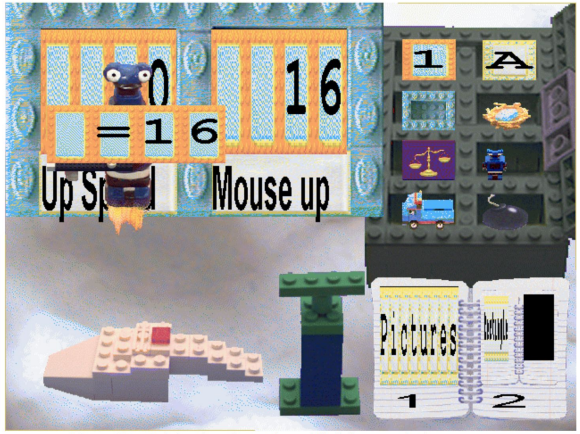


Figure 12 -- Training the robot so the paddle follows vertical mouse movements

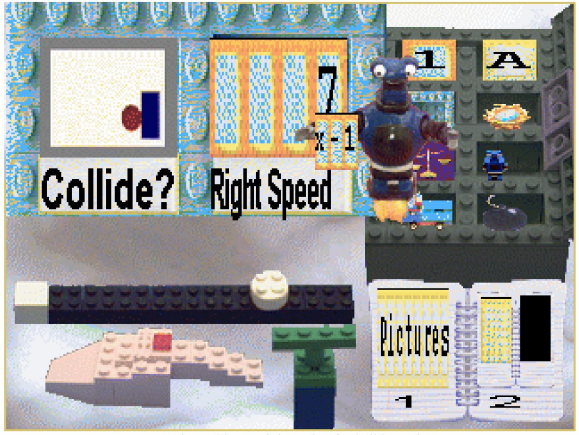


Figure 13 -- Training robot for ball bouncing

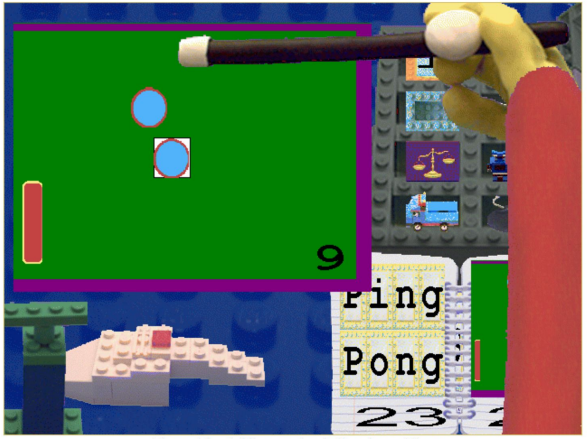
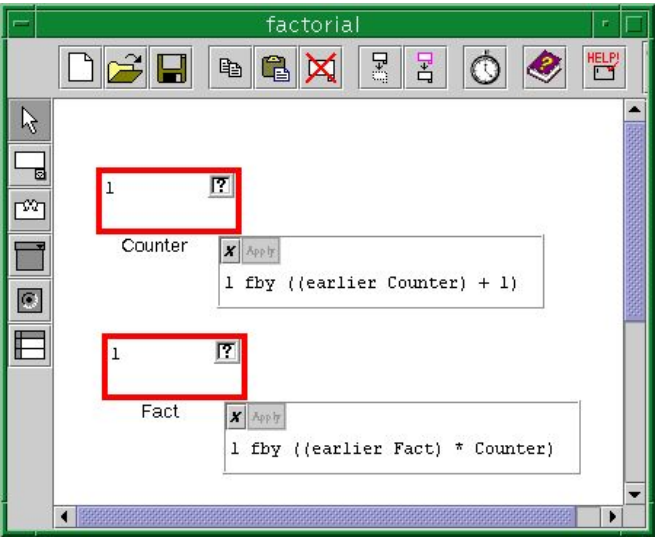
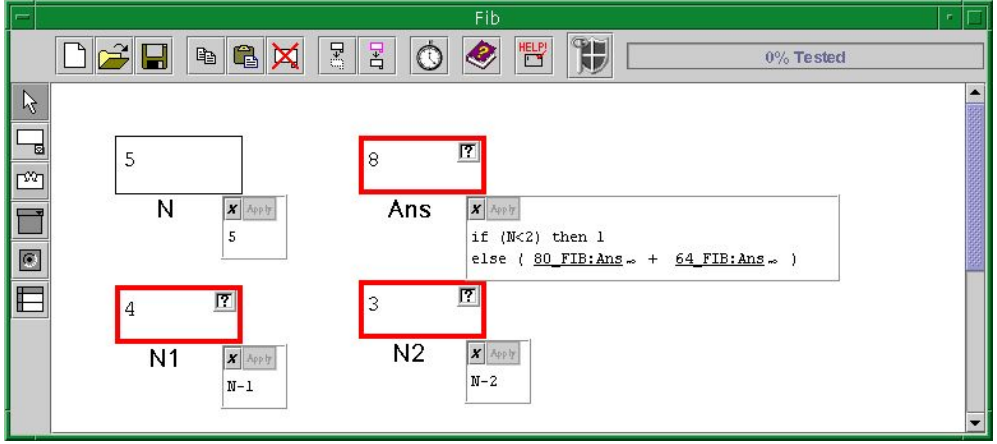


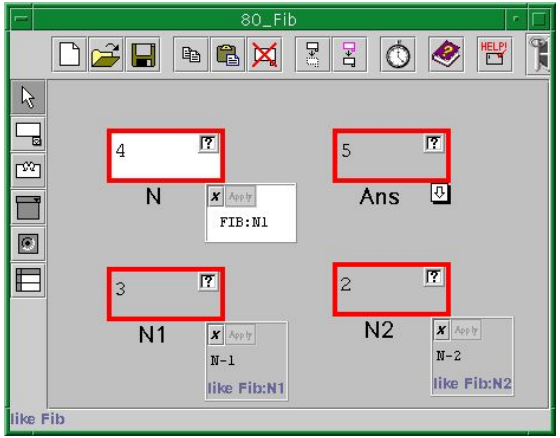
Figure 14 -- Adding another ball to the paddle game

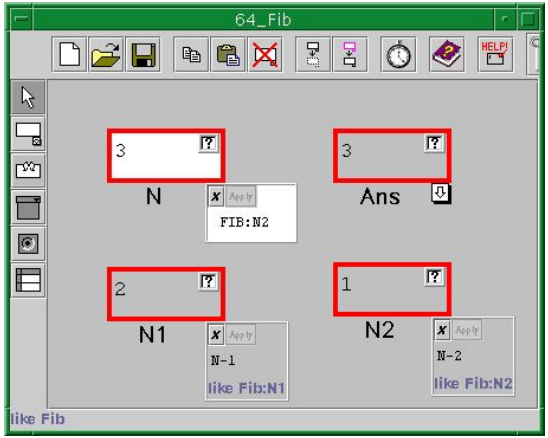
Forms/3

2001


factorial
1
Counter
1 fby ((earlier Counter) + 1)
1
Fact
1 fby ((earlier Fact) * Counter)


Fib
0% Tested
5
N
5
8
Ans
if (N<2) then 1
else (80_FIB:Ans.. + 64_FIB:Ans..)
4
N1
N-1
3
N2
N-2


80_Fib
4
N
FIB:N1
5
Ans
3
N1
N-1
like Fib:N1
2
N2
N-2
like Fib:N2
like Fib


64_Fib
3
N
FIB:N2
3
Ans
2
N1
N-1
like Fib:N1
1
N2
N-2
like Fib:N2
like Fib

FAR 2001

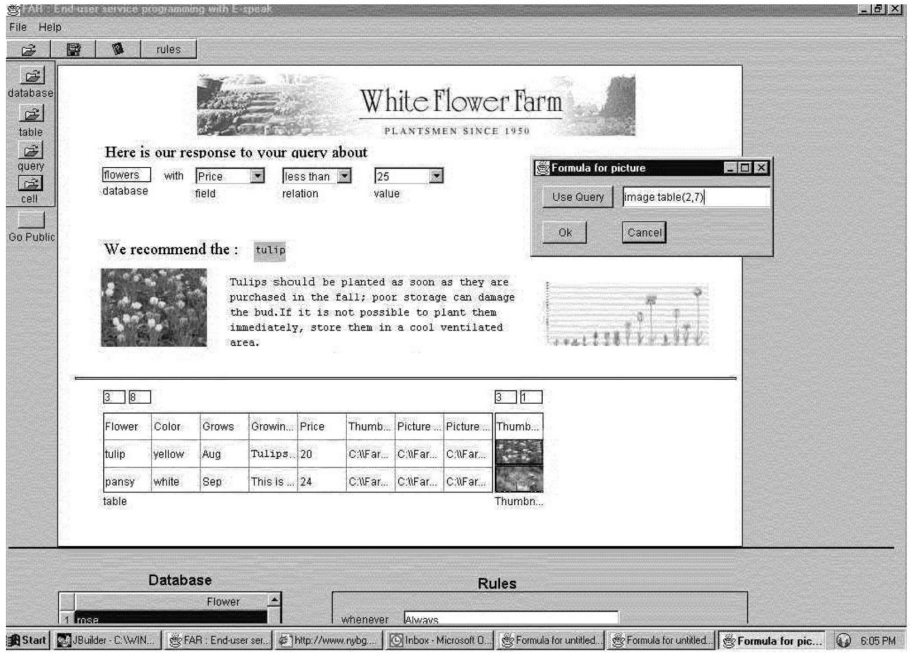


Figure 1: Snapshot of the flower advisor e-service in FAR as it is being created by the gardener. Everything shown in the web page area is a cell or table of cells, including the blocks of text, images, and so on.

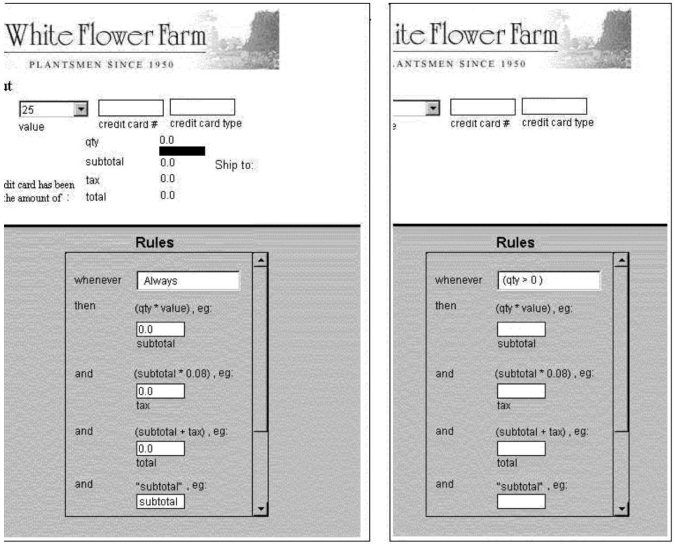
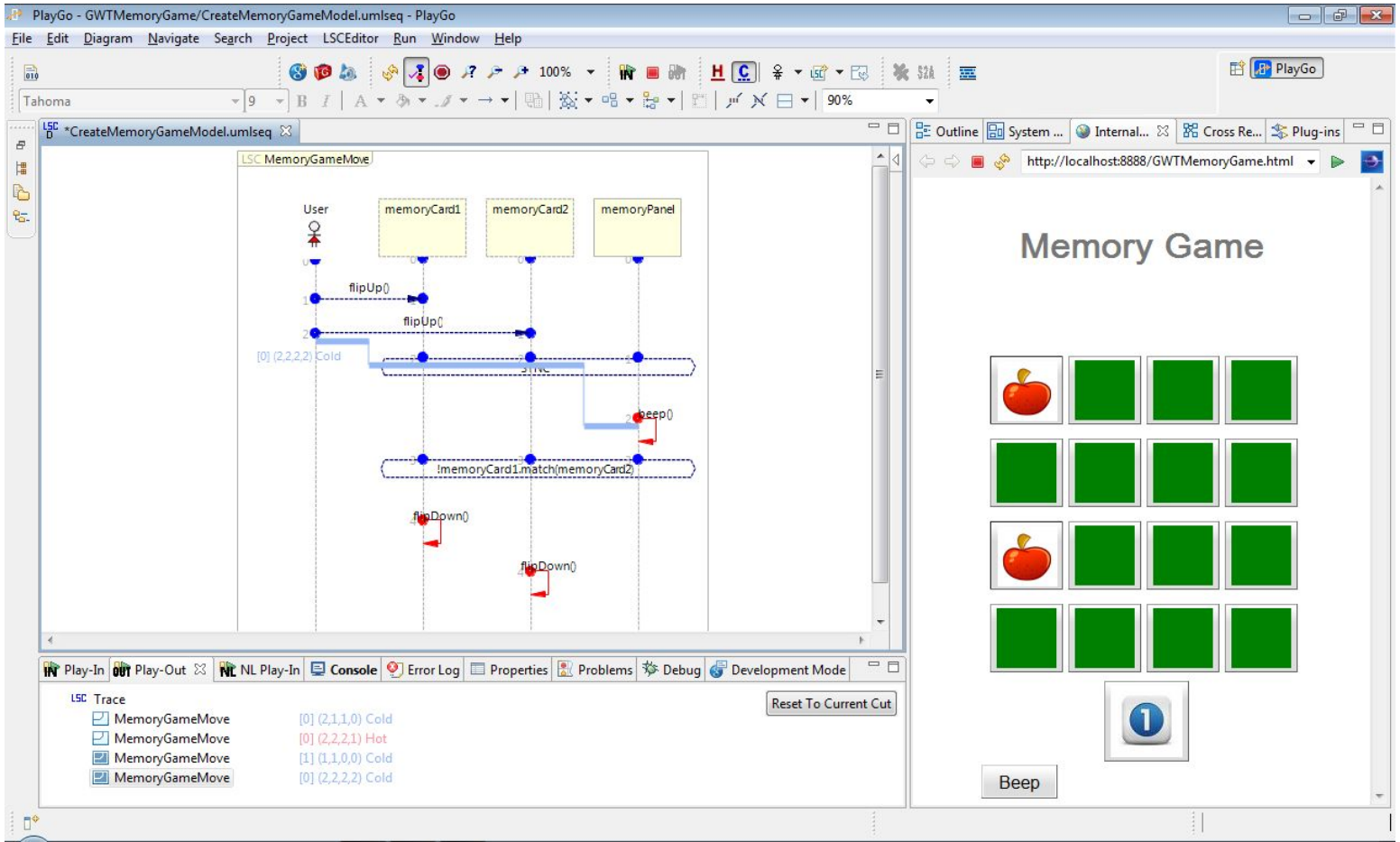
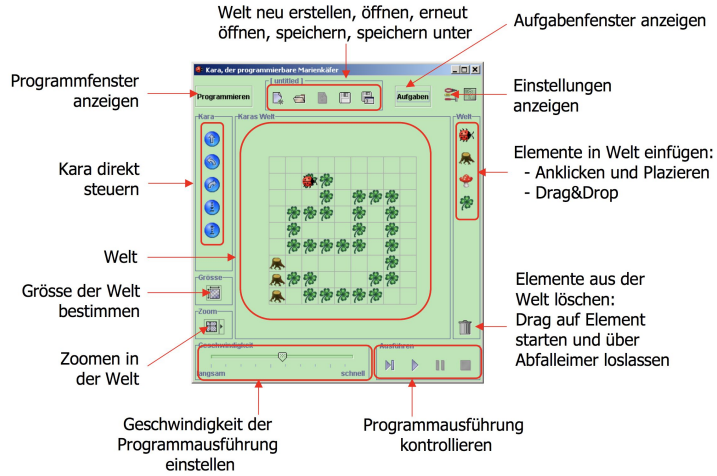


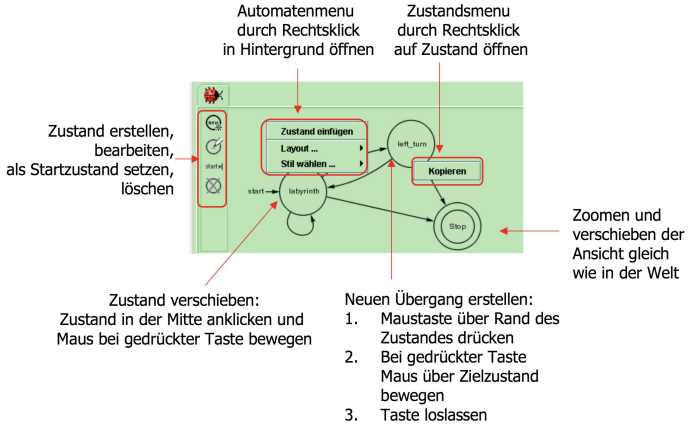
Figure 2: The flower advice example has been expanded, and the separator bar has been dragged upwards to make more room for the rules to be visible. (a) The gardener has added cells allowing a customer to purchase the recommended flowers if desired. The predicate of all these cells is initially "always". (b) The gardener changes the predicate, and the results are immediately reflected in the web page section. Since `qty` is currently 0, the predicate is false, and the cells' values are currently "no-value" (displays as blank).

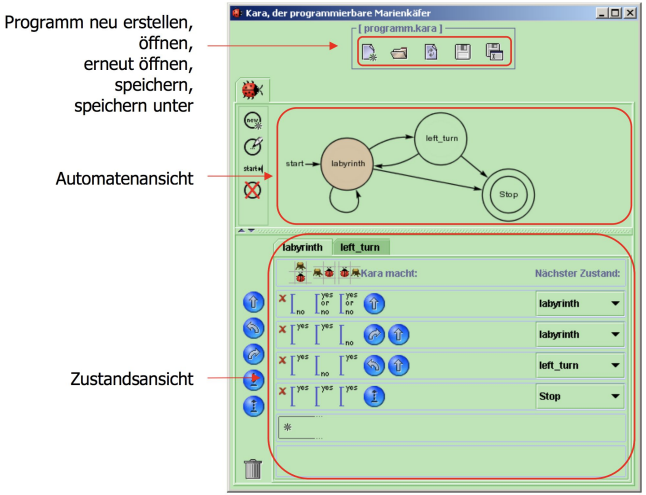
PlayGo 2003

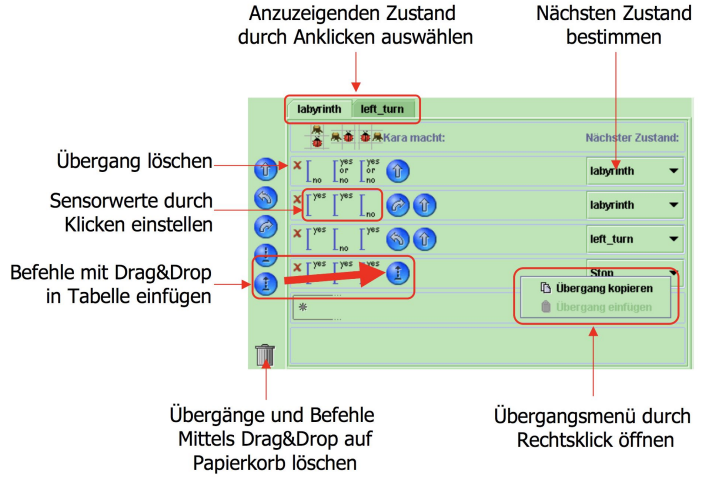

PlayGo - GWTMemoryGame/CreateMemoryGameModel.umlseq - PlayGo
File Edit Diagram Navigate Search Project LSCEditor Run Window Help
*CreateMemoryGameModel.umlseq
LSC MemoryGameMove
User
memoryCard1
memoryCard2
memoryPanel
flipUp()
flipUp()
[0] (2,2,2,2) Cold
beep()
!memoryCard1.match(memoryCard2)
flipDown()
flipDown()
Play-In
Play-Out
NL Play-In
Console
Error Log
Properties
Problems
Debug
Development Mode
LSC Trace
Reset To Current Cut
MemoryGameMove [0] (2,1,1,0) Cold
MemoryGameMove [0] (2,2,2,1) Hot
MemoryGameMove [1] (1,1,0,0) Cold
MemoryGameMove [0] (2,2,2,2) Cold
Outline
System ...
Internal...
Cross Re...
Plug-ins
http://localhost:8888/GWTMemoryGame.html
Memory Game
Beep

Kara 2001


Welt neu erstellen, öffnen, erneut öffnen, speichern, speichern unter
Aufgabenfenster anzeigen
Programmfenster anzeigen
Einstellungen anzeigen
Kara direkt steuern
Elemente in Welt einfügen:
- Anklicken und Plazieren
- Drag&Drop
Welt
Grösse der Welt bestimmen
Elemente aus der Welt löschen:
Drag auf Element starten und über Abfalleimer loslassen
Zoomen in der Welt
Geschwindigkeit der Programmausführung einstellen
Programmausführung kontrollieren


Automatenmenu durch Rechtsklick in Hintergrund öffnen
Zustandsmenu durch Rechtsklick auf Zustand öffnen
Zustand erstellen, bearbeiten, als Startzustand setzen, löschen
Zustand einfügen
Layout ...
Stil wählen ...
Kopieren
Zoomen und verschieben der Ansicht gleich wie in der Welt
Zustand verschieben:
Zustand in der Mitte anklicken und Maus bei gedrückter Taste bewegen
Neuen Übergang erstellen:
1. Maustaste über Rand des Zustandes drücken
2. Bei gedrückter Taste Maus über Zielzustand bewegen
3. Taste loslassen


Programm neu erstellen, öffnen, erneut öffnen, speichern, speichern unter
Automatenansicht
Zustandsansicht
Kara, der programmierbare Marienkäfer
programm.kara
labyrinth
left_turn
Stop
Kara macht:
Nächster Zustand:

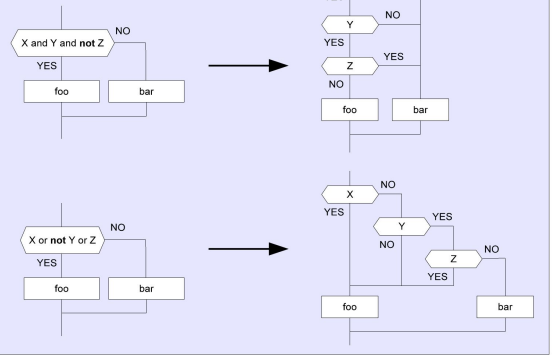

Anzuzeigenden Zustand durch Anklicken auswählen
Nächsten Zustand bestimmen
Übergang löschen
Sensorwerte durch Klicken einstellen
Befehle mit Drag&Drop in Tabelle einfügen
labyrinth
left_turn
Kara macht:
Nächster Zustand:
Übergang kopieren
Übergang einfügen
Übergänge und Befehle Mittels Drag&Drop auf Papierkorb löschen
Übergangsmenü durch Rechtsklick öffnen

Drakon

2011

Text form is shorter but it makes the reader do more mental work.
The reader has to imagine all possible outcomes of a logical expression each time he reads the diagram.

Visual form does not hide anything from the reader.
All paths of the algorithm are clearly visible.

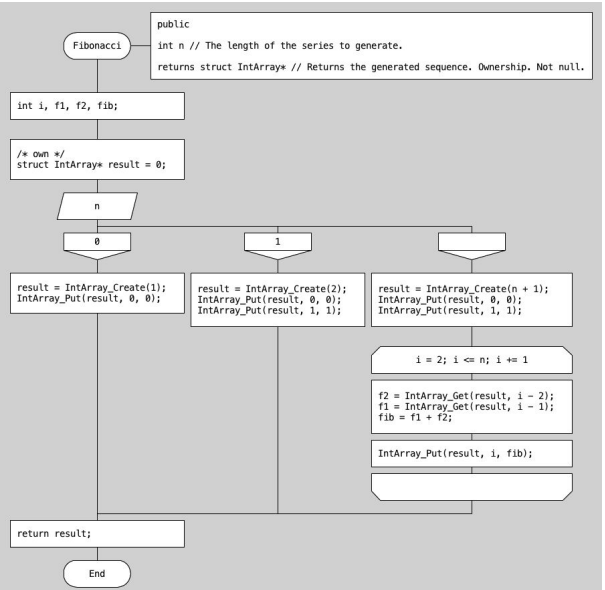


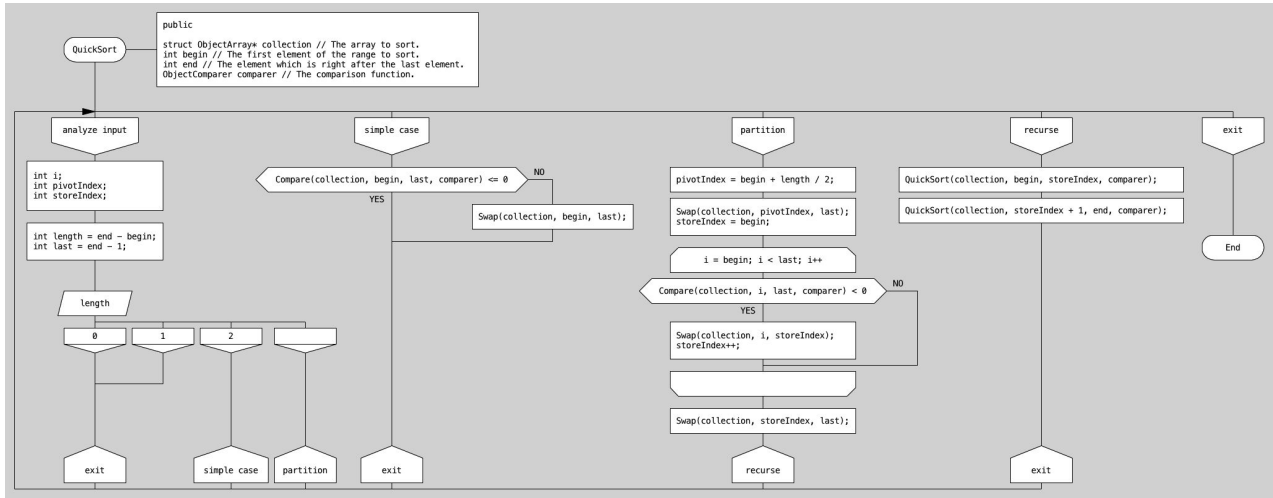
	Icon	Name of Icon
1		Title
2		End
3		Action
4		Question
5		Choice
6		Case
7		Headline
8		Address
9		Insertion
10		Shelf
11		Formal parameters
12		Begin of FOR loop
13		End of FOR loop

	Icon	Name of Icon
14		Output
15		Input
16		Pause
17		Period
18		Start timer
19		Synchronizer
20		Realtime parallel process
21		Comment
22		Right comment
23		Left comment
24		Loop arrow
25		Silhouette arrow
26		Connector
27		Concurrent process

	Macroicon	Name of Macroicon
1		Title with parameters
2		Fork
3	N = 2 / N > 2	Switch (number of cases N >= 2)
4		SIMPLE loop
5		SWITCH loop
6		FOR loop
7		WAIT loop
8		Action by timer
9		Shelf by timer
10		Fork by timer

	Macroicon	Name of Macroicon
11		Switch by timer
12		SIMPLE loop by timer
13		SWITCH loop by timer
14		FOR loop by timer
15		WAIT loop by timer
16		Insertion by timer
17		Output by timer
18		Input by timer
19		Start timer by timer
20		Parallel process by timer
21		TREE loop





SpecTRM-ML 1999

State

Supervisory-Interface

↳Pilot-Displays

↳ Resolution-Advisory

↳**Vertical-Control**

DEFINITION

= Blank

INITIALLY

= Other

Some RA-Strength$_{s\text{-}277}$ **in state** Increase-2500fpm	F	F	F
Some Reversal$_{s\text{-}282}$ **in state** Reversed	F	F	F
Composite-RA$_{s\text{-}266}$ **in state** Climb	F	•	•
Composite-RA$_{s\text{-}266}$ **in state** Descend	F	•	•
Corrective-Climb$_{s\text{-}263}$ **in state** Yes	•	F	•
Corrective-Descend$_{s\text{-}264}$ **in state** Yes	•	•	F
Crossing-Geometry$_{m\text{-}388}$	F	F	F

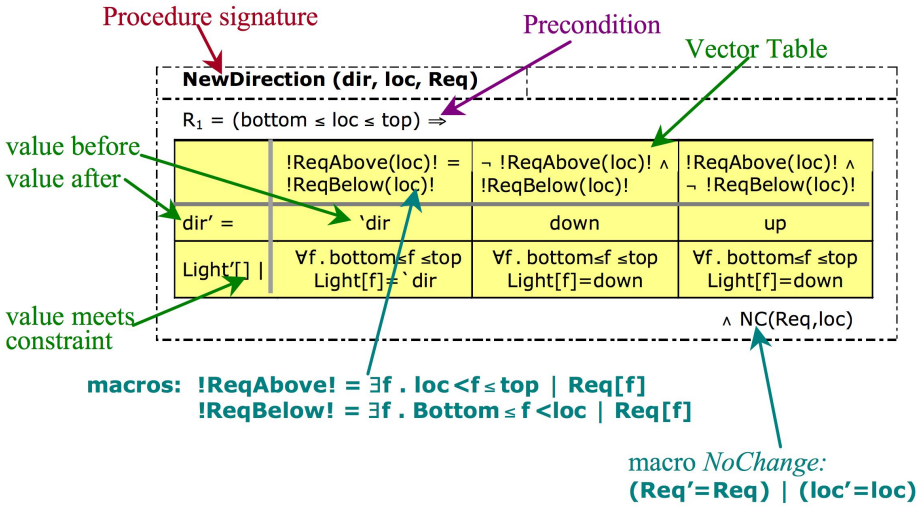
= Increase

Some RA-Strength$_{s\text{-}277}$ **in state** Increase-2500fpm	T
Climb-Inhibit$_{s\text{-}243}$ **in mode** Inhibited	T
Descend-Inhibit$_{s\text{-}245}$ **in mode** Inhibited	T

= Crossing

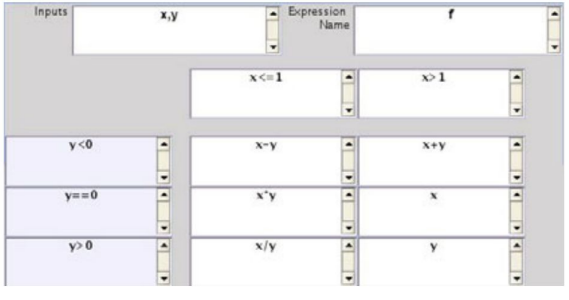
Some Reversal$_{s\text{-}282}$ **in state** Reversed	F	F	F	F
Composite-RA$_{s\text{-}266}$ **in state** Climb	T	T	•	•
Composite-RA$_{s\text{-}266}$ **in state** Descend	•	•	T	T
Some RA-Strength$_{s\text{-}277}$ **in state** Increase-2500fpm	F	F	F	F
Corrective-Climb$_{s\text{-}263}$ **in state** Yes	F	•	F	•
Corrective-Descend$_{s\text{-}264}$ **in state** Yes	•	F	•	F
Crossing-Geometry$_{m\text{-}388}$	F	F	F	F

Parnas Tables 2001



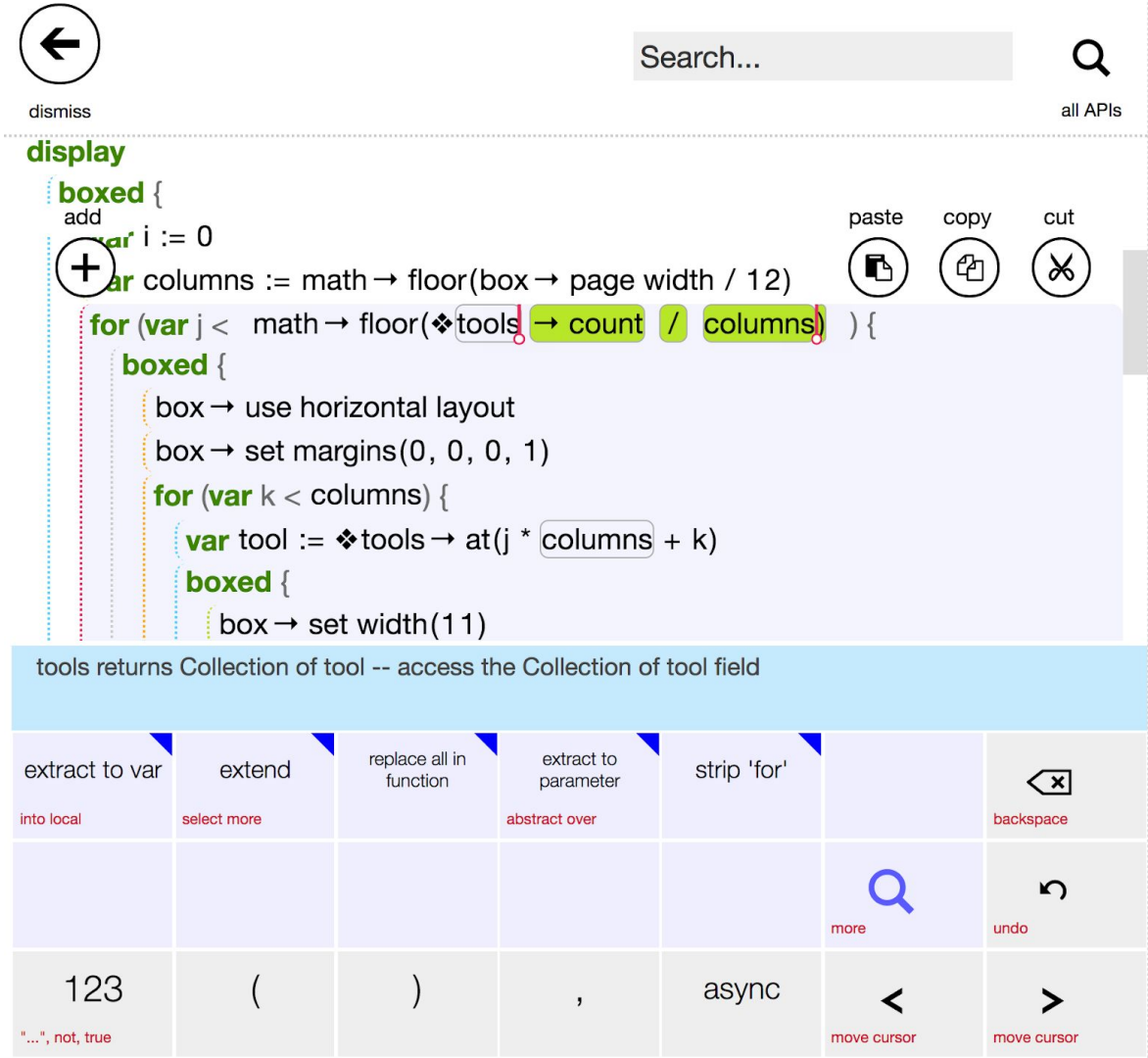
Tabular Expressions 2011

$$f(x,y) \stackrel{df}{=} \begin{cases} x+y & \text{if } x>1 \wedge y<0 \\ x-y & \text{if } x\le 1 \wedge y<0 \\ x & \text{if } x>1 \wedge y=0 \\ xy & \text{if } x\le 1 \wedge y=0 \\ y & \text{if } x>1 \wedge y>0 \\ x/y & \text{if } x\le 1 \wedge y>0 \end{cases}$$



TouchDevelop

2011



YinYang 2011

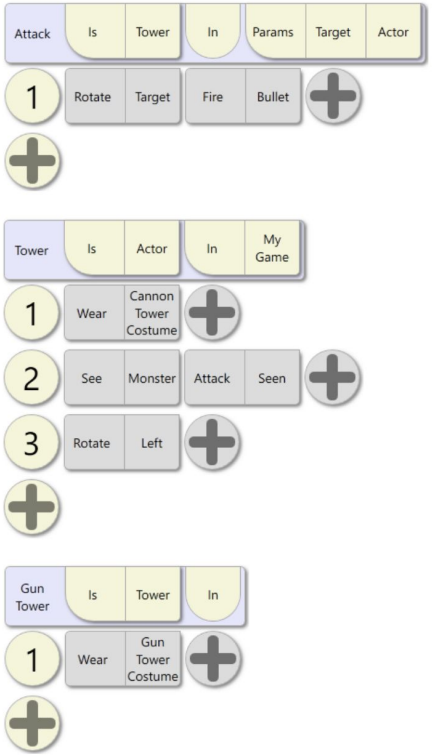


Figure 4. Definitions of the `Attack`, `Tower`, and `Gun Tower` tiles.

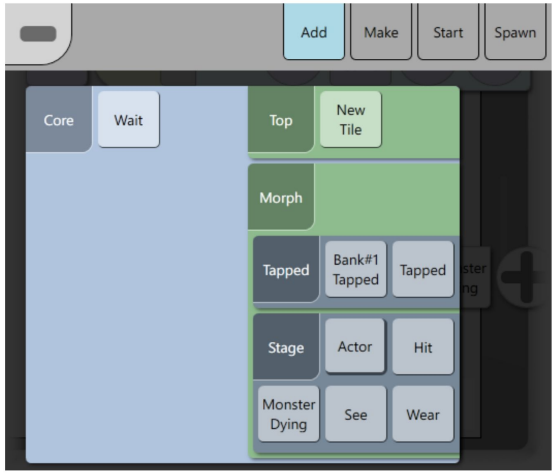
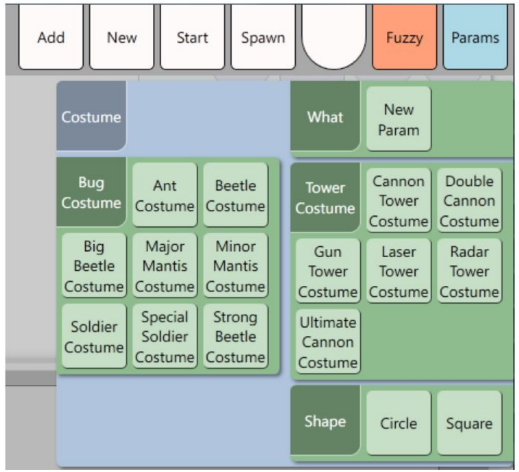


Figure 7. The root context menu (unscaled) for adding a new act in a `Monster` tile that extends the `Actor` tile. When the menu is active, programming content is grayed out. The menu has two parts: a top-positioned bar of menu modes, actions, and menu-wide options, and a menu of options that appears close to where the user tapped. The top level of `Core` panel of the menu contains the second-level `Top` and `Morph` panels; the second-level `Morph` panel contains the third-level `Tapped` and `Stage` panels.

the costume. However, since the `Monster` tile has not yet extended the `Bug` tile, bug costumes are not visible in this context menu. By selecting the `Fuzzy` option, additional costumes become visible in the menu:



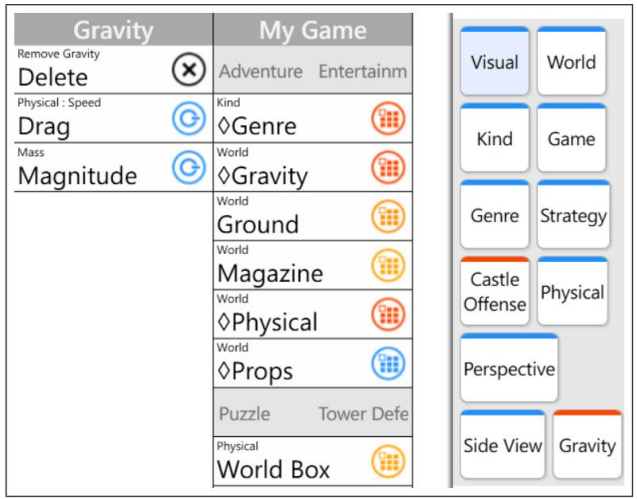


Figure 6. The YinYang editor after `Gravity` has been selected for the `My Game` object; code input menus are on the left; `My Game` labels the object's root menu; `Gravity` labels the `Gravity` sub-menu; and traits extended by the `My Game` object are on the right.

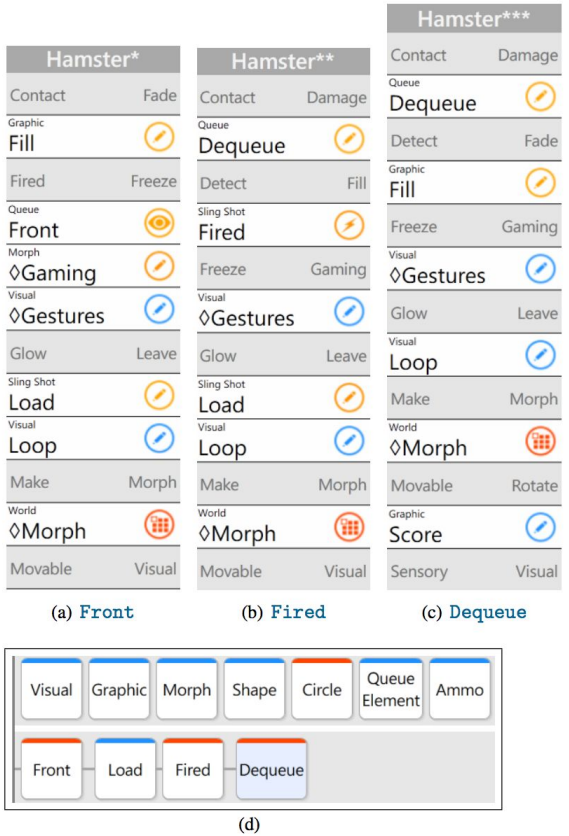
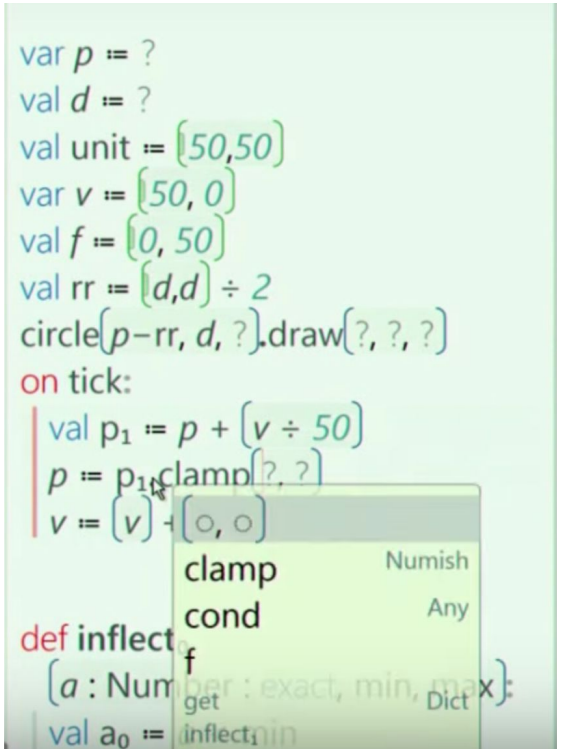
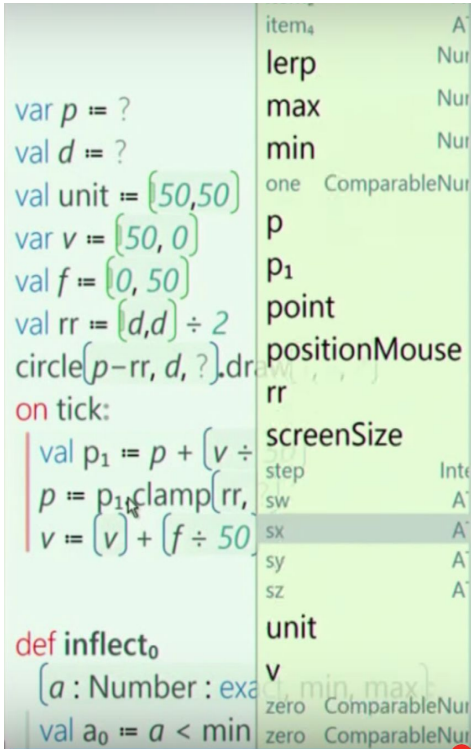


Figure 7. Input menus for specifying behaviors of a `Hamster` object where (a) the object's code is empty; (b) after the `Front` state is selected in (a); (c) after the `Fired` event is selected in (b); and (d) the object's code after `Dequeue` is selected from (c).

Sean McDirmid APX





Sean-boxing
2017

```
var cx ≔ document.querySelector canvas .getContext 2d
var total ≔ results.reduce sum choice ⇒
    return sum + choice.count
  | 0
this.helloNurse this | this
1 + 2 + 3 + 4 + 5 + 1 + 1 + 2

  a multi-line comment
  that keeps going until here

var currentAngle ≔ -0.5 × Math.π;
  before comment 1 + 1 after
var centerX ≔ 300, centerY ≔ 150
  Add code to draw the slice labels in this loop.
results.forEach result ⇒
    var sliceAngle ≔ result.count ÷ total × 2 × Math.π
    cx.beginPath()
      do comments show up here?
    cx.arc centerX | centerY | 100 | currentAngle
      currentAngle + sliceAngle
    currentAngle Δ sliceAngle
    cx.lineTo centerX | centerY
    cx.fillStyle ≔ result.color
    cx.fill()
```

Greenfoot 2006

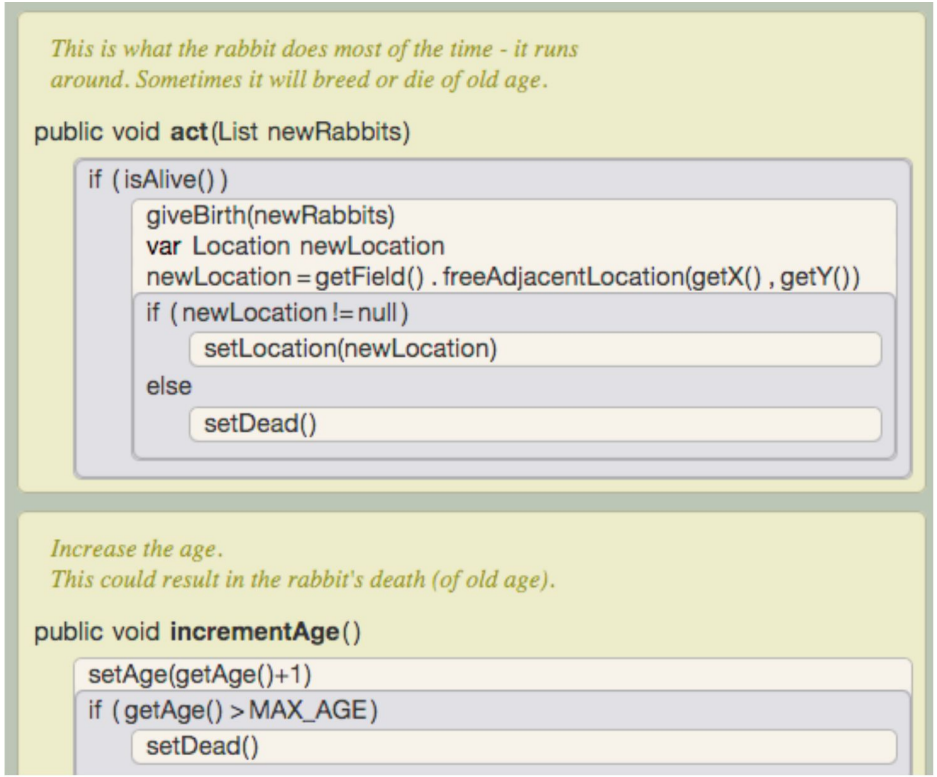


Figure 4. Code representation in a frame-based editor

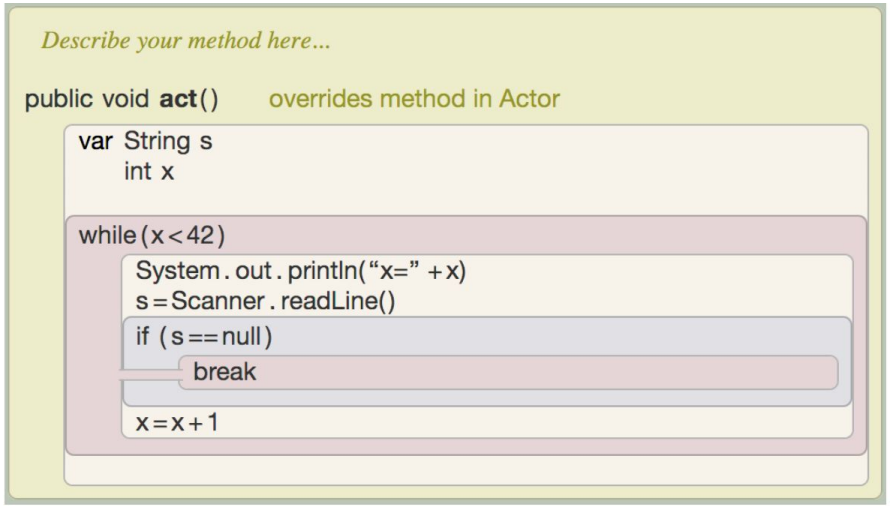


Figure 6. Representation of a break statement

Figure 7. An if-statement with empty slots

Figure 8. Optional text slots: invisible without focus (left) and visible when holding keyboard focus (right)

Figure 9. Cursors: a frame cursor (left) and a text cursor (right)

Figure 10. Preview of delete operations

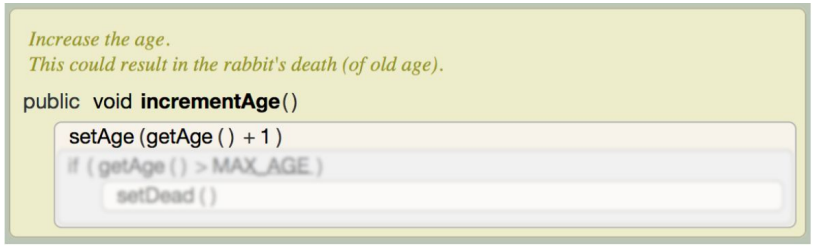


Figure 11. A disabled frame

Figure 13. A method frame with empty slots

Figure 14. Text entry in a choice slot

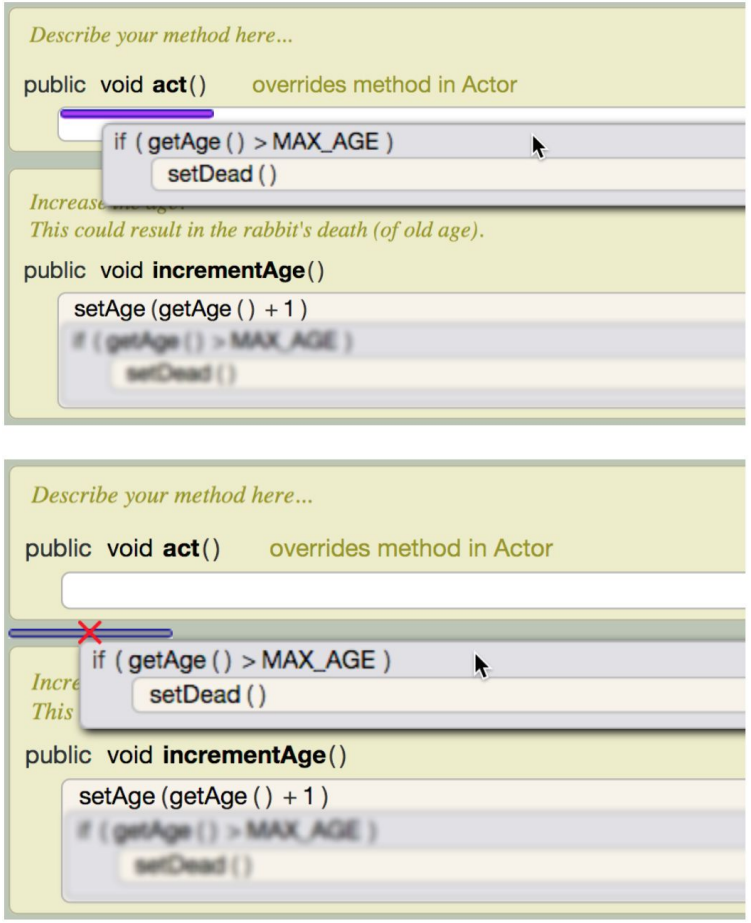


Figure 12. Cursor indicates valid (top) and invalid (bottom) drop targets. The drag source (at bottom of each) is blurred during drag.

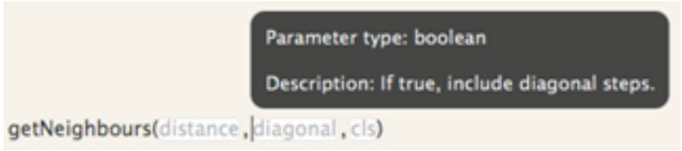


Figure 15. Prompt text and tooltip for parameters

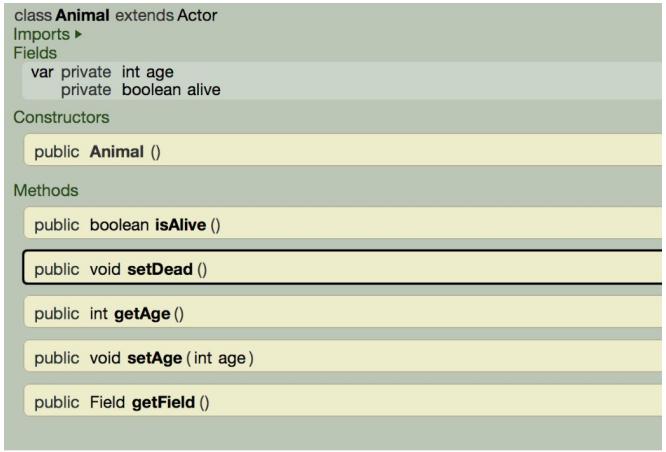


Figure 17. The Bird's Eye View

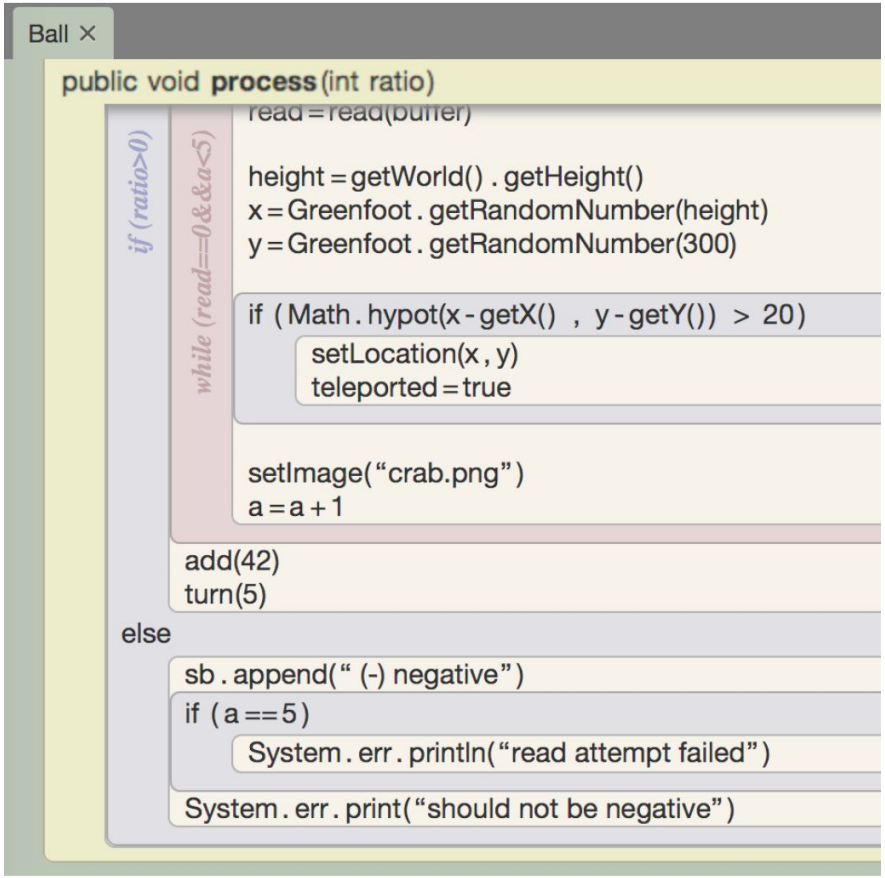


Figure 16. Method signature pinned to top of screen

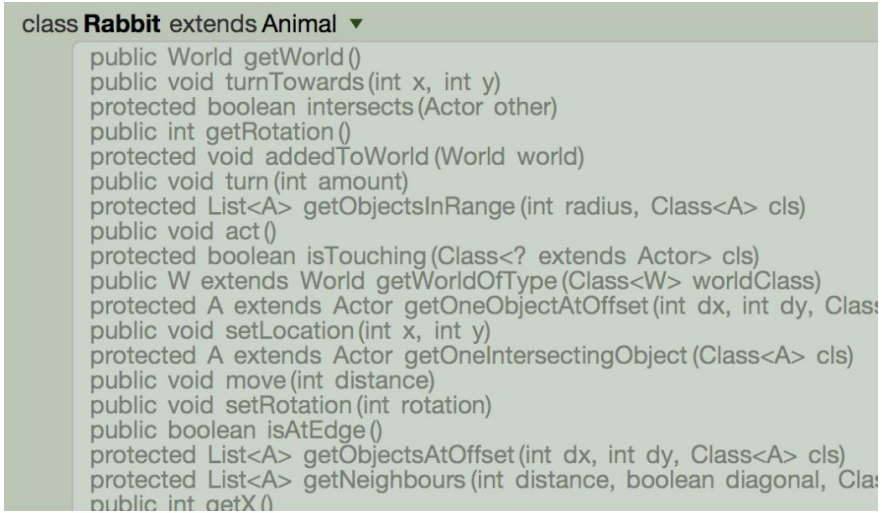


Figure 18. List of inherited methods

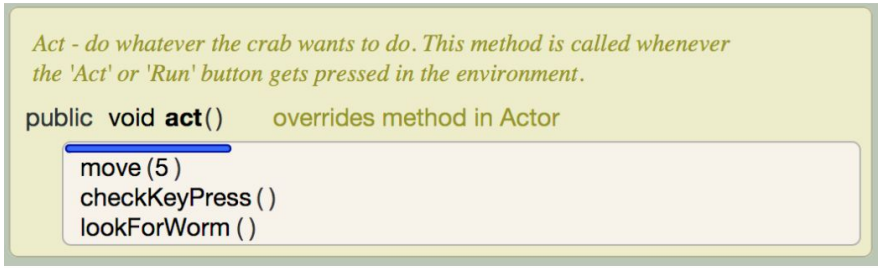


Figure 19. The override annotation

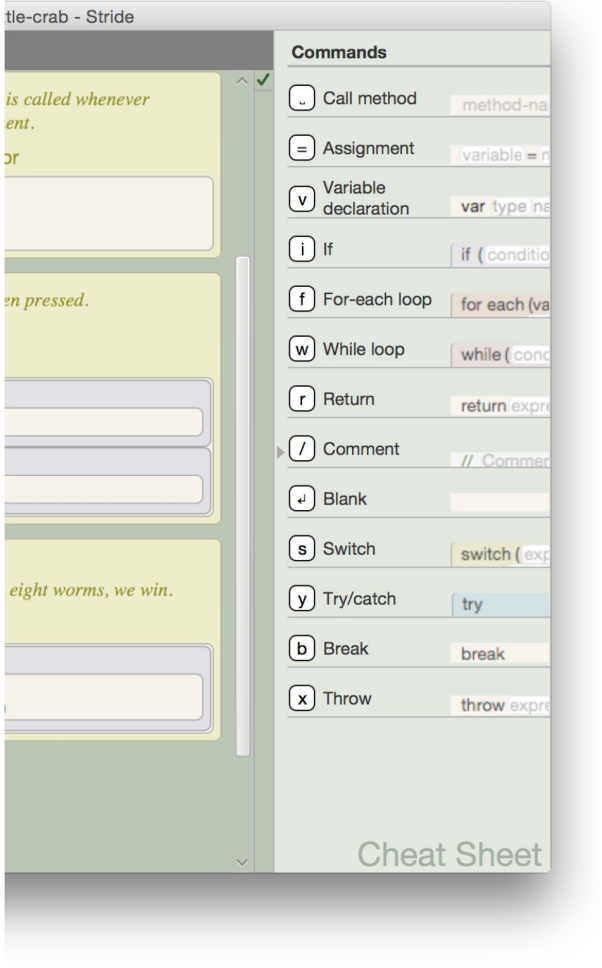


Figure 20. The "Cheat Sheet", which appears on the right of the editor

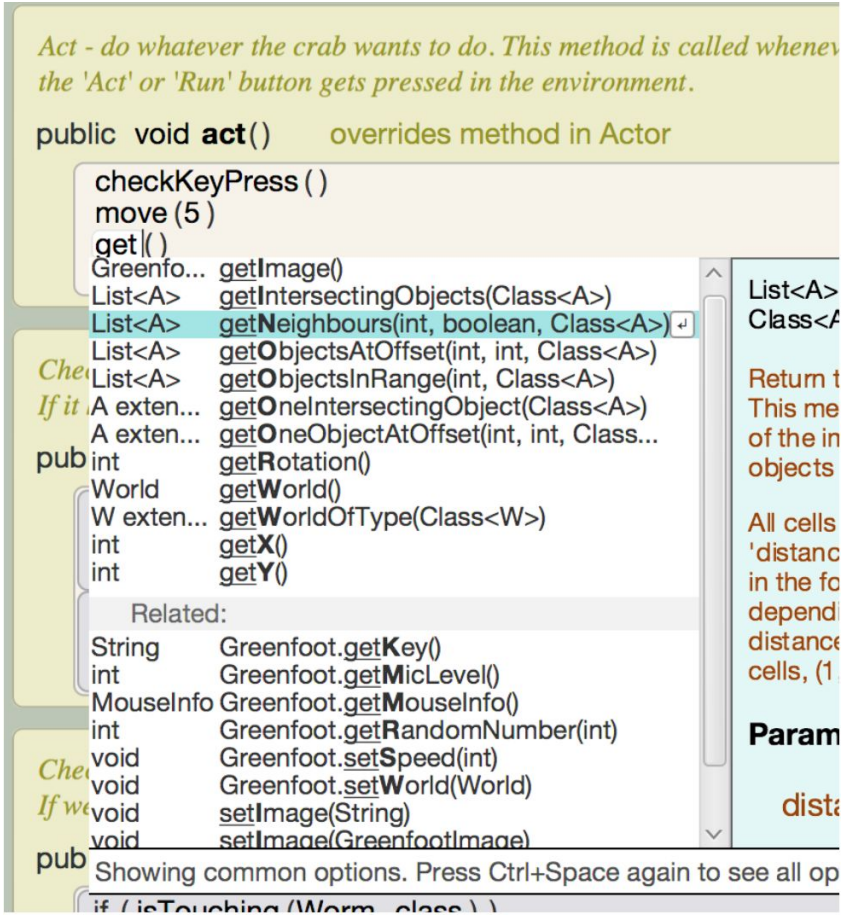


Figure 21. Code completion in Stride

Dylan 1992

File Edit Text Project Browse Debug Windows Help

Project: tiles (inactive)

Contents of
tiles (inactive)

- Dyla
- dylan-framework (inactive)
- Dylan-user
  - documentati
    - Copyright (C) 1994, Apple Computer, Inc. All rights reserved.
    - MacDylan Tiles Program
    - Version 1
    - Version 2
    - Version 3
    - Version 4
    - To Do
  - library & module
    - Copyright (C) 1994, Apple Computer, Inc. All rights reserved.
    - define library **tiles**
      use Dylan;
      use mac-toolbox;
      use Dylan-Framework;

      export tiles;
      end library;
    - define module **tiles**
      use Apple-Dylan;
      use dylan-framework,
      // exclude sealed generic with conflicting name
      // This is because the framework's <grid-view> is
      // a bit like our <cell-view>.
      exclude: { point-to-cell };
      use mac-toolbox;

      export
      init-tiles,
      run-tiles;
      end module;
- tiles
  - cell view
    - /* Copyright (C) 1994, Apple Computer, Inc. All rights reserved. */
    - cells
    - define class **<cell>** (<object>)
      // Cells sit in a view, and have (x,y) coordinates
      // and neighbors.
      slot view :: <view>,
      required-init-keyword: view:;

      slot row :: <integer>,
      required-init-keyword: row:;

      slot column :: <integer>,
      required-init-keyword: column:;

Dylan-user

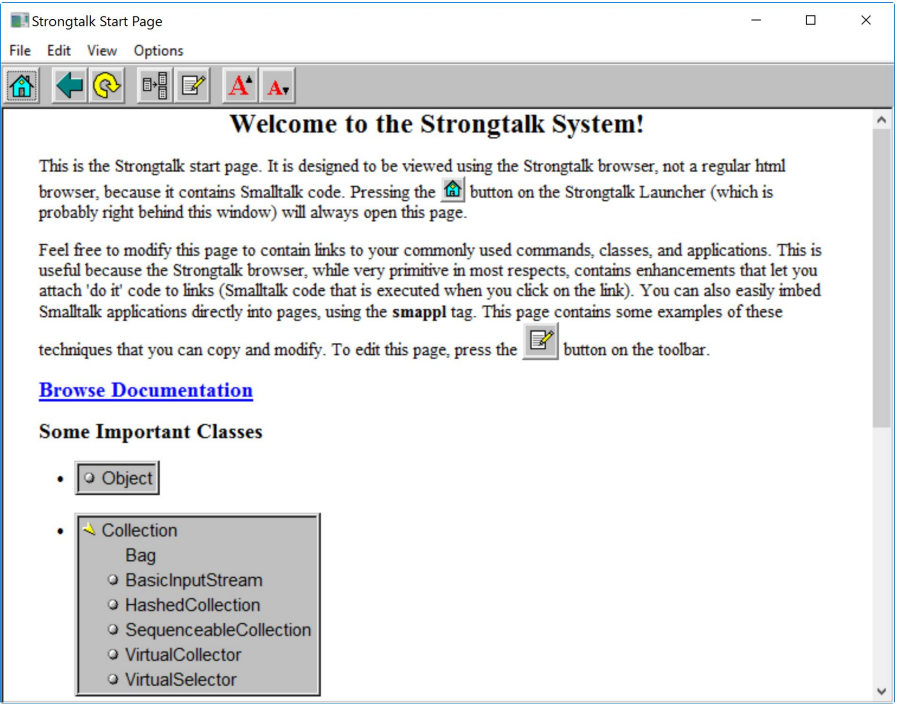
Function Family: + (object1, object2)

Function Family of
+ (object1, object2)

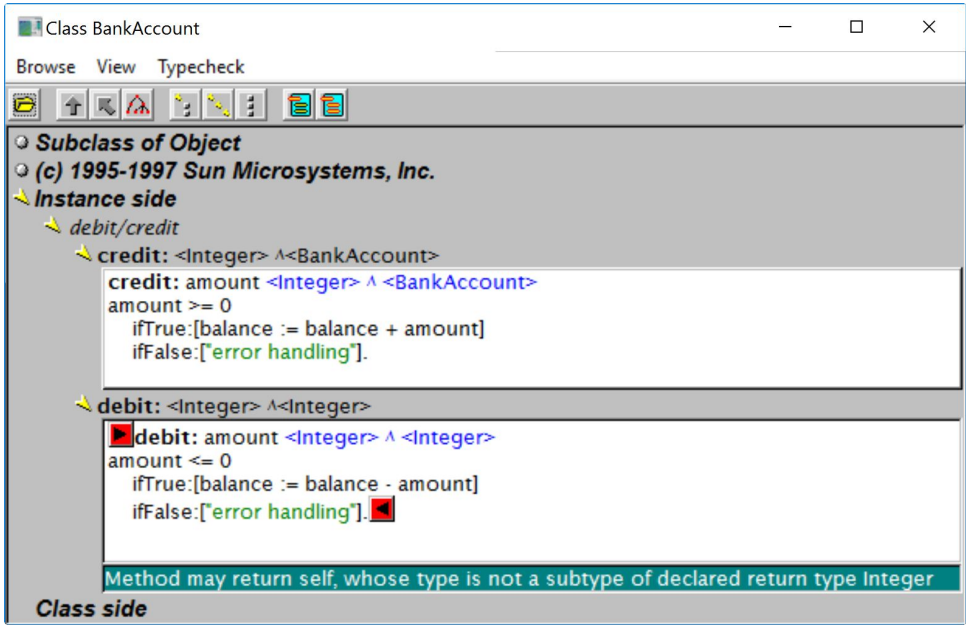
- + (x :: <small-integer>, y :: <small-integer>) => + :: <integer>
  - No source code available
- + (p :: <machine-pointer>, o :: <integer>)
- + (o :: <integer>, p :: <machine-pointer>)
- + (object1, object2)
- + (r1 :: <ratio>, r2 :: <ratio>)
- + (r :: <ratio>, n :: <integer>)
- + (n :: <integer>, r :: <ratio>)
- + (x :: <big-integer>, y :: <small-integer>) => v :: <integer>
- + (x :: <small-integer>, y :: <big-integer>) => v :: <integer>
- + (x :: <big-integer>, y :: <big-integer>) => v :: <integer>
- + (x :: <double-float>, y :: <double-float>) => result :: <double-float>
- + (x :: <double-float>, y :: <rational>) => result :: <double-float>
- + (x :: <rational>, y :: <double-float>) => result :: <double-float>
- + (p1 :: <point>, p2 :: <point>) => sum :: <point>
- + (a :: <number>, b :: <point>) => result :: <point>
  define method \+ (a :: <number>, b :: <point>) => result :: <poi
  point(a + b.h, a + b.v);
  end method;
- + (a :: <point>, b :: <number>) => result :: <point>
  define method \+ (a :: <point>, b :: <number>) => result :: <poi
  point(a.h + b, a.v + b);
  end method;
- + (a :: <rect>, b :: <rect>) => result :: <rect>
- + (a :: <rect>, b :: <number>) => result :: <rect>
- + (a :: <number>, b :: <rect>) => result :: <rect>
- + (a :: <rect>, b :: <point>) => result :: <rect>

Dylan

Strongtalk 1993



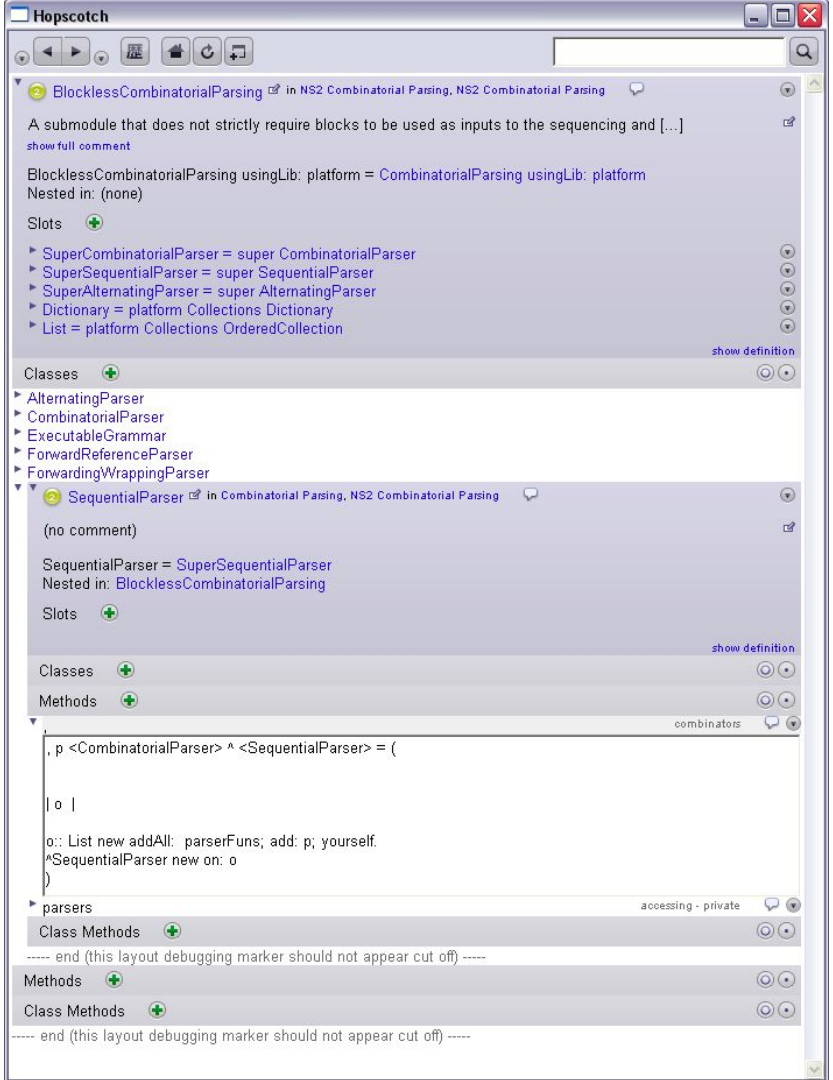
This shows a literate programming style. In particular, we have two Strongtalk class hierarchy browsers embedded in a document. The second, for Collection and its subclasses, is partly expanded. The browsers are of course live and can be interacted with (e.g., expanded, collapsed, inspected etc.) from within the document.

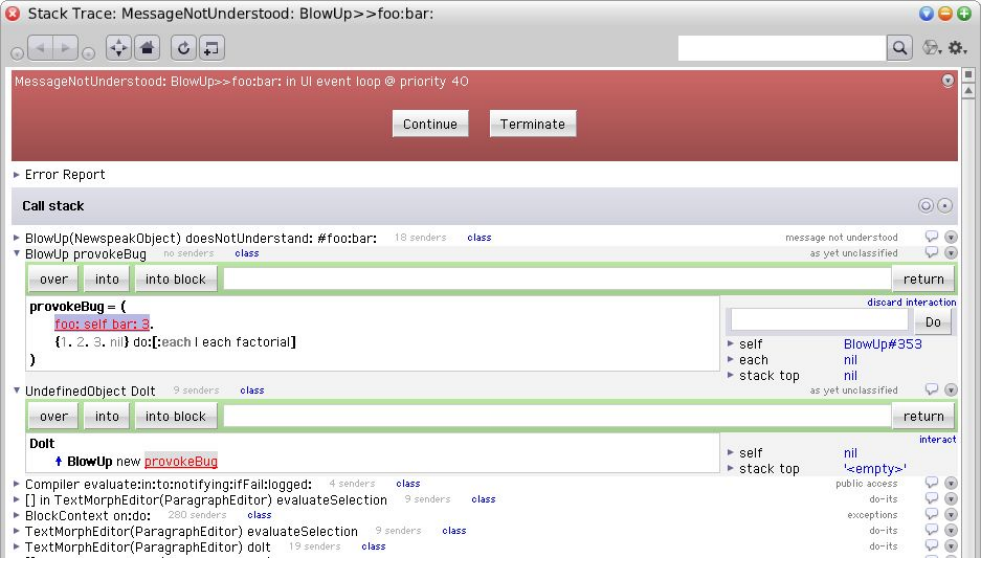


The class browser differs from the classic Smalltalk one. It shows a tree view of the methods, so multiple methods can be viewed simultaneously. There are options to open all methods, which gives you something a bit like looking at a file.
Note the **credit:** method - the edges of the method editor give the visual impression of sticking out rather than being recessed, as they are in the **debit:** method. This is a cue that the text has been modified but not yet saved.
The **debit:** method has been typechecked. The type errors are reported, conventionally (though it was new back then) at the bottom of the method editor. The error is highlighted by two arrow widgets.

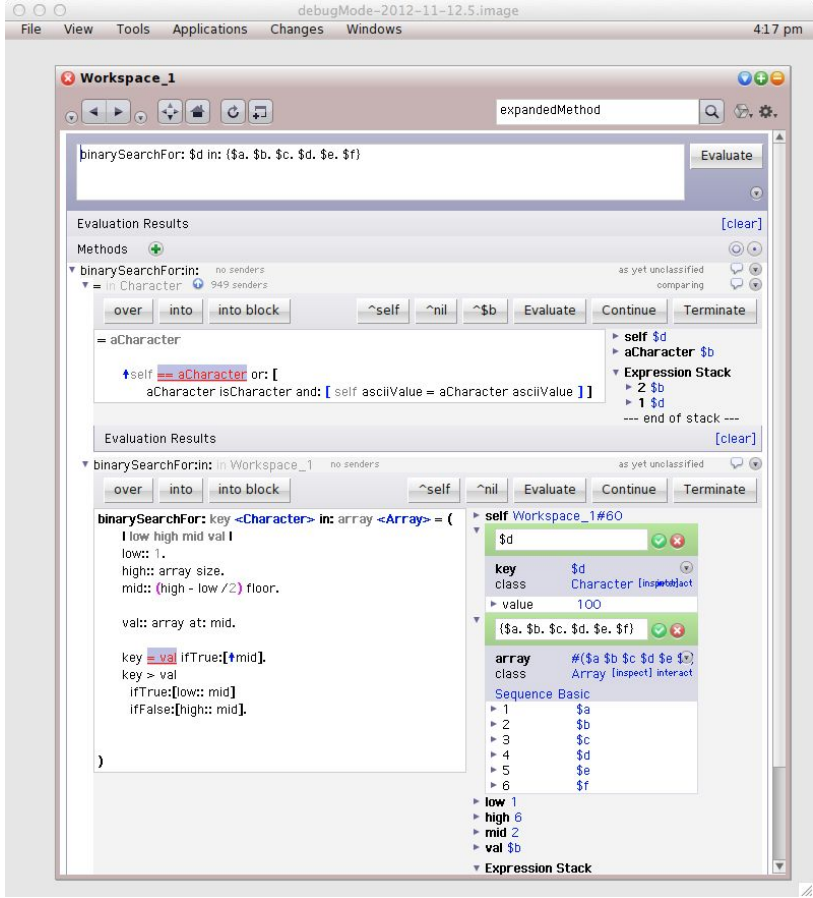
Hopscotch 2008

Hopscotch is a navigation based UI, like a web browser or current mobile platforms. Hence, the window has some general purpose facilities at the top - a search pane, navigation arrows, history button etc. **There are no tabs.** The history page is more scalable. Navigation makes any other open view accessible within two clicks - history button and the desired view.
The class being viewed is shown below. It is a tree view, inheriting something from the Self and Strongtalk. We have a nested class, SequentialParser, expanded, and with it one of the methods is expanded in place. This has the same property the Strongtalk browser had, that you can view multiple methods at once.





Newspeak-debugger. It inherits from the Self and Strongtalk debuggers, the stack view, which allows multiple frames to be open at once and corresponds to the natural, white board visualization of the call stack. I actually have a post with screenshots of those: https://gbracha.blogspot.com/2008/07/debugging-visual-metaphors.html.
An open frame shows the method code on the left, and local and parameter bindings on the right. Each variable can be expanded into an inspector to see its fields, access its class, evaluate expressions within its scope etc.

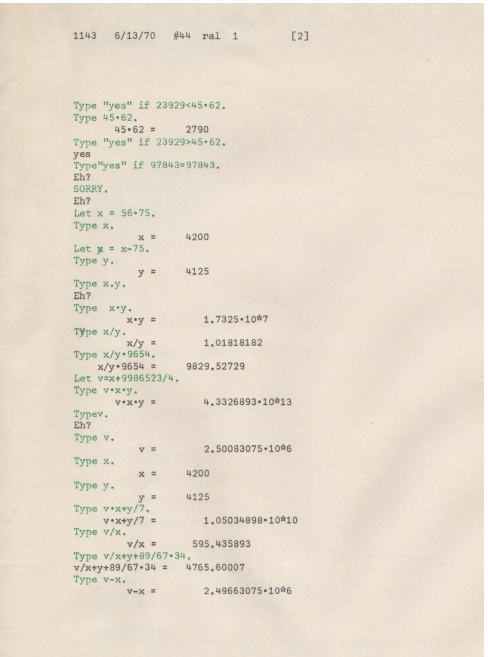


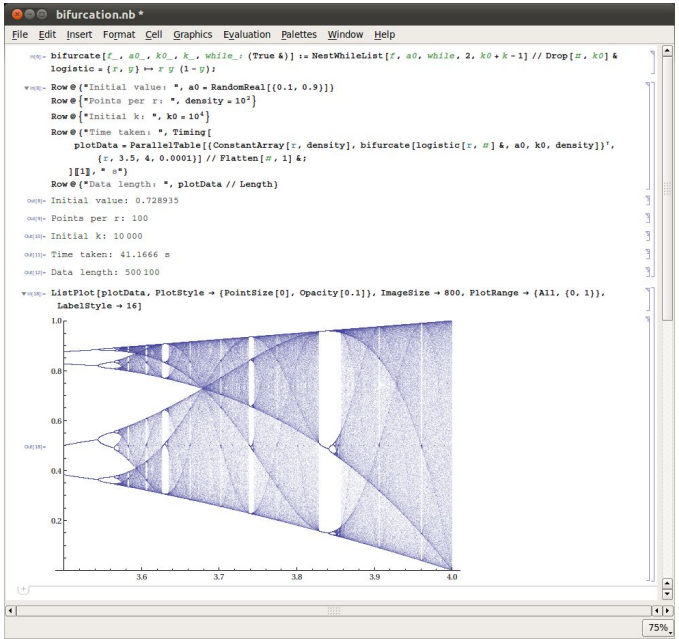
Exemplar workspace. See also https://gbracha.blogspot.com/2013/04/making-methods-live.html.
The workspace has an editable pane at the top, in which multiple snippets can be kept and evaluated, much like the classic Smalltalk workspace (which is notably different than a REPL, precisely because multiple things can be kept around).
Below, the workspace has an area for workspace-specific methods. This is unusual. In fact, the workspace is an instance of a custom-made class, which is where the methods really go.
he method is presented like an activation record in the debugger, so in addition the code, as shown in the conventional class browser, we have live values, we can execute the code etc. In this shot, we are stepping into the call to **=**. Essentially the same debugger UI as above.

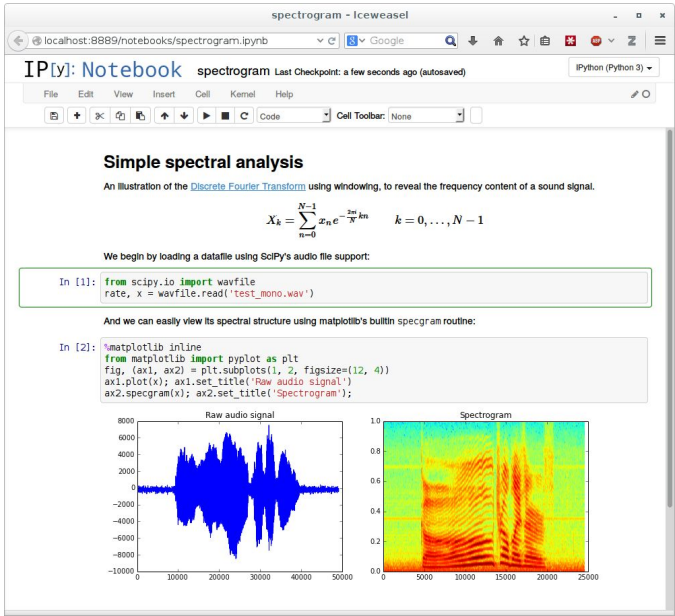
JOSS
1966

Mathematica
1988

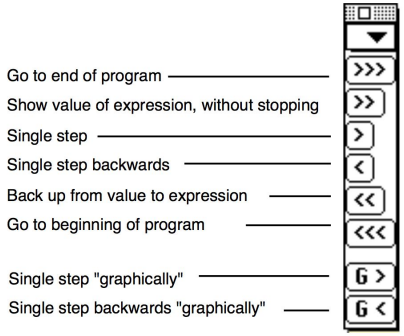
IPython
2001

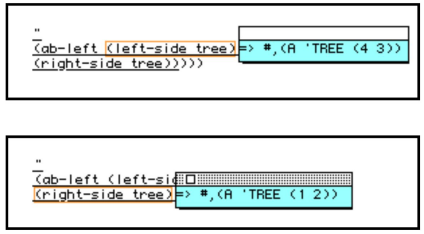




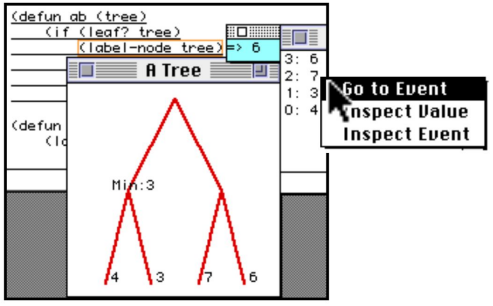


ZStep 95

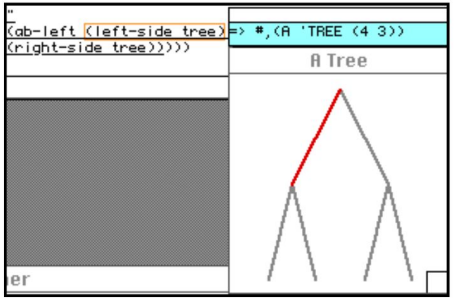


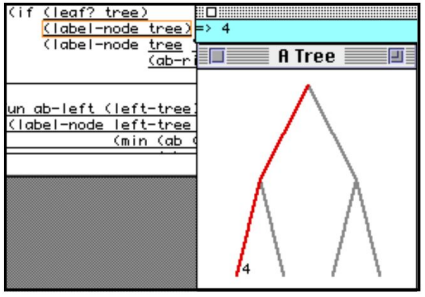


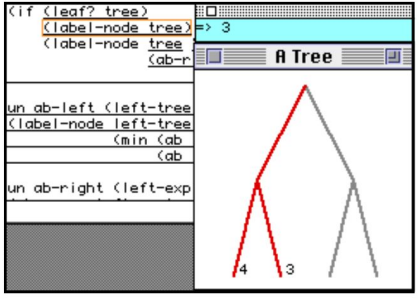
The value window moves through the code



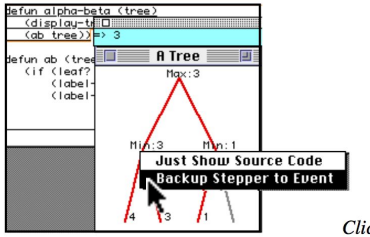
The history of values of an expression



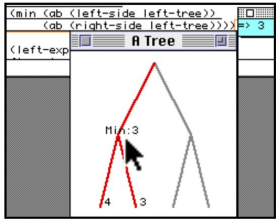


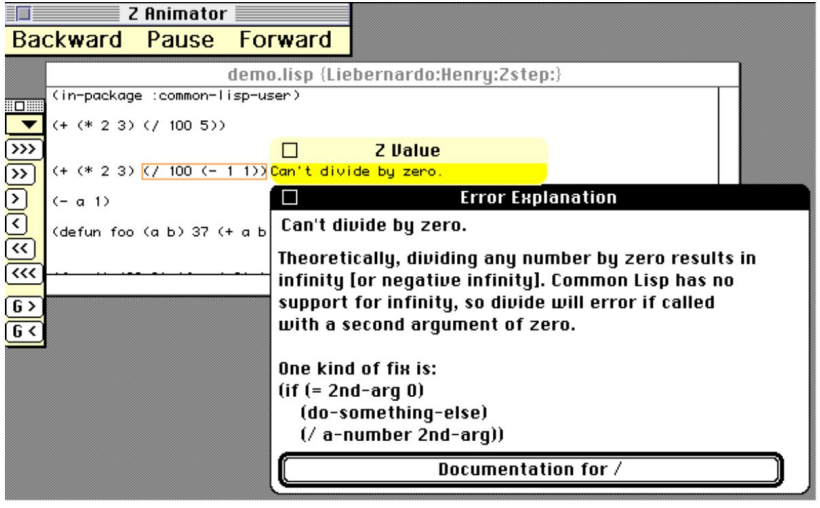


Three successive "graphic steps"



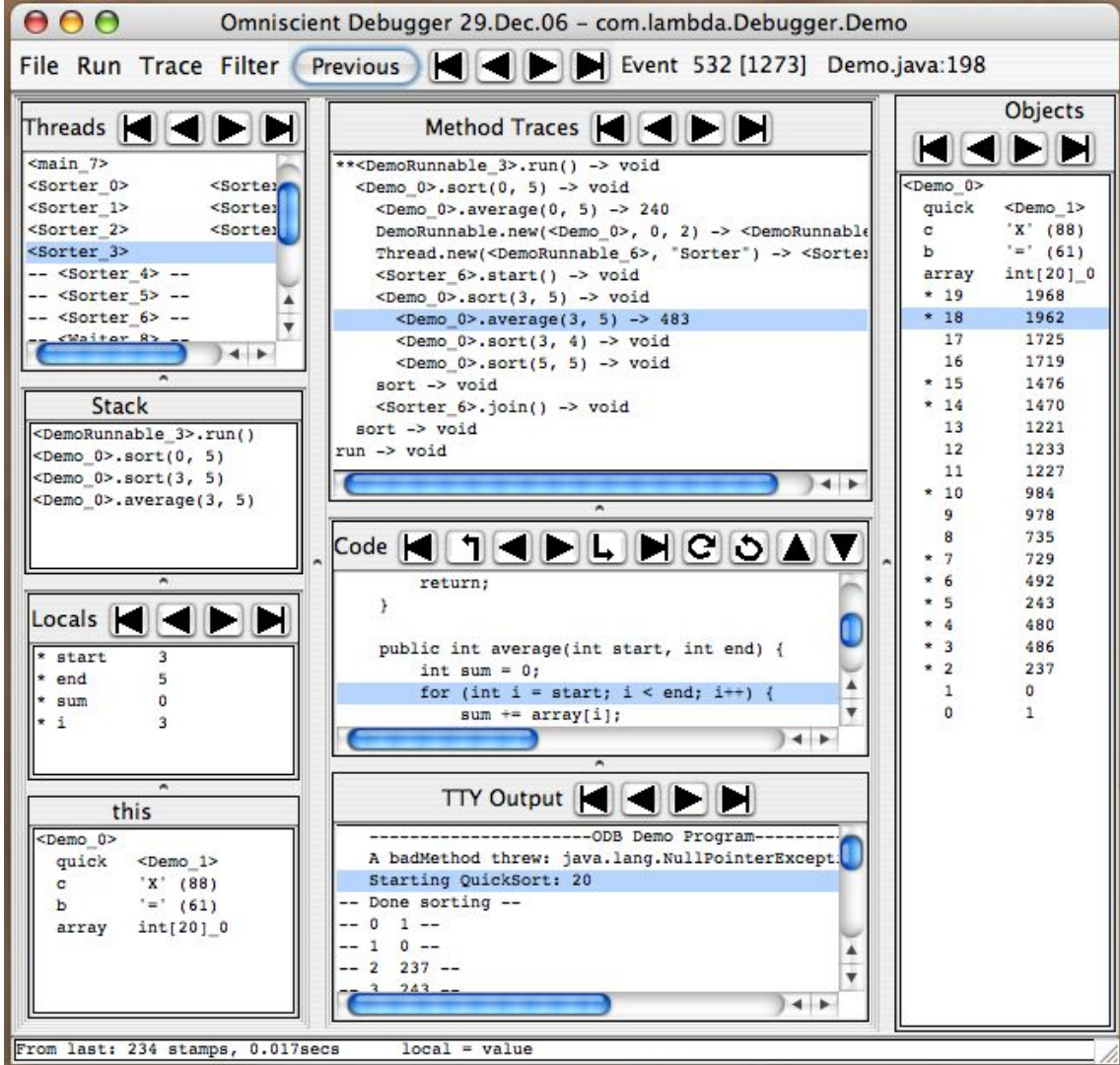
Clicking on a graphical object backs up the stepper to the event which drew it





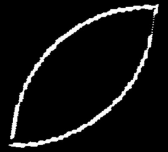
Omniscient Debugger

2007

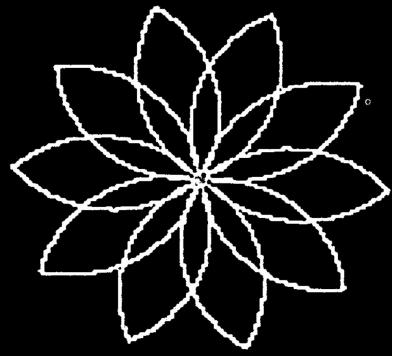


Logo 1967

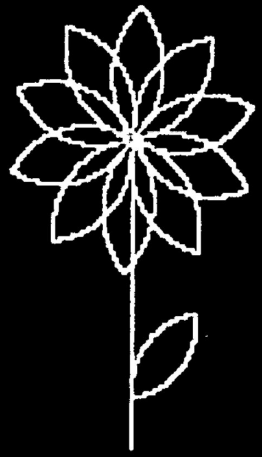
```
TO PETAL

QCIRCLE 50
RIGHT 90
QCIRCLE 50
RIGHT 90
END
```



```
TO NEWFLOWER
REPEAT 10
   PETAL
   RIGHT 360/10
END
```



```
TO PLANT
NEWFLOWER
BACK 50
PETAL
BACK 50
END
```



Leogo 1997

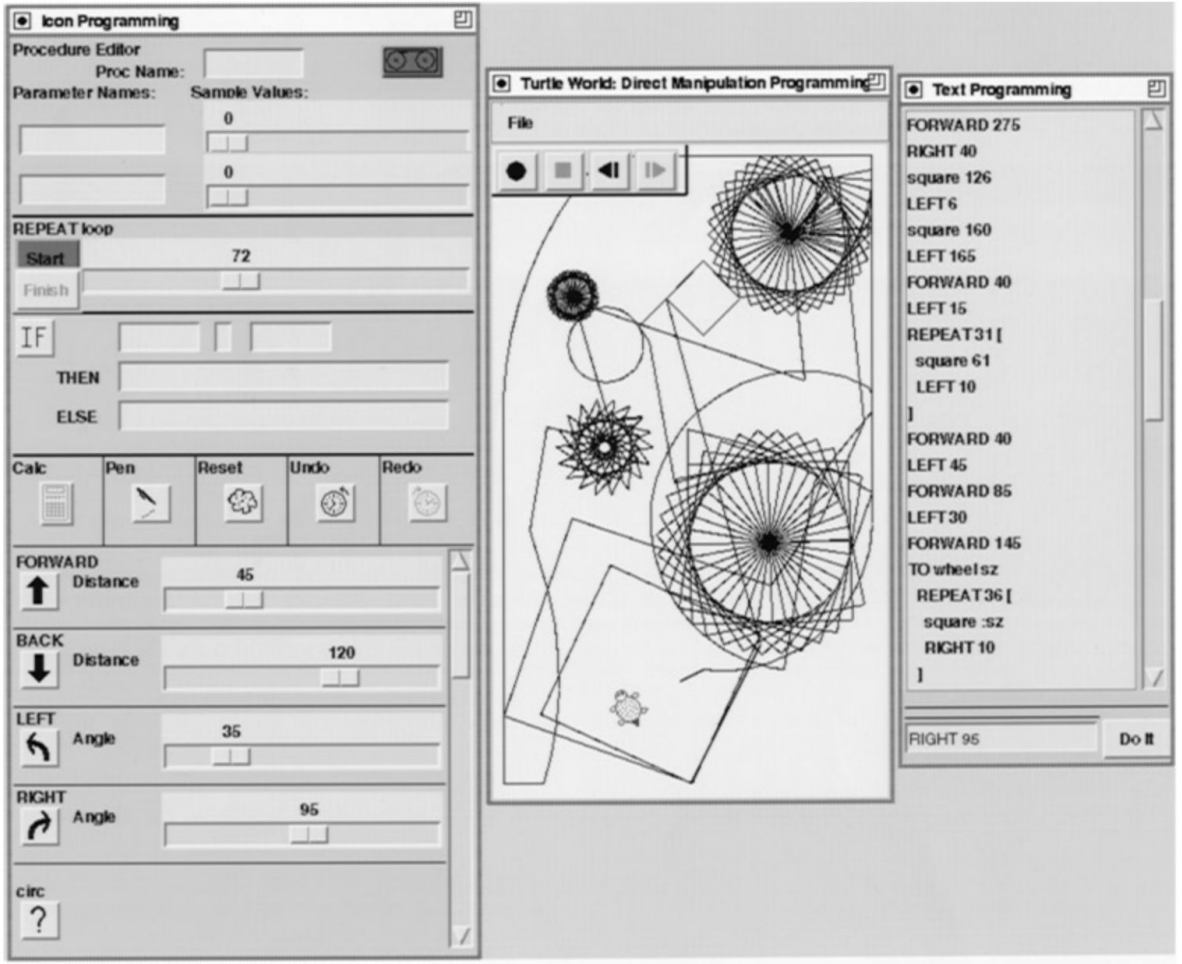


Figure 1. Leogo's three programming paradigms: iconic, direct manipulation, and textual.

Flogo II 2003

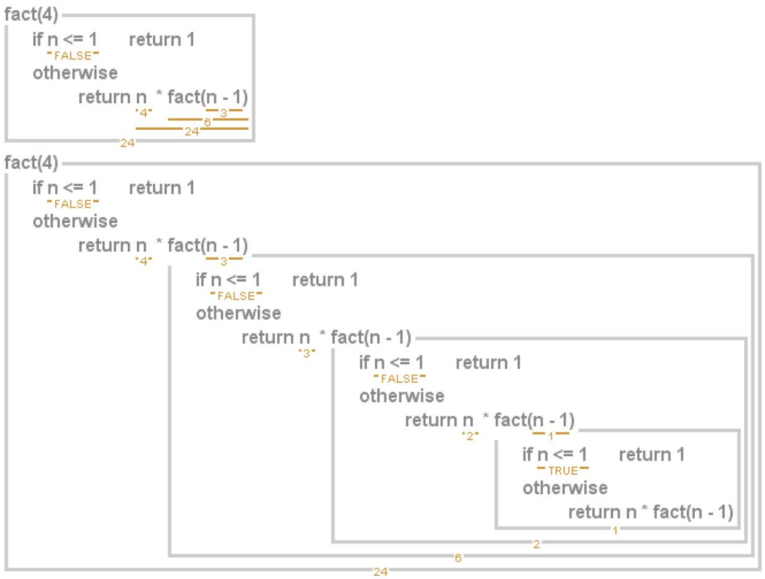


Fig. 5: A recursive function call opened up one level deep (above), and to its full depth (below).

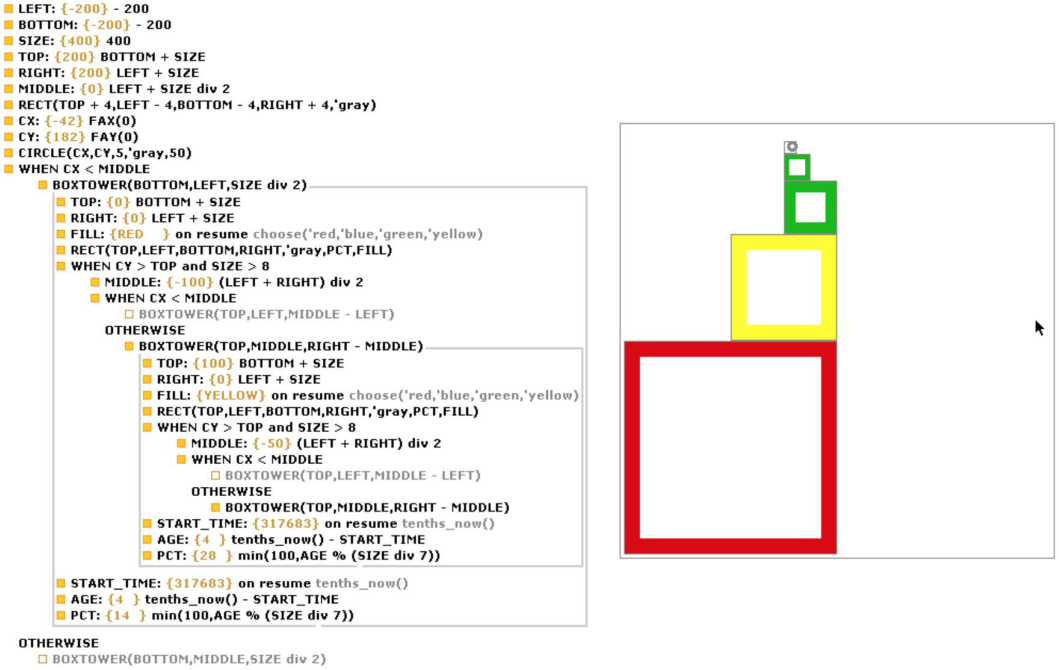


Fig. 9. Snapshot of live program execution of BOXTOWER program. Note live nested display of recursive calls.

Python Tutor 2015

Python 2.7

```
1  def listSum(numbers):
2    if not numbers:
3      return 0
4    else:
5      (f, rest) = numbers
6      return f + listSum(rest)
7
8  myList = (1, (2, (3, None)))
9  total = listSum(myList)
```

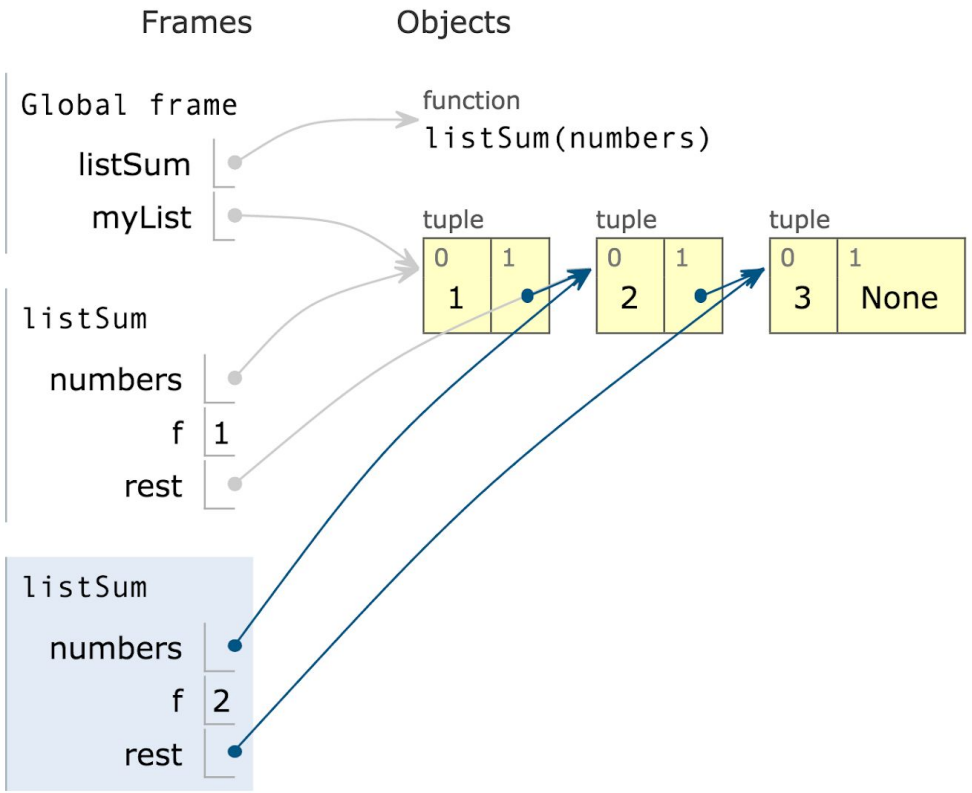
Edit code

line that has just executed

next line to execute

< Back Step 11 of 22 Forward >

Visualized using Python Tutor by Philip Guo



Legible Mathematics 2014

$(1 + 4 \times 3 + \text{length})^{\wedge 2} = 256$

$(1 + 12 + \text{length})^{\wedge 2}$

$(13 + \text{length})^{\wedge 2}$

$(13 + 3)^{\wedge 2}$

$(16)^{\wedge 2}$

$16^{\wedge 2}$

256

$(1 + 4 \times 3 + \text{length})^{\wedge 2} = 256$

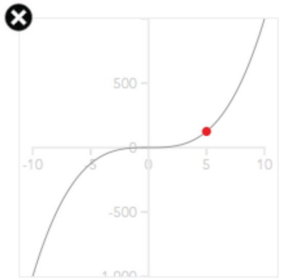
$(1 + 4 \times 3 + \text{length})^{\wedge 2} = 400$

16

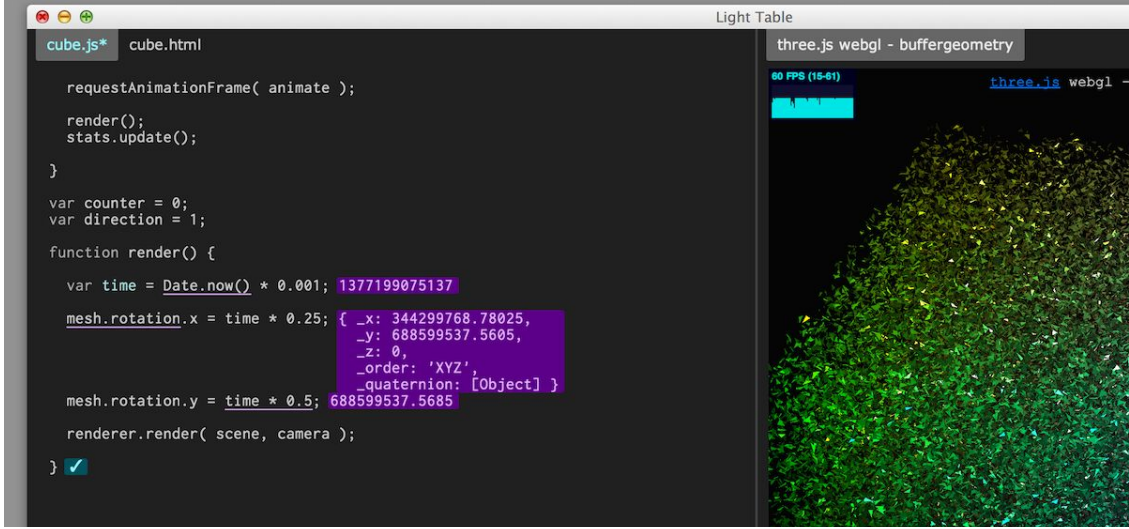
$\text{length} = 3 \quad = 3$

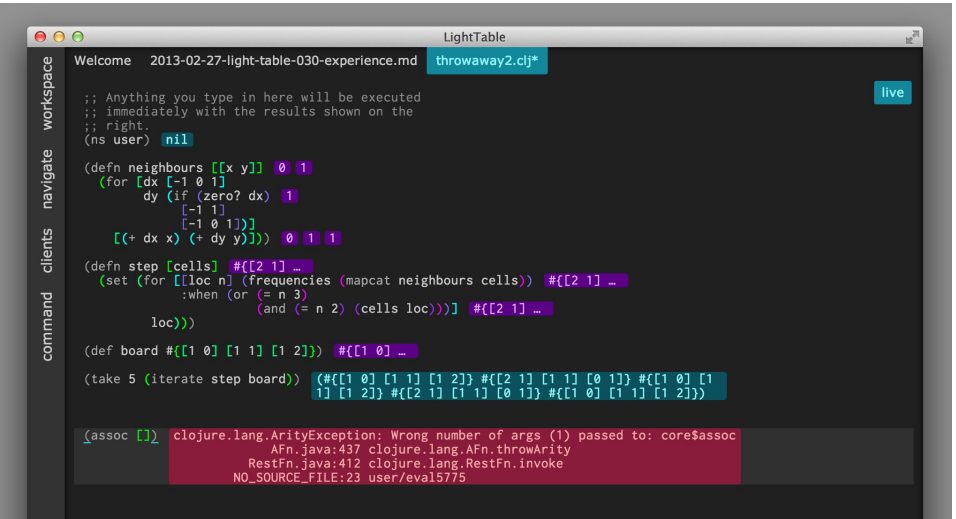
$(\text{length} + 2)^{\wedge 3} = 125$

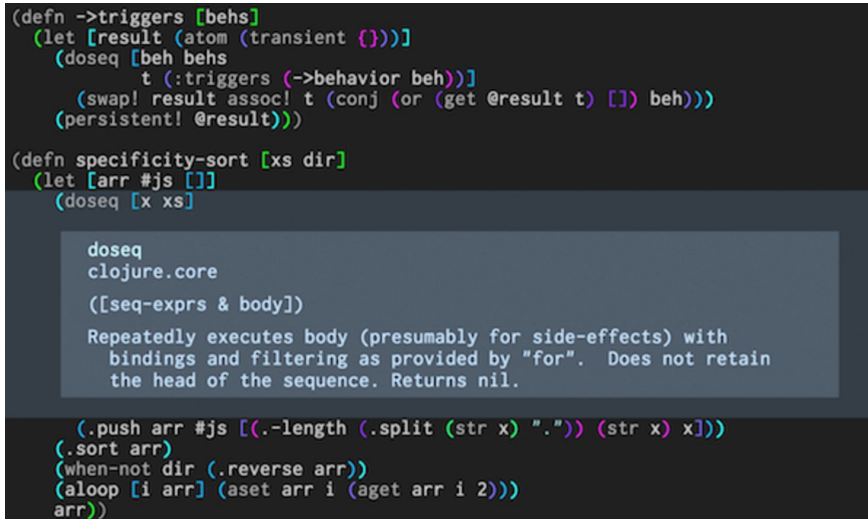
x y



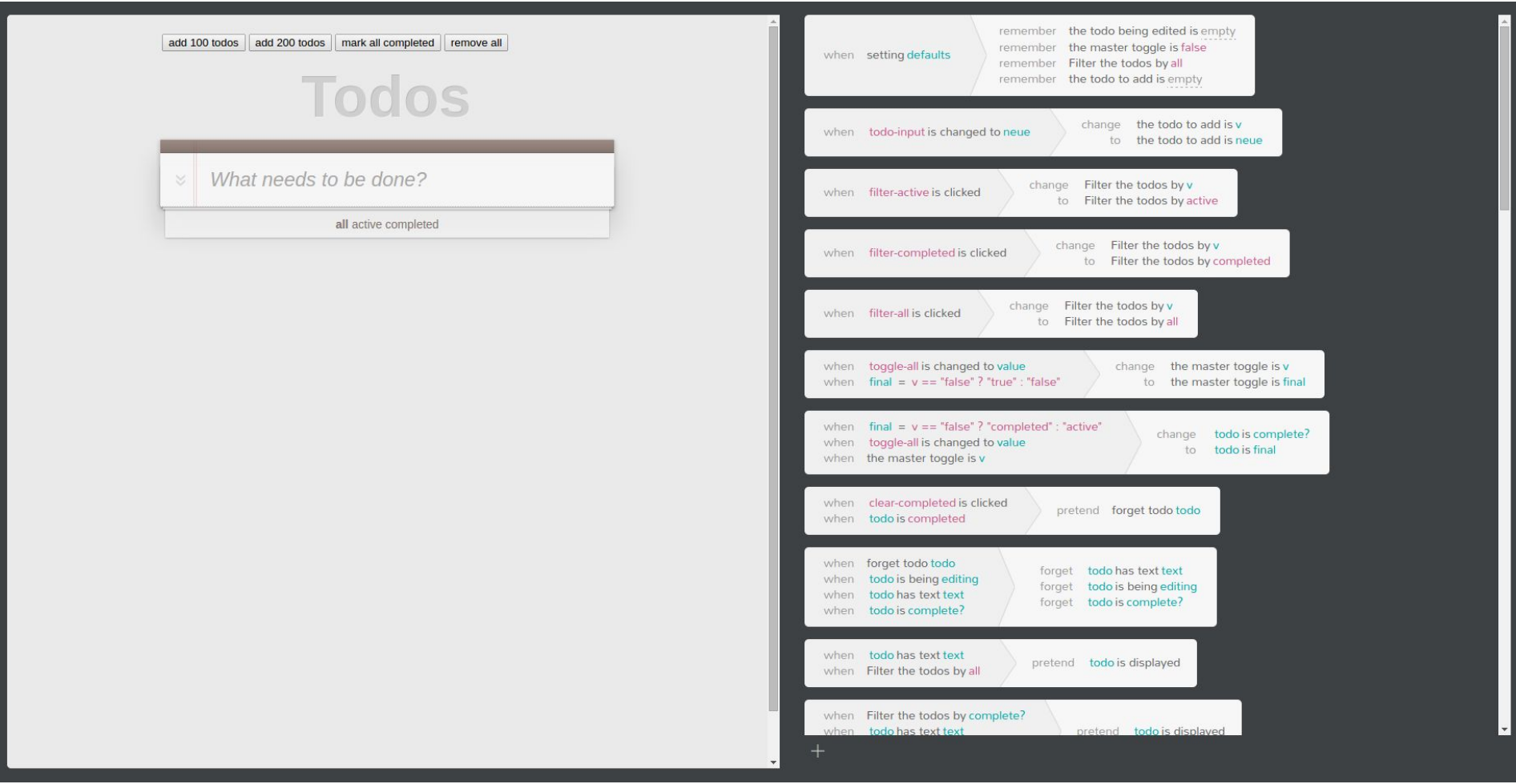
LightTable
2012







Eve 2014


add 100 todos
add 200 todos
mark all completed
remove all
Todos
What needs to be done?
all active completed
when setting defaults
remember the todo being edited is empty
remember the master toggle is false
remember Filter the todos by all
remember the todo to add is empty
when todo-input is changed to neue
change the todo to add is v
to the todo to add is neue
when filter-active is clicked
change Filter the todos by v
to Filter the todos by active
when filter-completed is clicked
change Filter the todos by v
to Filter the todos by completed
when filter-all is clicked
change Filter the todos by v
to Filter the todos by all
when toggle-all is changed to value
when final = v == "false" ? "true" : "false"
change the master toggle is v
to the master toggle is final
when final = v == "false" ? "completed" : "active"
when toggle-all is changed to value
when the master toggle is v
change todo is complete?
to todo is final
when clear-completed is clicked
when todo is completed
pretend forget todo todo
when forget todo todo
when todo is being editing
when todo has text text
when todo is complete?
forget todo has text text
forget todo is being editing
forget todo is complete?
when todo has text text
when Filter the todos by all
pretend todo is displayed
when Filter the todos by complete?
when todo has text text
pretend todo is displayed

Eve
2016

Flappy Eve

When a player starts the game, we commit a `#world`, a `#player`, and some `#obstacles`. These will keep all of the essential state of the game. All of this information could have been stored on the world, but for clarity we break the important bits of state into objects that they effect.

- The `#world` tracks the distance the player has travelled, the current game screen, and the high score.
- The `#player` stores his current y position and (vertical) velocity.
- The `obstacles` have their (horizontal) offset and gap widths. We put distance on the world and only keep two obstacles; rather than moving the player through the world, we keep the player stationary and move the world past the player. When an obstacle goes off screen, we will wrap it around, update the placement of its gap, and continue on.

Setup

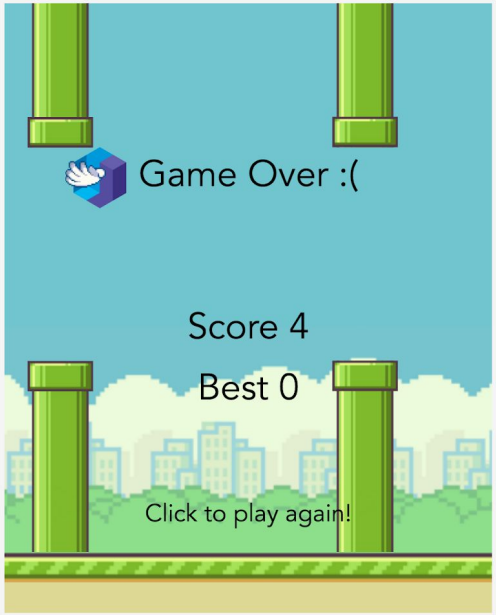
Add a flappy eve and a world for it to flap in:

```
commit
  [#player #self name: "eve" x: 25 y: 50 velocity: 0]
  [#world screen: "menu" frame: 0 distance: 0 best: 0 gravity: -0.061]
  [#obstacle gap: 35 offset: 0]
  [#obstacle gap: 35 offset: -1]
```

Next we draw the backdrop of the world. The player and obstacle will be drawn later based on their current state. Throughout the app we use resources from @bhauman's flappy bird demo in clojure. Since none of these things change over time, we commit them once when the player starts the game.

Draw the game world!

```
search
  world = [#world]

commit @browser
  world <- [#div style: [user-select: "none" -webkit-user-select: "none" -
moz-user-select: "none"]  children:
    [#svg #game-window viewBox: "10 0 80 100", width: 480 children:
      [#rect x: 0 y: 0 width: 100 height: 53 fill: "rgb(112, 197, 206)" sort:
```



ConQuer
1997

Q2 ✓Employee
+– drives Car
+– works for ✓Branch

Q3 ✓USbranch
+– **not** achieved Rank = 1 in Year < 1998
+– **is** Branch
+– is headed by Employee
+– has ✓EmployeeName
+– **maybe** drives ✓Car

Q4 ✓$Employee_1$
+– lives in $City_1$
+– was born in $Country_1$
+– supervises $Employee_2$
+– lives in $City_1$
+– was born in $Country_2$ <> $Country_1$

Q5 ✓Employee
+– owns Car_1
+– **not** drives Car_1
+– **count**(Car_1) **for** Employee > 1

SIEUFERD 2016

Ethanol-Related Organizations

OpenSecrets Name	Total Amounts in 2012-dollars	Lobbying Reports: Type	Amount	Year	CPI Lookup: Consumer Price Idx
Low Carbon Synthetic Fuels Association	41,170	q2	40,000	2011	0.9716
Maple Etanol SRL	74,870	q1	10,000	2010	0.9560
		q4	20,000	2009	0.9315
		q3	30,000	2009	0.9315
		q2	10,000	2009	0.9315
Algal Biomass Org	=sum([Amount] / [Consumer Price Idx])				1.0000
					1.0000
					0.9716
					0.9716
					0.9560
American Council on Renewable Energy	246,962	q1t	20,000	2009	0.9315
		q1	30,000	2009	0.9315
		q4	30,000	2008	0.9313
		q4	40,000	2008	0.9313
		q3	30,000	2008	0.9313

Figure 1: Spreadsheet-style formula editing in a nested relational result. The query structure is encoded in the table header, which shows three joined table instances (bold labels), one-to-many relationships (≺), sorting (⫧), active filters (⏷, ↖), and a formula (f_x).

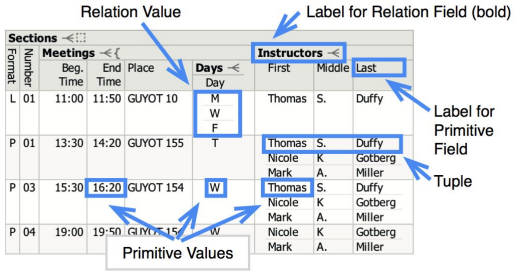


Figure 3: Terminology of the nested relational data model, illustrated on a nested table layout.

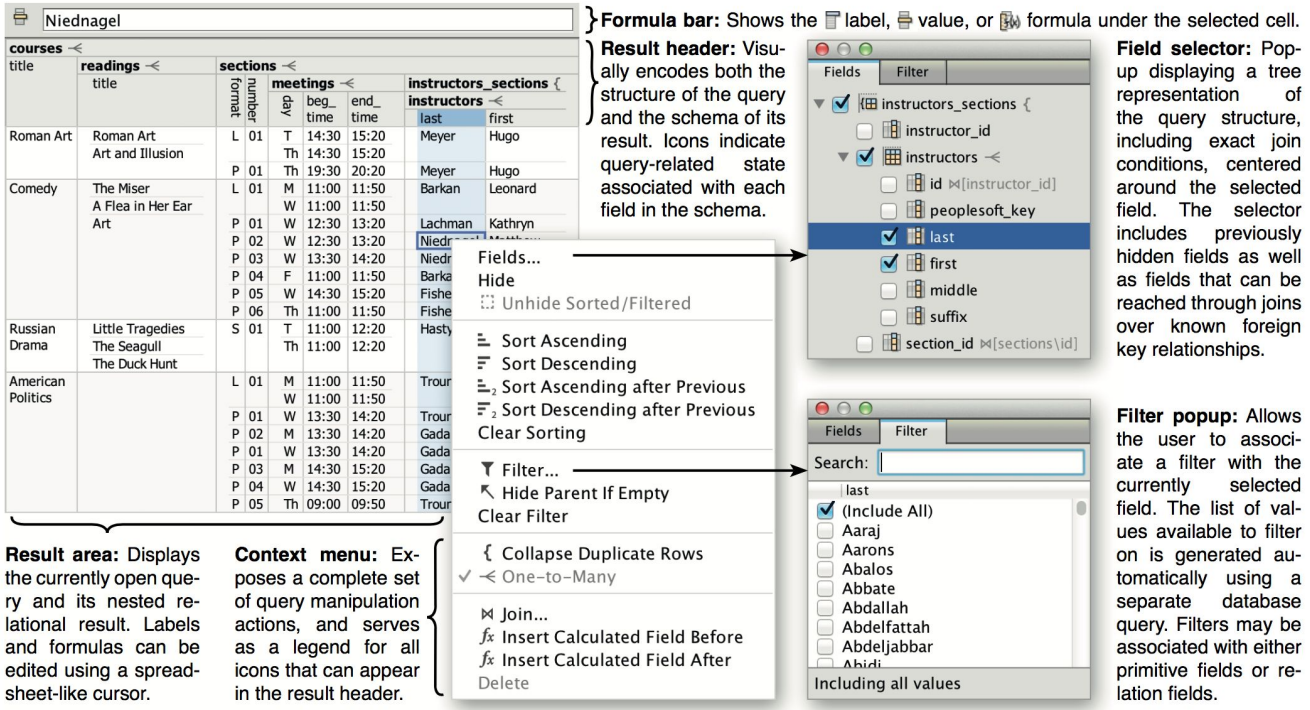
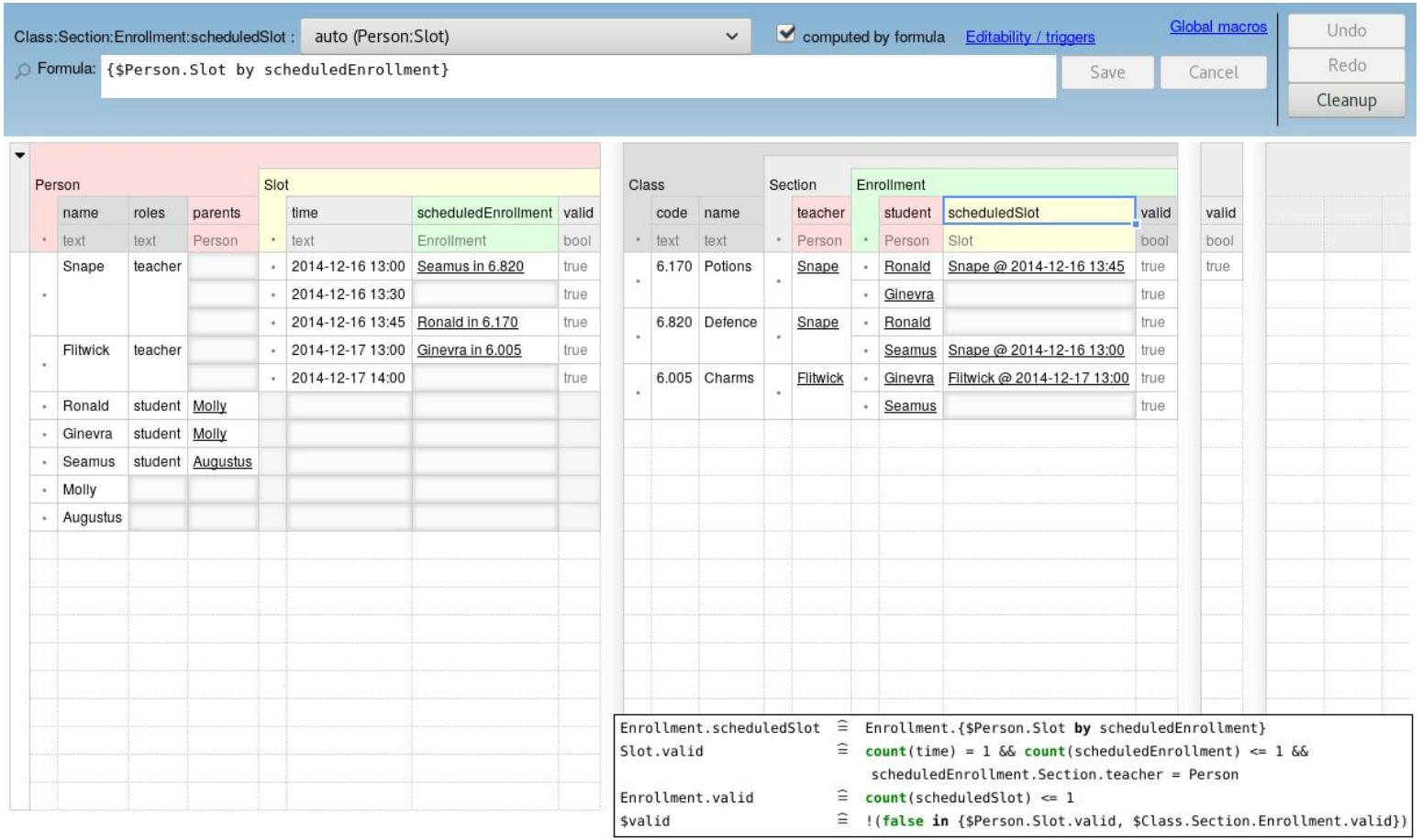
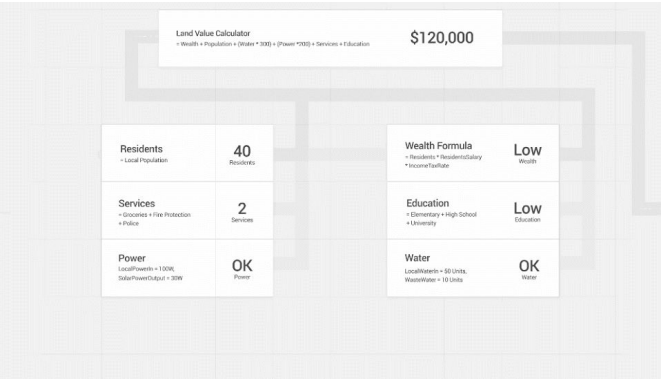


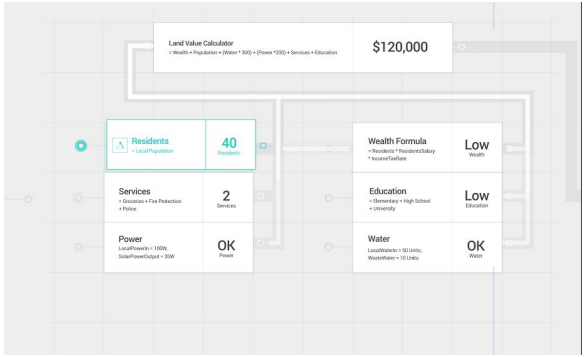
Figure 2: The SIEUFERD query interface. To create queries, users start from a simple tabular view of a table in the database and add filters, formulas, and nested relations. The integrated result and query representation is displayed continuously as the user interacts with the data. The particular query above instantiates six database tables (one per nested relation), contains five joins (each child relation against its parent), and is evaluated using five generated SQL queries (one for each one-to-many relationship ≺). This query was constructed purely by checking off the appropriate fields and foreign key relationships in the field selector.

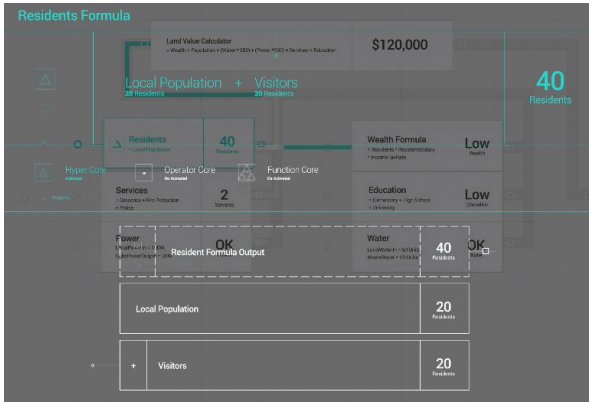
Object Spreadsheets 2016

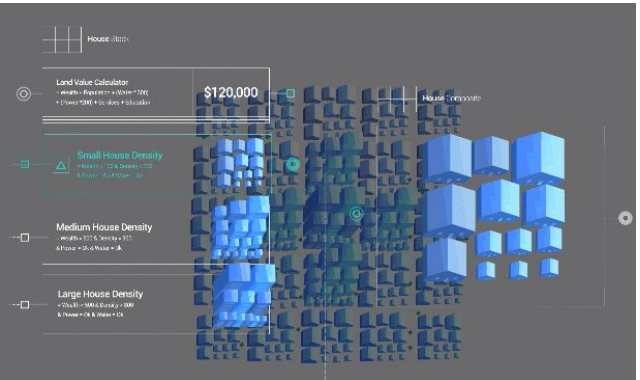


Grid 2013

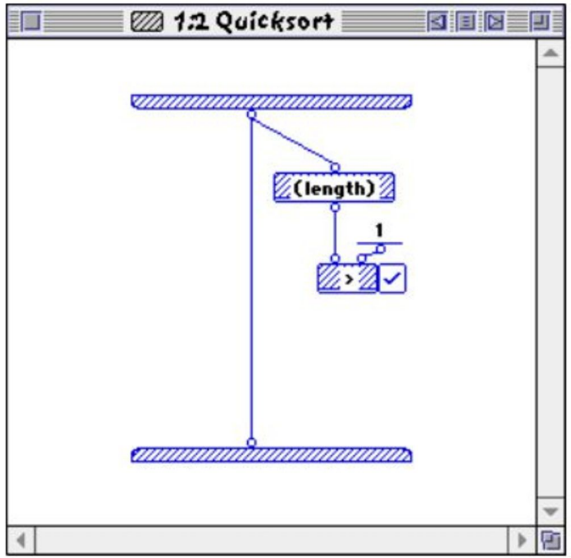

Land Value Calculator
$120,000
Residents
40
Residents
Services
2
Services
Power
OK
Power
Wealth Formula
Low
Wealth
Education
Low
Education
Water
OK
Water

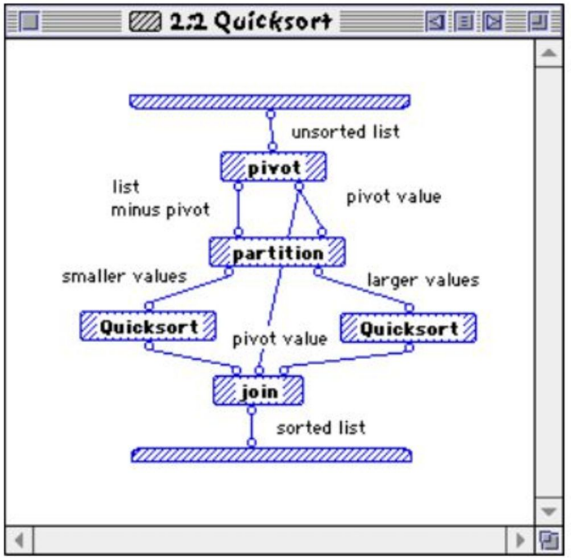

Land Value Calculator
$120,000
Residents
40
Residents
Services
2
Services
Power
OK
Power
Wealth Formula
Low
Wealth
Education
Low
Education
Water
OK
Water


Residents Formula
Land Value Calculator
$120,000
Local Population + Visitors
40
Residents
Hyper Core
Operator Core
Function Core
Resident Formula Output
Local Population
20
Visitors
20

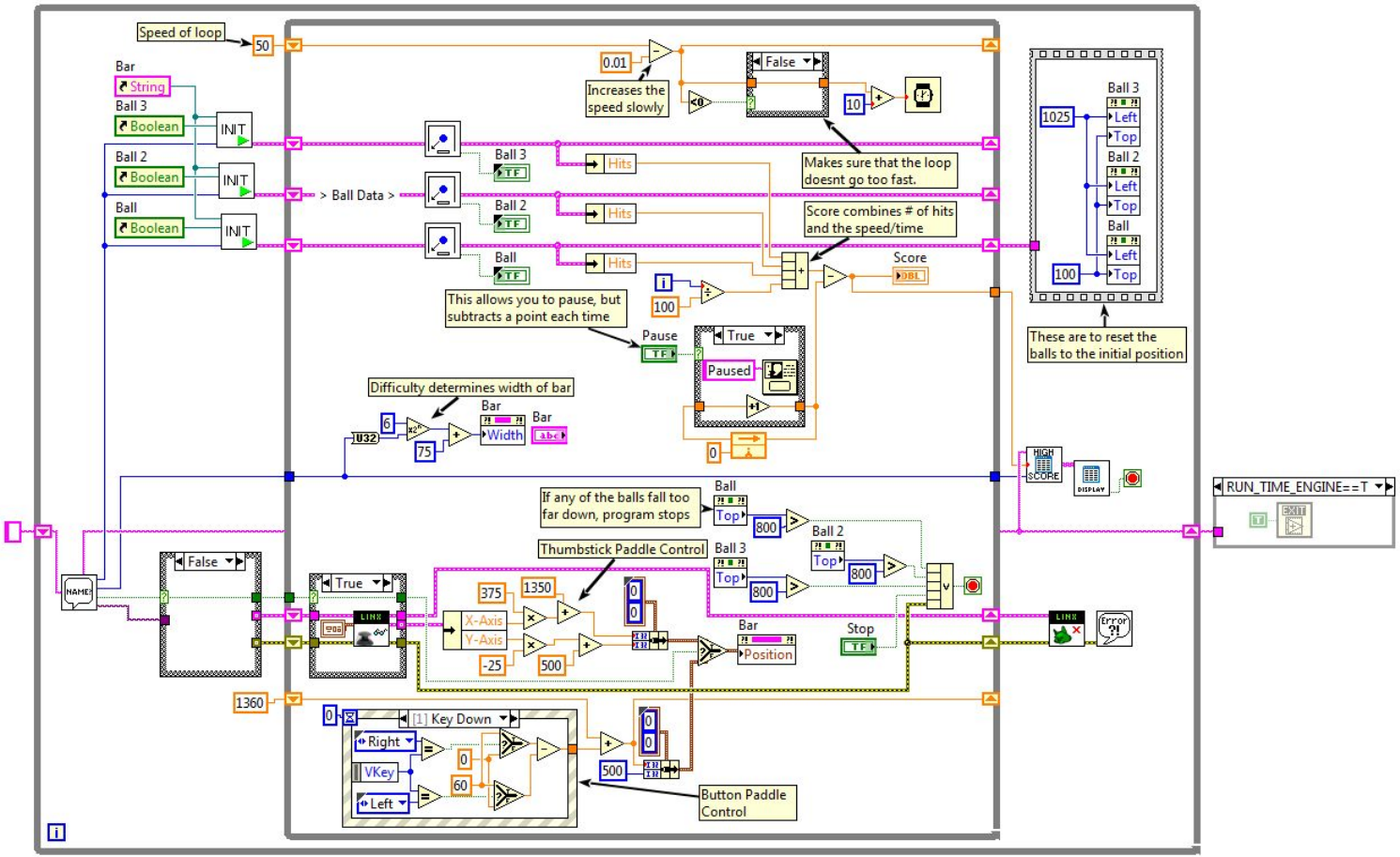

Land Value Calculator
$120,000
Small House Density
Medium House Density
Large House Density

ProGraph 1983





LabVIEW 1986



Fabrik 1988

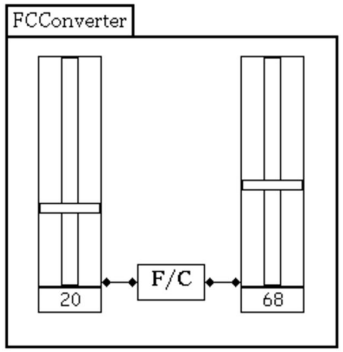


Figure 2a. *Bidirectional diagram using two Slider controls to achieve a Fahrenheit-to-Centigrade converter.*

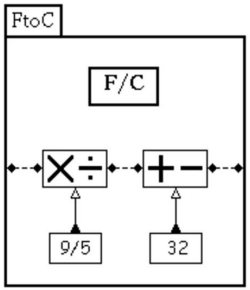


Figure 2b. *Internal diagram for the F/C component used in the diagram at left.*

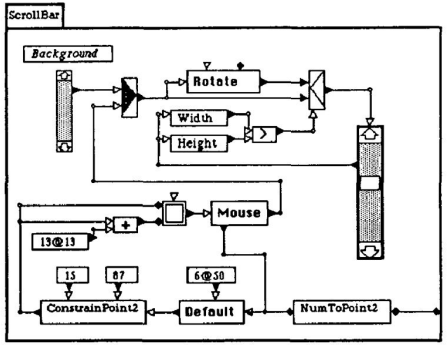


Figure 4a. A simple scrollbar diagram. Logic is provided for rotating when the image is wider than high.

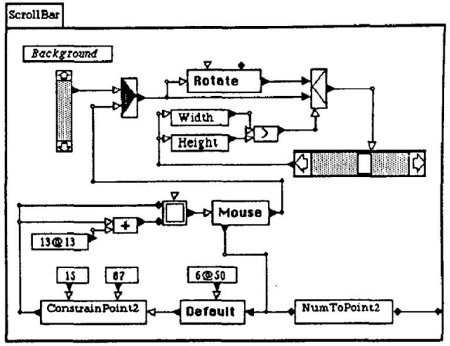


Figure 4b. Demonstrating the rotation logic. All display and mouse-tracking operations are properly scaled and rotated.

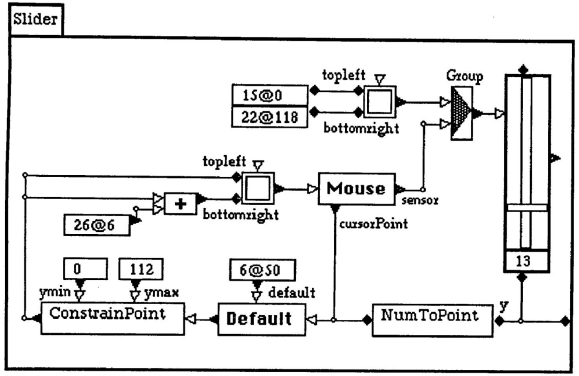


Figure 3. A Fabrik diagram computes the image for the slider in figure 2. The Mouse component sensitizes the slider image to support input as well as output.

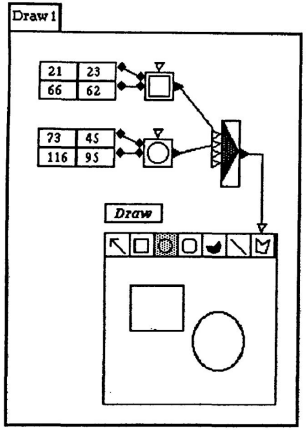


Figure 5a. The Draw component automatically lays out diagrams as the user creates a drawing.

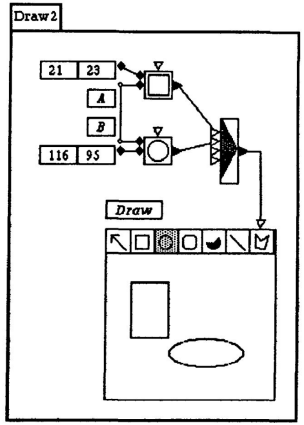


Figure 5b. By editing the generative diagram, the top-left of the oval is tied to the bottom-right of the rectangle.

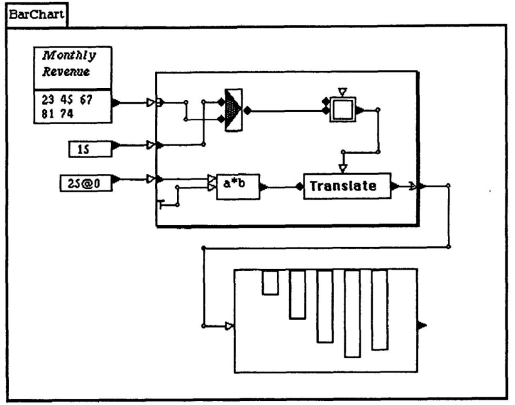
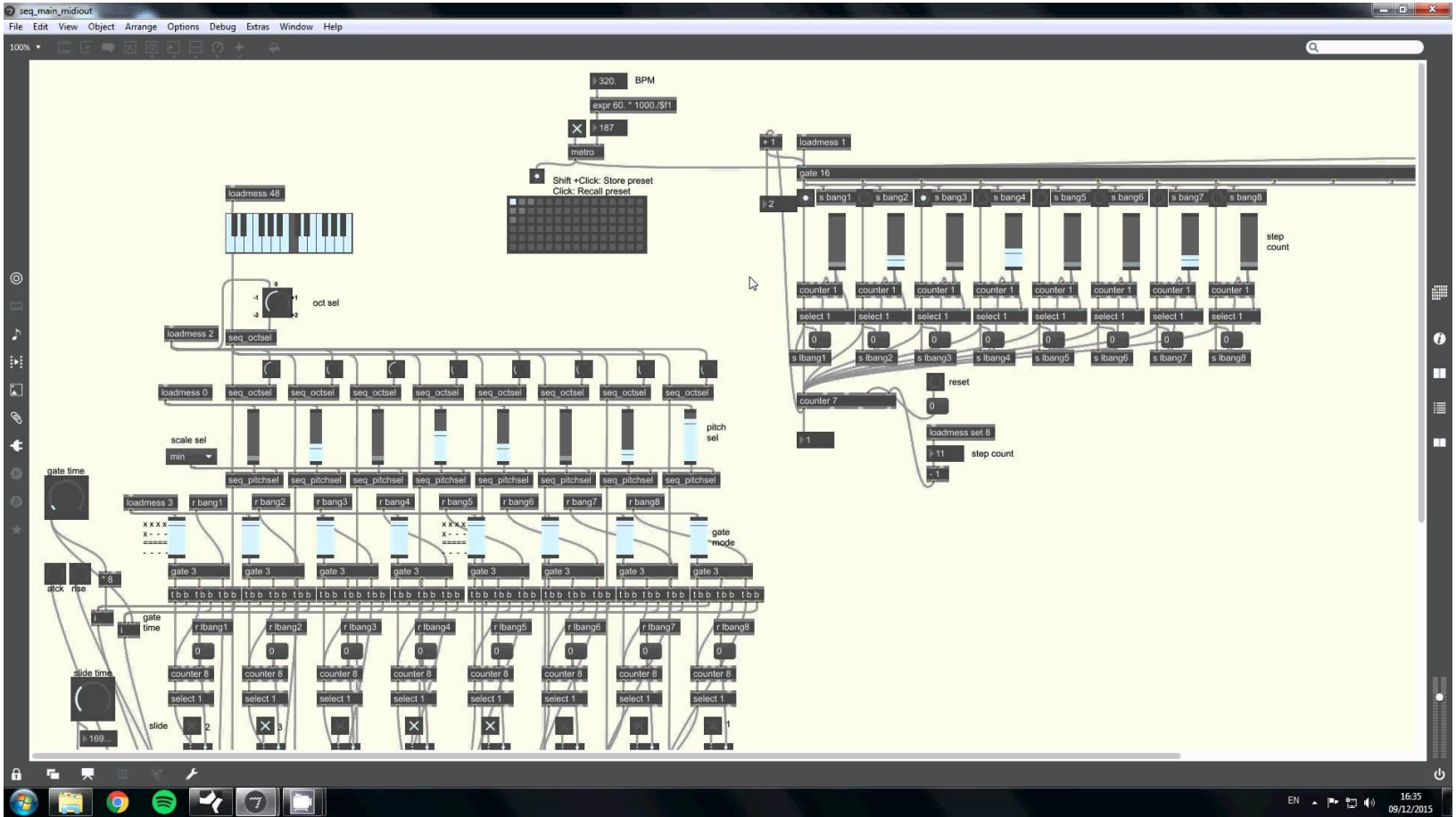
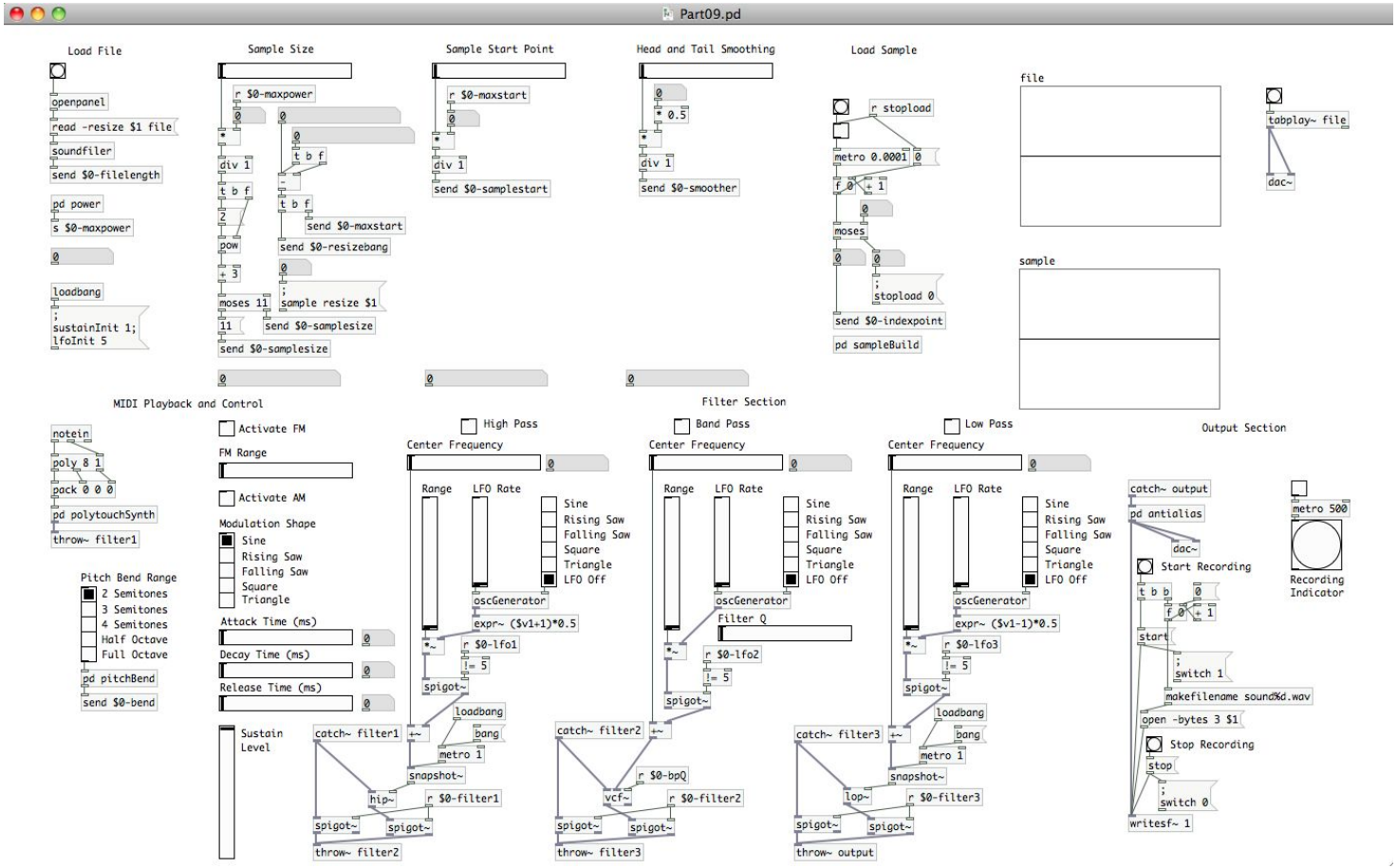


Figure 6. Streaming gateways provide iterative capability. Here numbers are converted to rectangles, resulting in a simple bar chart.

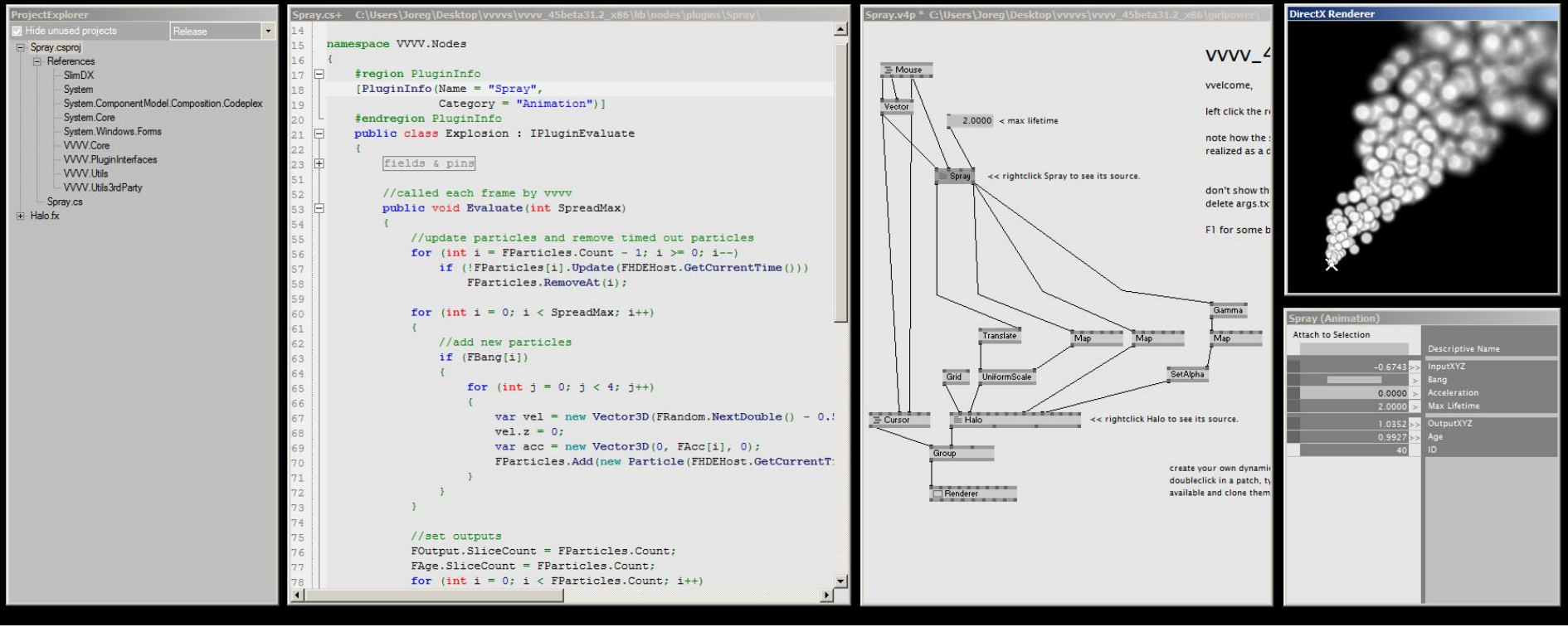
MaxMSP 1990


seq_main_midiout
File Edit View Object Arrange Options Debug Extras Window Help
320. BPM
expr 60. * 1000./$f1
187
metro
Shift +Click: Store preset
Click: Recall preset
loadmess 48
oct sel
loadmess 2
seq_octsel
loadmess 0
scale sel
min
pitch sel
gate time
loadmess 3
r bang1
r bang2
r bang3
r bang4
r bang5
r bang6
r bang7
r bang8
gate mode
atck
rise
gate time
r lbang1
r lbang2
r lbang3
r lbang4
r lbang5
r lbang6
r lbang7
r lbang8
slide time
slide
loadmess 1
gate 16
s bang1
s bang2
s bang3
s bang4
s bang5
s bang6
s bang7
s bang8
step count
counter 1
select 1
s lbang1
s lbang2
s lbang3
s lbang4
s lbang5
s lbang6
s lbang7
s lbang8
reset
counter 7
loadmess set 8
step count
counter 8
EN
16:35
09/12/2015

Pure Data 1996

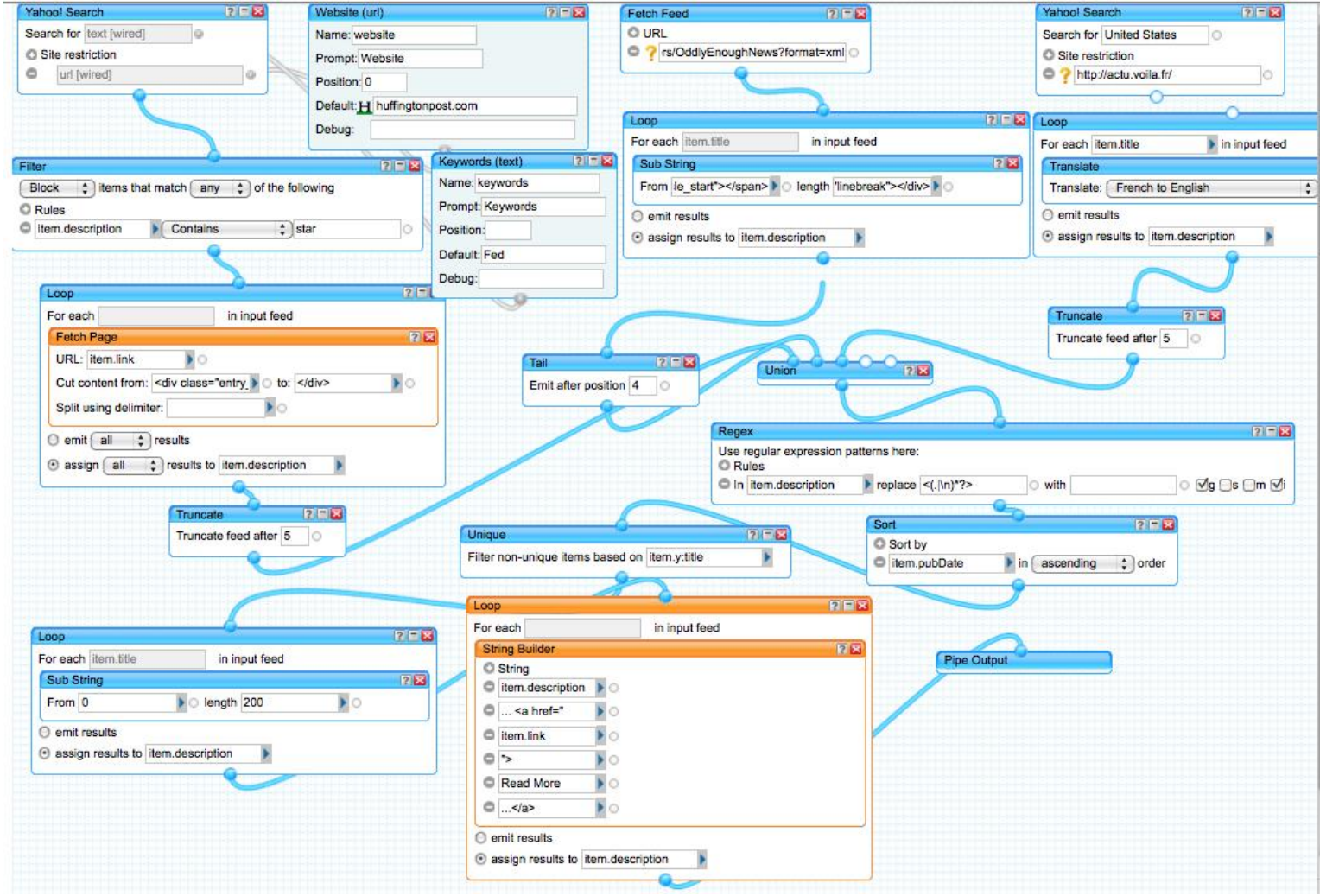


vvvv 1998

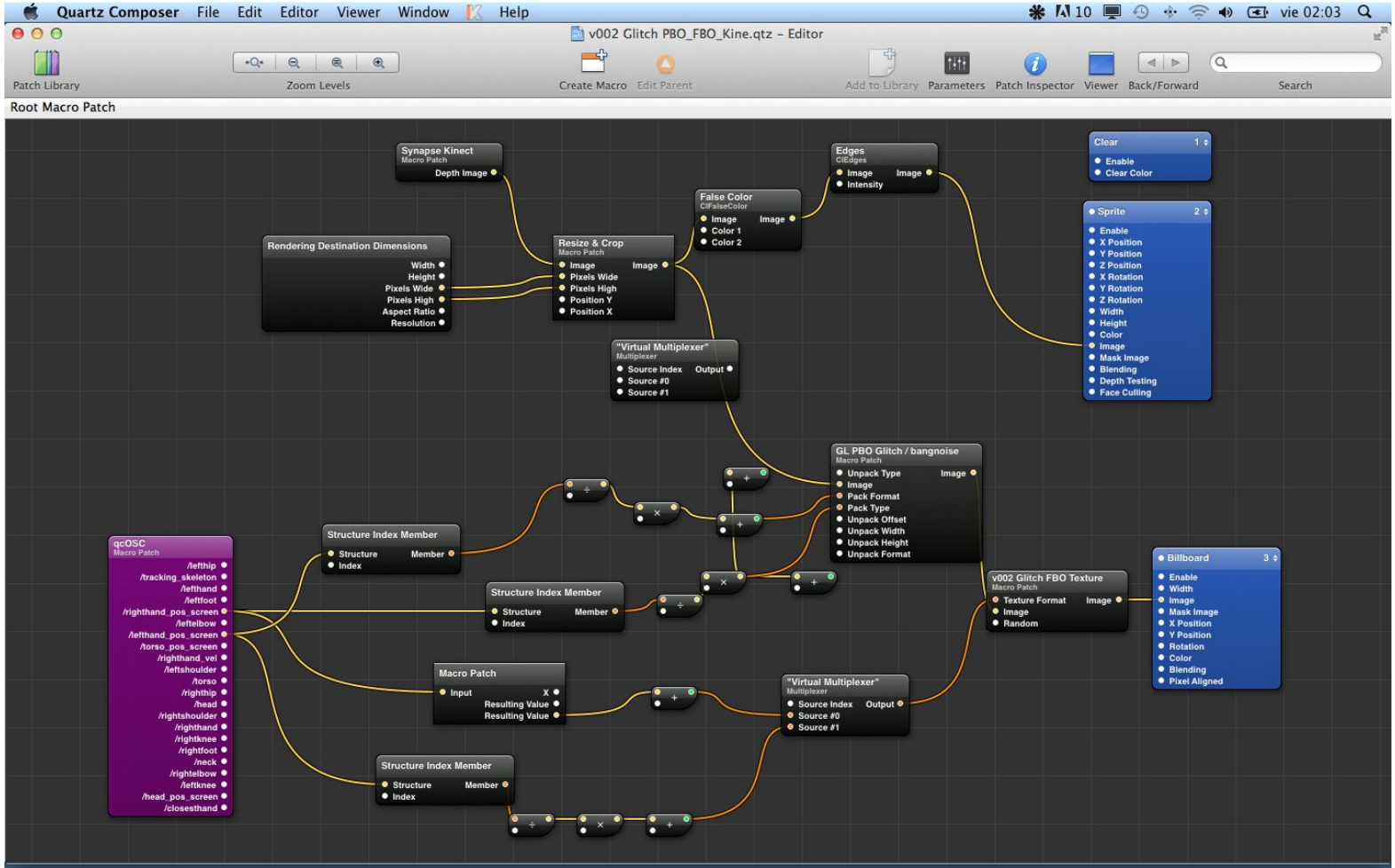

ProjectExplorer
Hide unused projects
Release
Spray.csproj
References
SlimDX
System
System.ComponentModel.Composition.Codeplex
System.Core
System.Windows.Forms
VVVV.Core
VVVV.PluginInterfaces
VVVV.Utils
VVVV.Utils3rdParty
Spray.cs
Halo.fx
Spray.cs+ C:\Users\Joreg\Desktop\vvvvs\vvvv_45beta31.2_x86\lib\nodes\plugins\Spray\
namespace VVVV.Nodes
{
#region PluginInfo
[PluginInfo(Name = "Spray",
Category = "Animation")]
#endregion PluginInfo
public class Explosion : IPluginEvaluate
{
fields & pins
//called each frame by vvvv
public void Evaluate(int SpreadMax)
{
//update particles and remove timed out particles
for (int i = FParticles.Count - 1; i >= 0; i--)
if (!FParticles[i].Update(FHDEHost.GetCurrentTime()))
FParticles.RemoveAt(i);
for (int i = 0; i < SpreadMax; i++)
{
//add new particles
if (FBang[i])
{
for (int j = 0; j < 4; j++)
{
var vel = new Vector3D(FRandom.NextDouble() - 0.
vel.z = 0;
var acc = new Vector3D(0, FAcc[i], 0);
FParticles.Add(new Particle(FHDEHost.GetCurrentT
}
}
}
//set outputs
FOutput.SliceCount = FParticles.Count;
FAge.SliceCount = FParticles.Count;
for (int i = 0; i < FParticles.Count; i++)
Spray.v4p * C:\Users\Joreg\Desktop\vvvvs\vvvv_45beta31.2_x86\girlpower\
Mouse
Vector
2.0000
< max lifetime
Spray
<< rightclick Spray to see its source.
Gamma
Translate
Map
Map
Map
Grid
UniformScale
SetAlpha
Cursor
Halo
<< rightclick Halo to see its source.
Group
Renderer
vvvv_4
vvelcome,
left click the r
note how the
realized as a c
don't show th
delete args.tx
F1 for some b
create your own dynami
doubleclick in a patch, t
available and clone them
DirectX Renderer
Spray (Animation)
Attach to Selection
Descriptive Name
-0.6743
InputXYZ
Bang
0.0000
Acceleration
2.0000
Max Lifetime
1.0352
OutputXYZ
0.9927
Age
40
ID

Yahoo Pipes

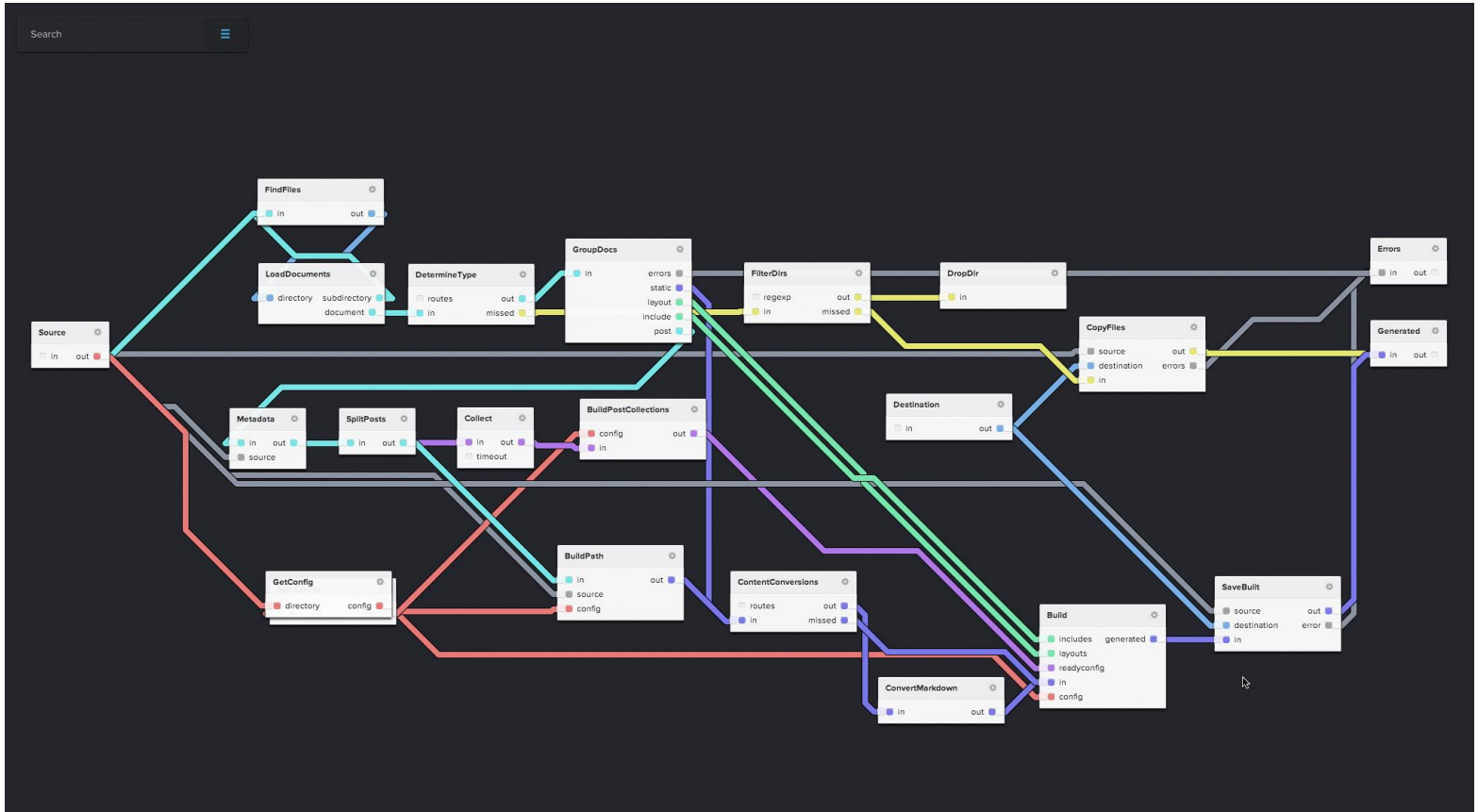
2007



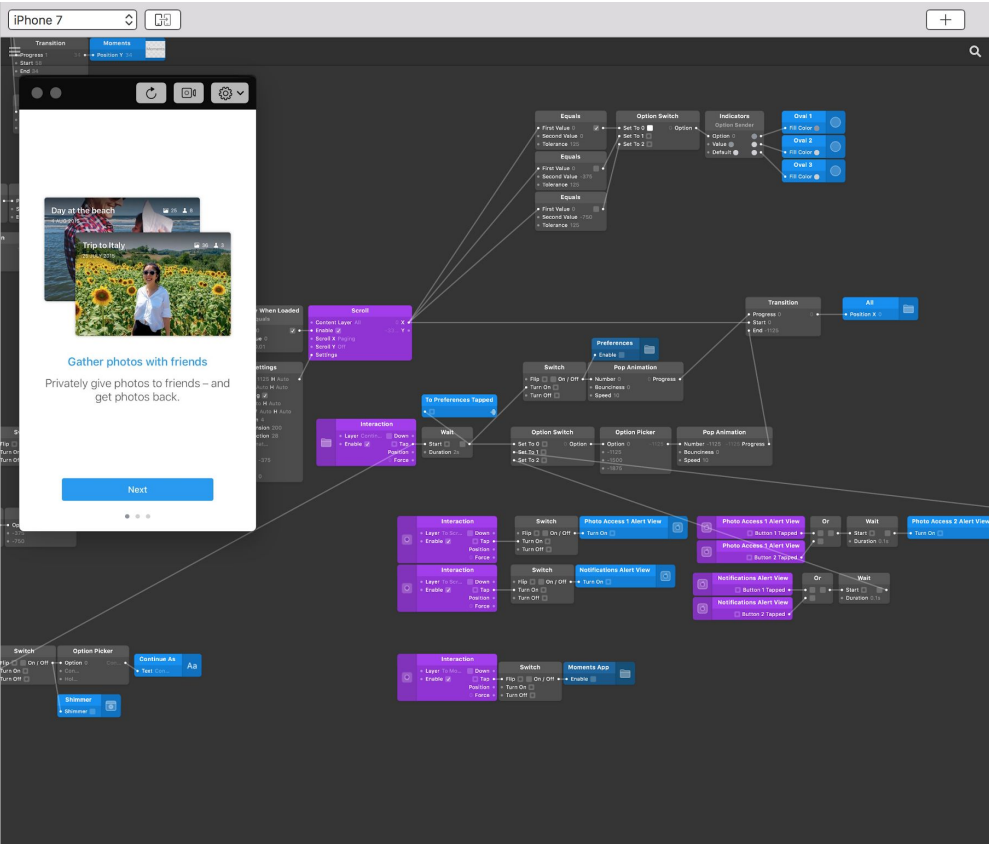
Quartz Composer 2005

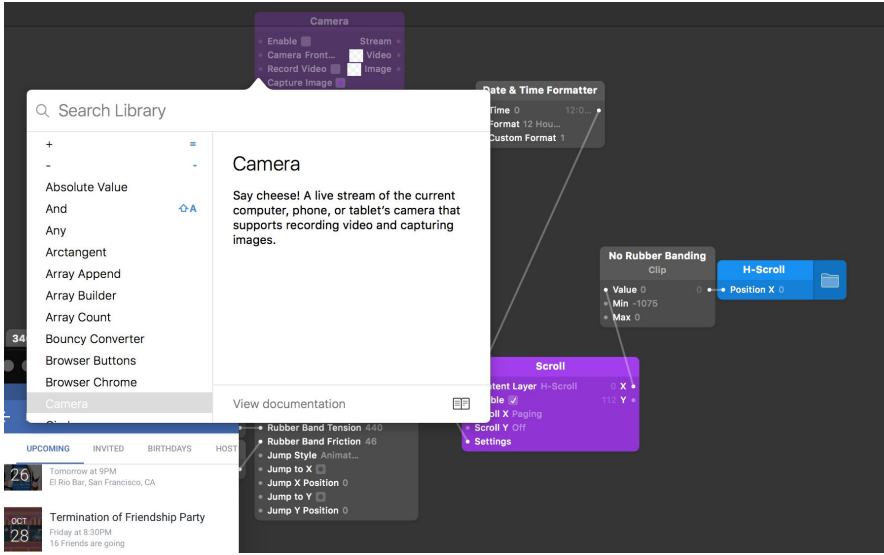


NoFlo 2014

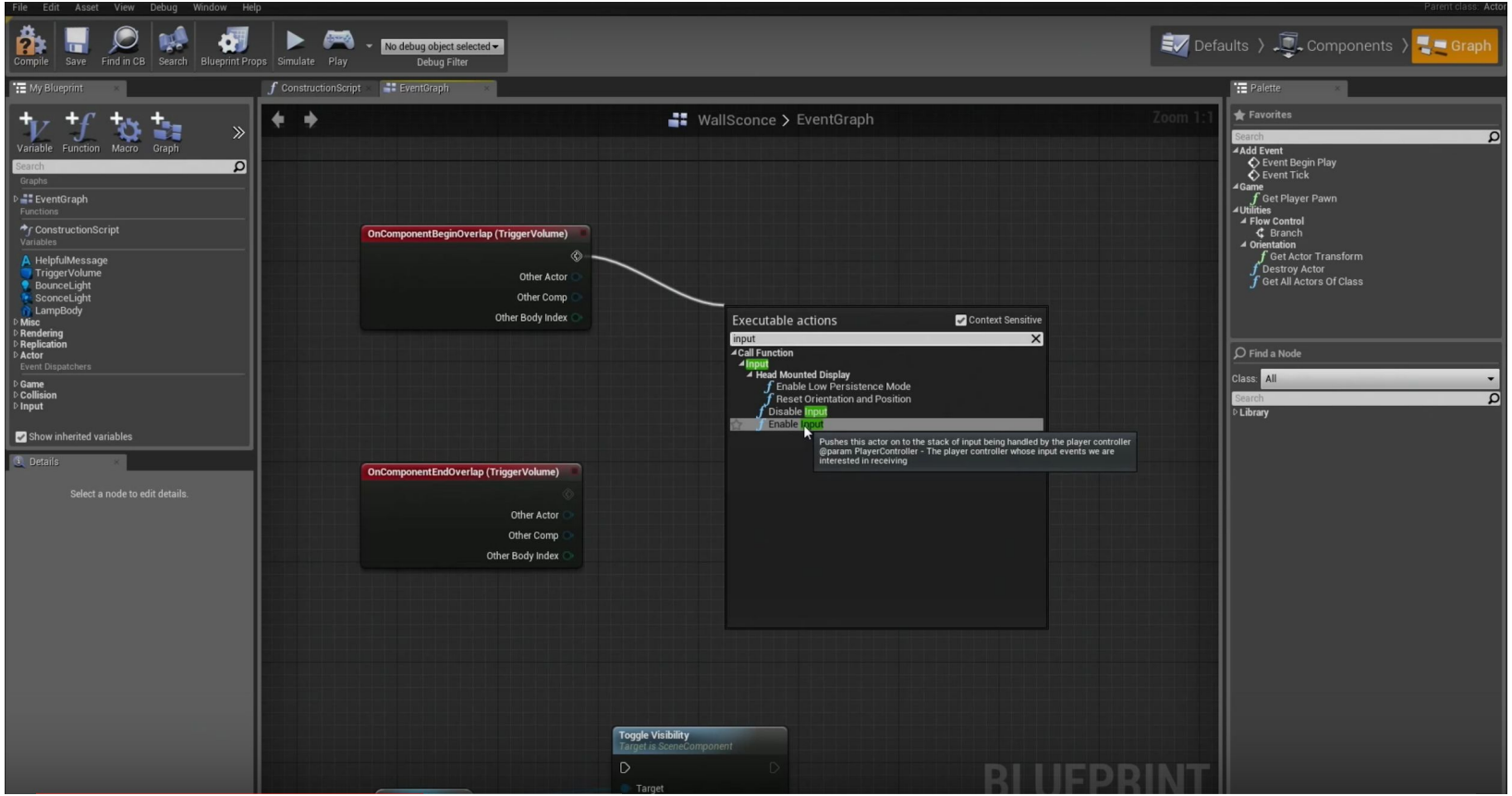

Search
FindFiles
in
out
LoadDocuments
directory
subdirectory
document
DetermineType
routes
out
in
missed
GroupDocs
in
errors
static
layout
include
post
FilterDirs
regexp
out
in
missed
DropDir
in
Errors
in
out
Source
in
out
CopyFiles
source
destination
in
out
errors
Generated
in
out
Metadata
in
out
source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

Origami 2017

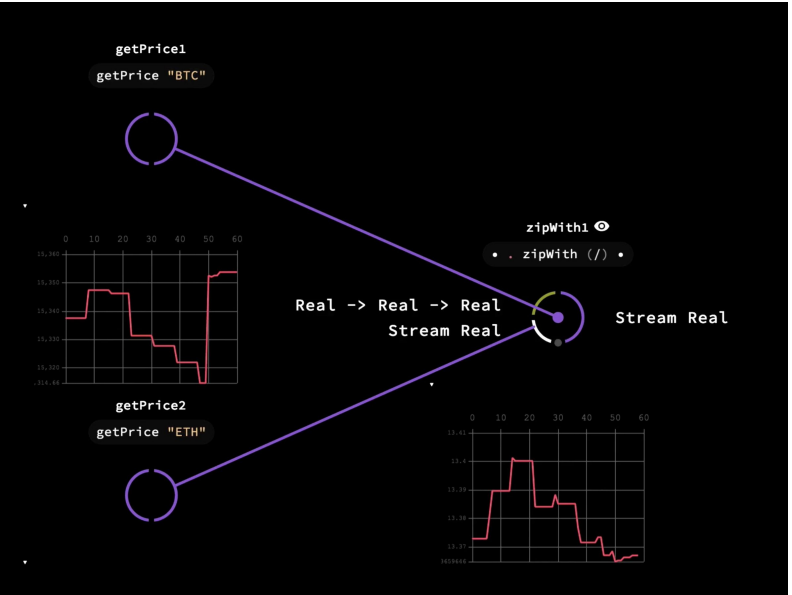


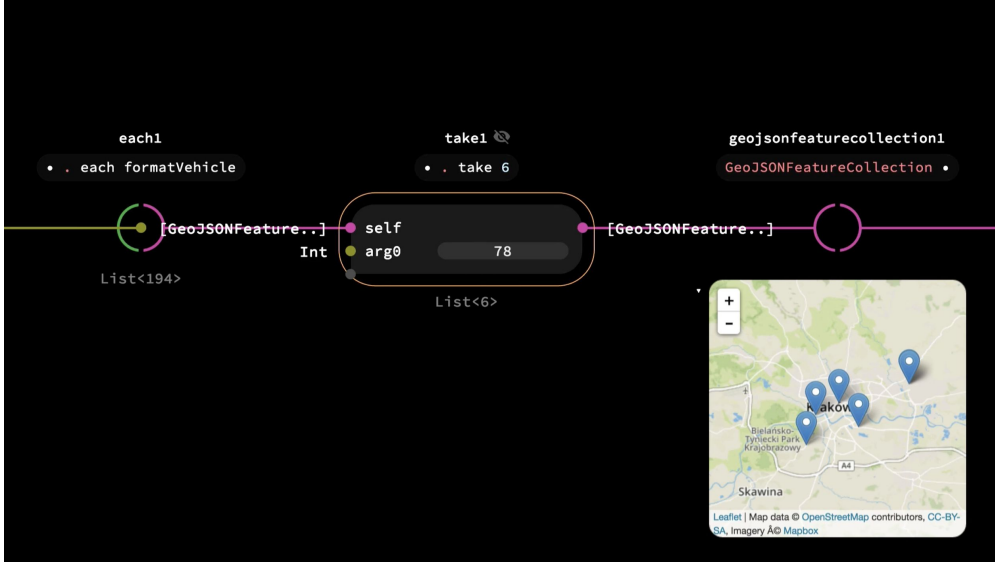


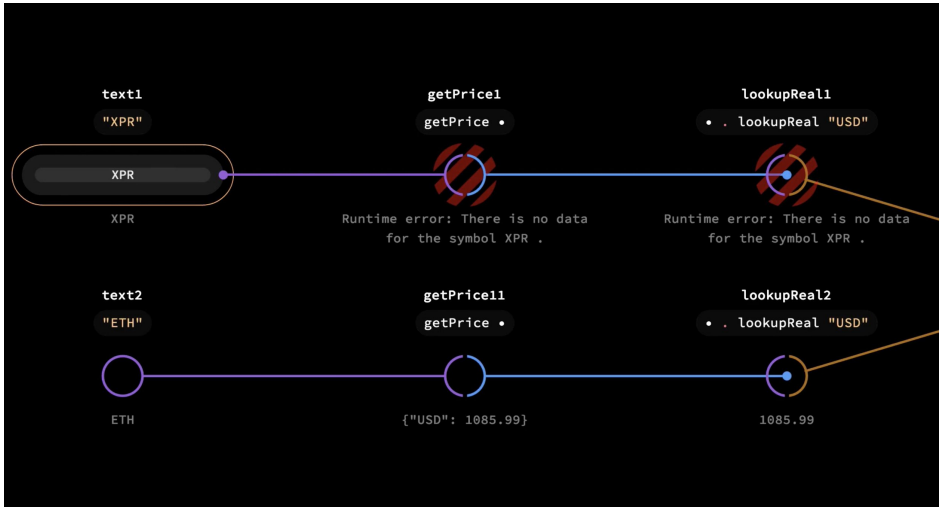
Unreal Blueprint 2014
(contextual autocompletion)



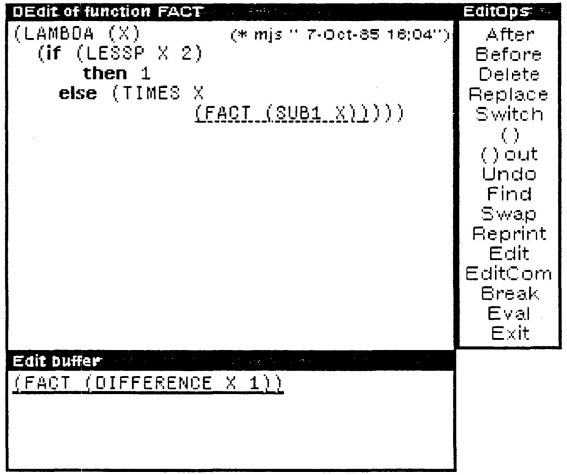
Luna 2018


getPrice1
getPrice "BTC"
zipWith1
. zipWith (/) .
Real -> Real -> Real
Stream Real
Stream Real
getPrice2
getPrice "ETH"

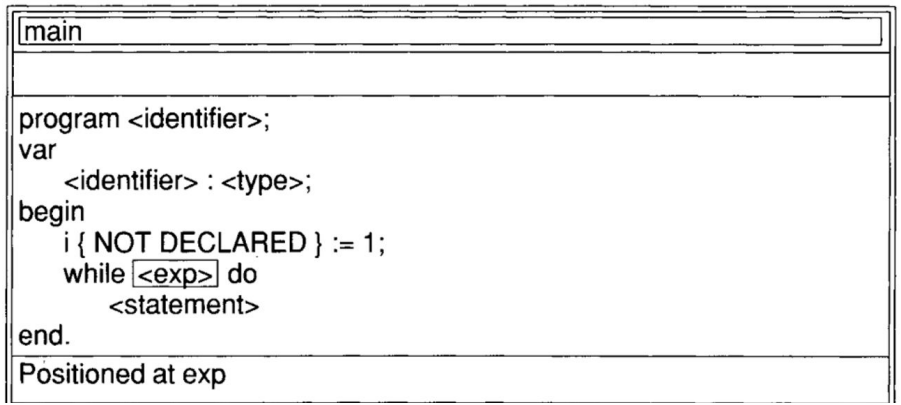

each1
. each formatVehicle
take1
. take 6
geojsonfeaturecollection1
GeoJSONFeatureCollection
[GeoJSONFeature..]
List<194>
self
Int
arg0
78
List<6>
[GeoJSONFeature..]
Skawina
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery Â© Mapbox


text1
"XPR"
XPR
XPR
getPrice1
getPrice
Runtime error: There is no data for the symbol XPR .
lookupReal1
. lookupReal "USD"
Runtime error: There is no data for the symbol XPR .
text2
"ETH"
ETH
getPrice11
getPrice
{"USD": 1085.99}
lookupReal2
. lookupReal "USD"
1085.99

Early structure editors



Interlisp DEdit 1973



Cornell Program Synthesizer/
Synthesizer Generator 1981

Pecan 1984

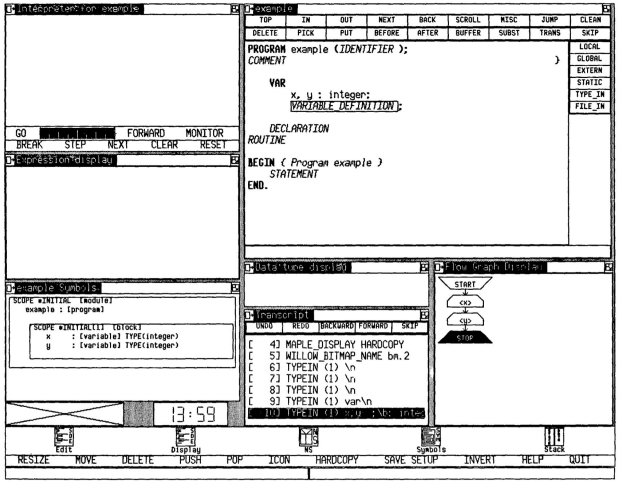


Figure 2: PECAN Display Showing Editing

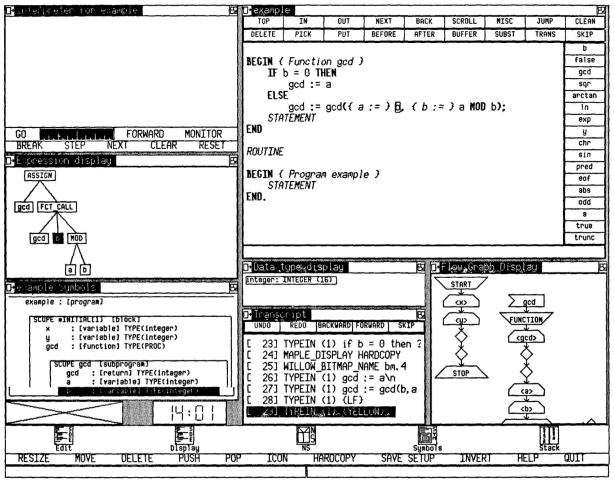


Figure 3: PECAN Display Showing Semantic Views

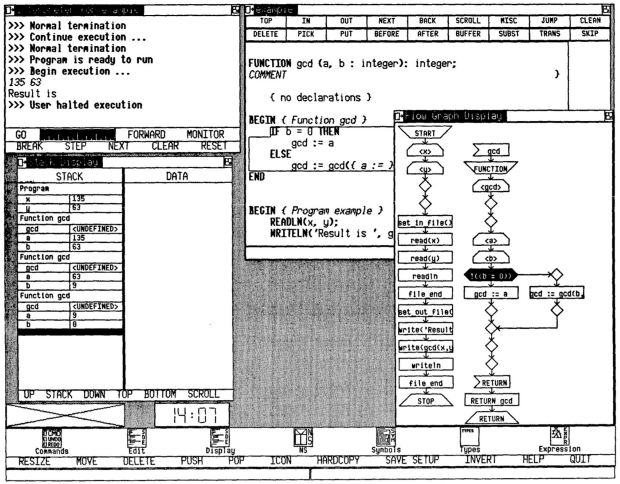
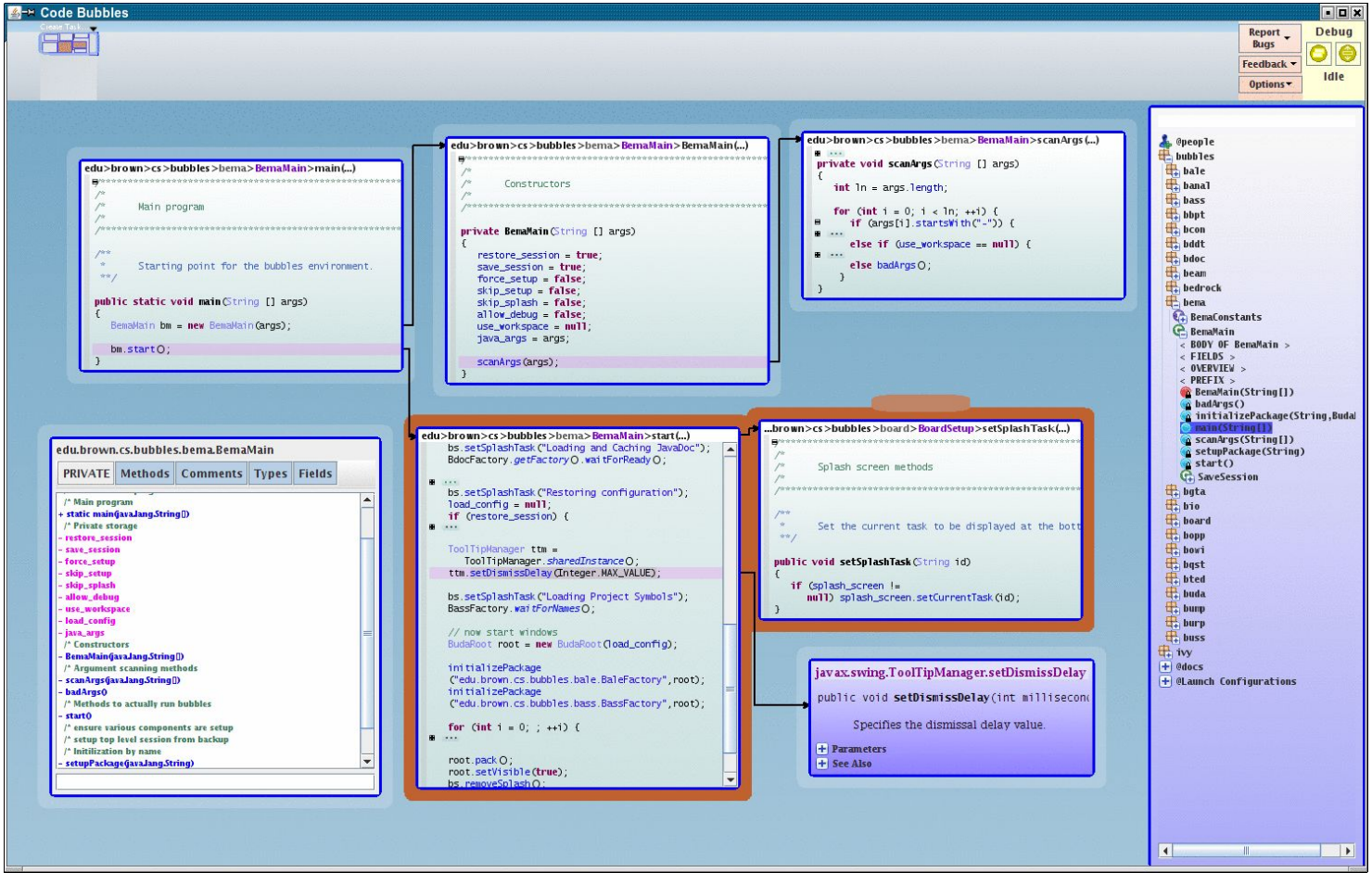


Figure 4: PECAN Display Showing Execution

Code Bubbles 2010



Patchworks 2014

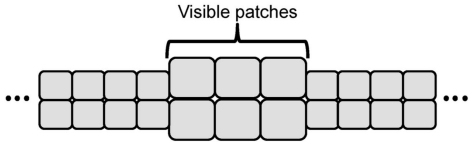


Figure 5. The conceptual ribbon of patches.

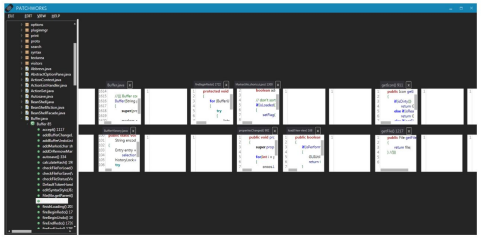


Figure 6. Patchworks' ribbon view, which provides a bird's eye view of the ribbon.

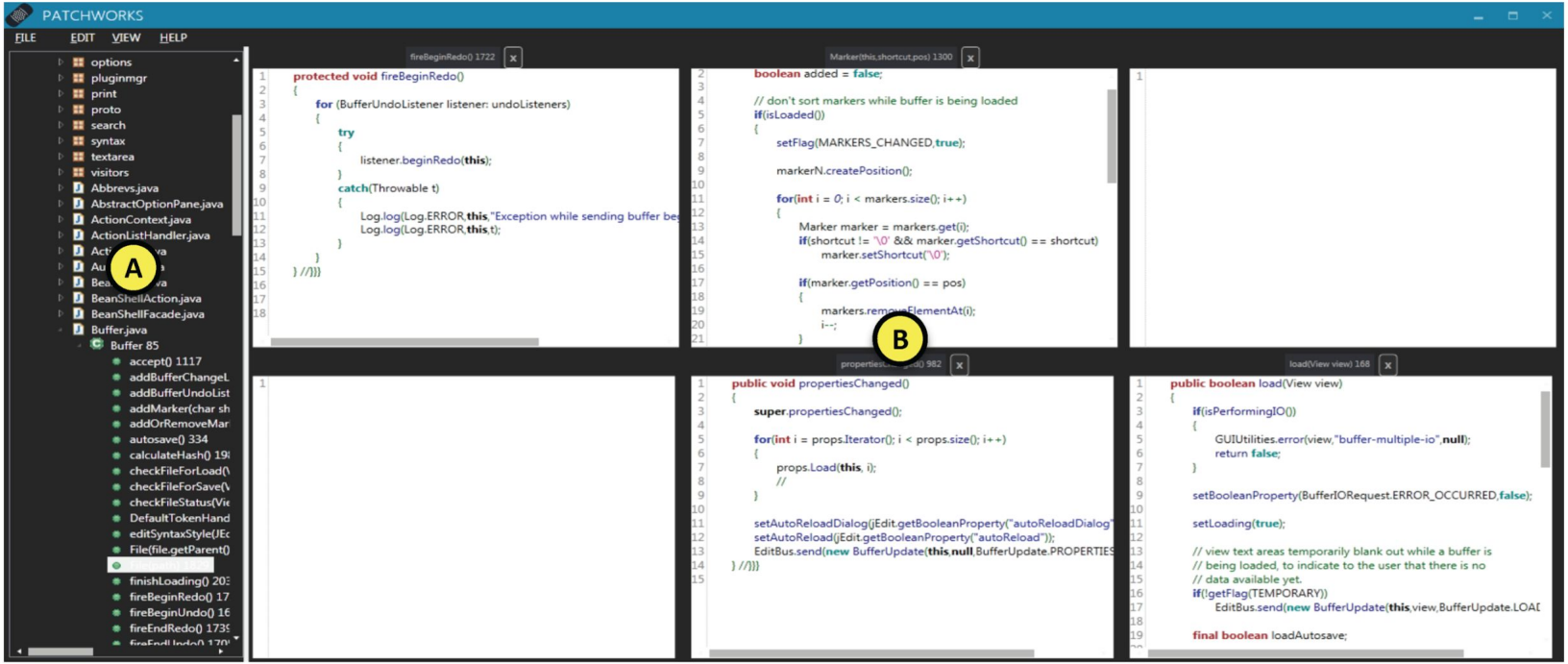
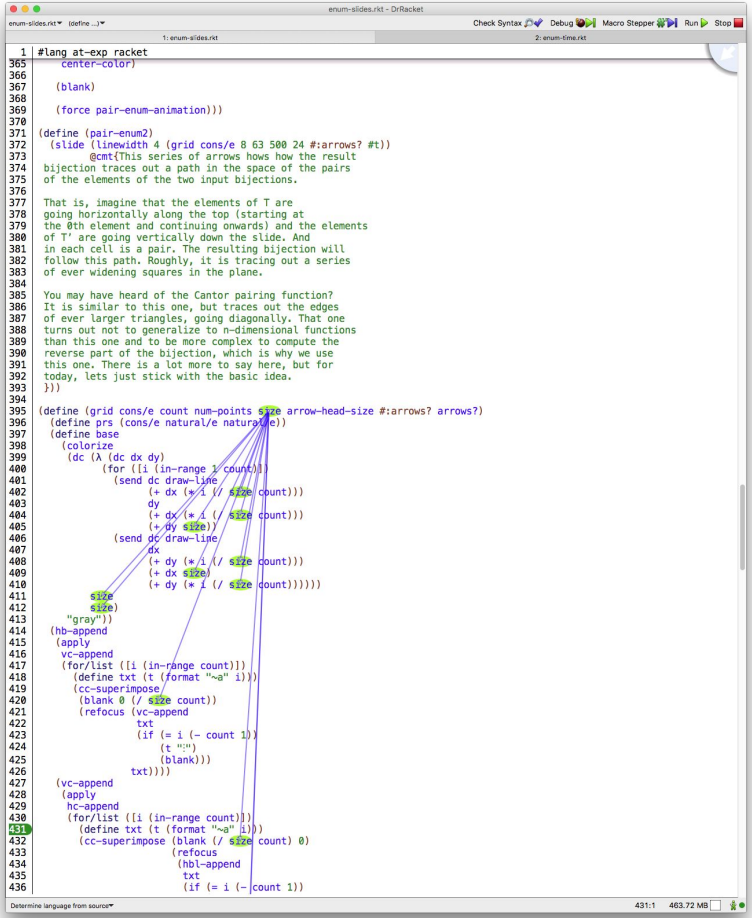
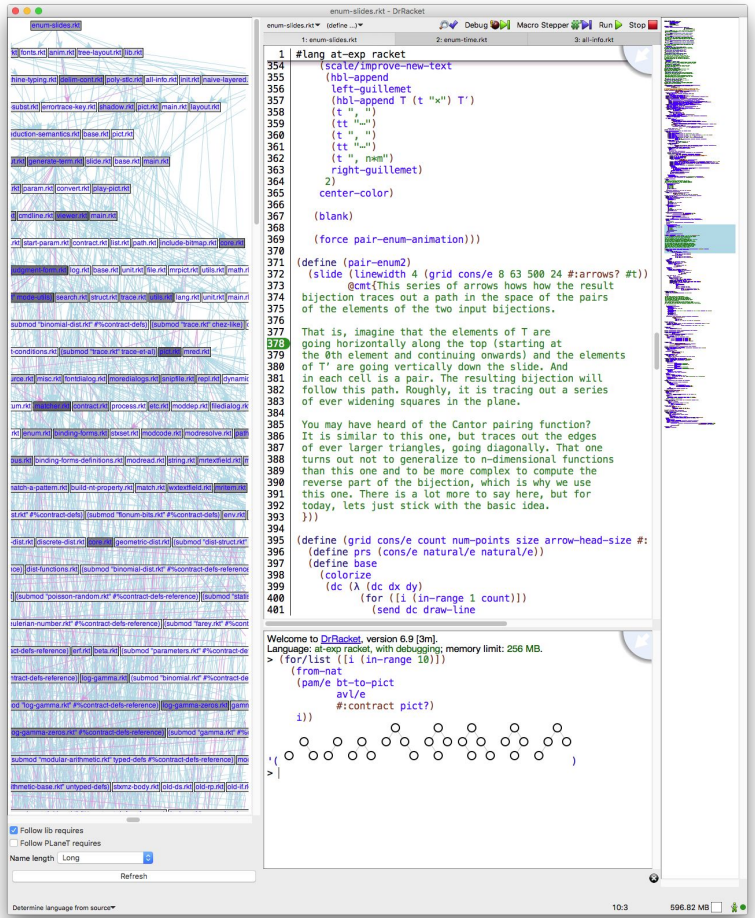


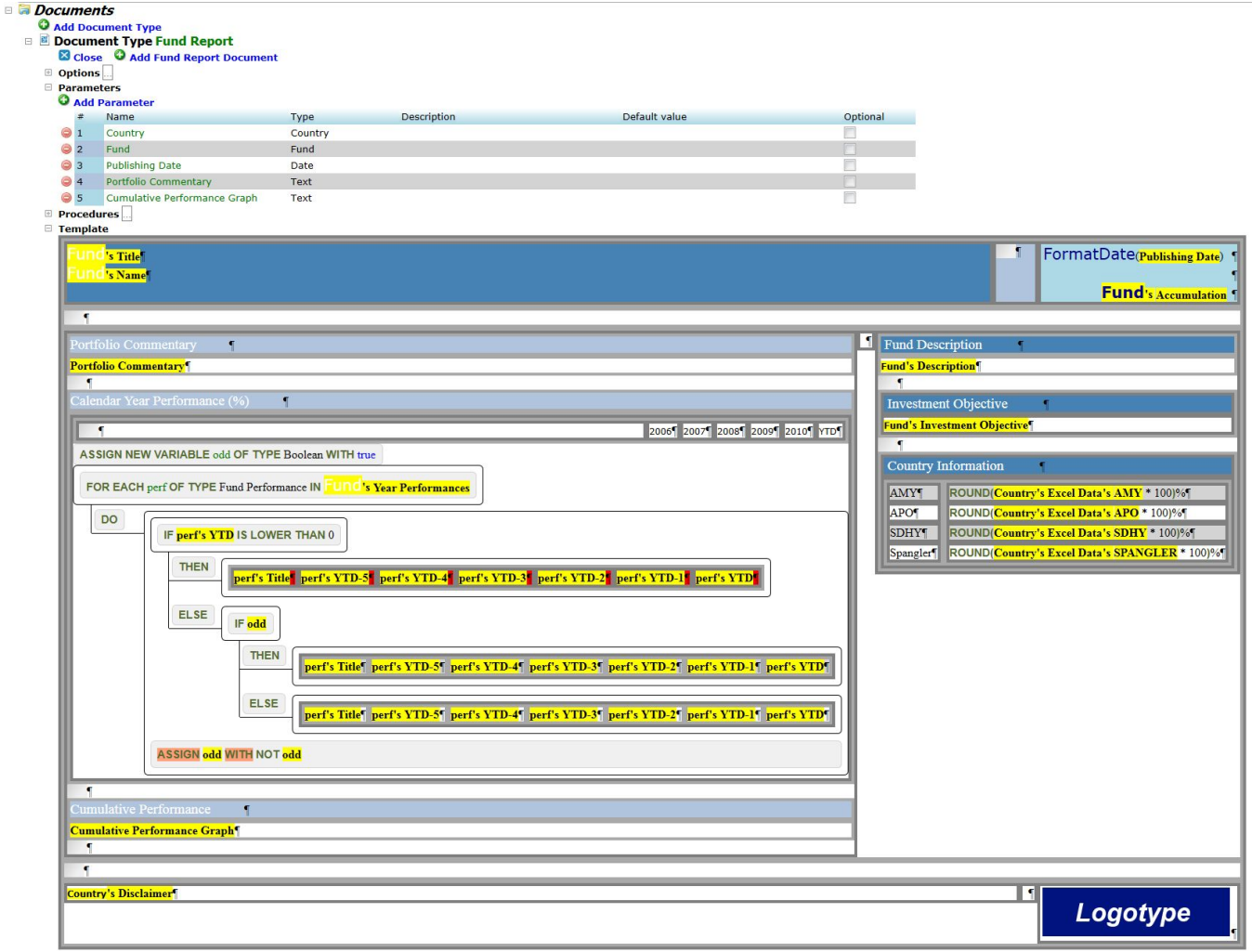
Figure 3. The Patchworks editor, including (A) a package explorer and (B) a 3×2 patch grid. Four of the patches contain code fragments and two are empty.

DrRacket 1997





Intentional Software 2002



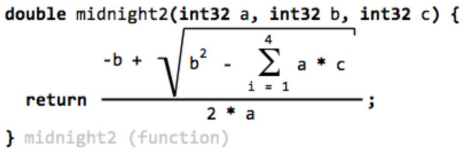
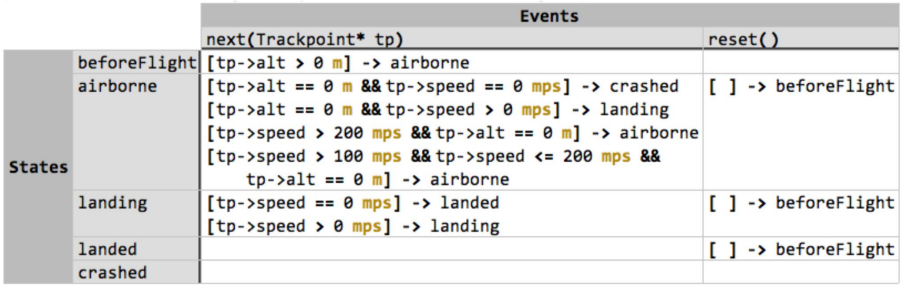


Fig. 2. Mathematical symbols used in C expressions embedded into C functions.



		Events	
		next(Trackpoint* tp)	reset()
States	beforeFlight	[tp->alt > 0 m] -> airborne	
	airborne	[tp->alt == 0 m && tp->speed == 0 mps] -> crashed [tp->alt == 0 m && tp->speed > 0 mps] -> landing [tp->speed > 200 mps && tp->alt == 0 m] -> airborne [tp->speed > 100 mps && tp->speed <= 200 mps && tp->alt == 0 m] -> airborne	[ ] -> beforeFlight
	landing	[tp->speed == 0 mps] -> landed [tp->speed > 0 mps] -> landing	[ ] -> beforeFlight
	landed		[ ] -> beforeFlight
	crashed		

Fig. 4. A state machine represented as a table; also shows nested headers.

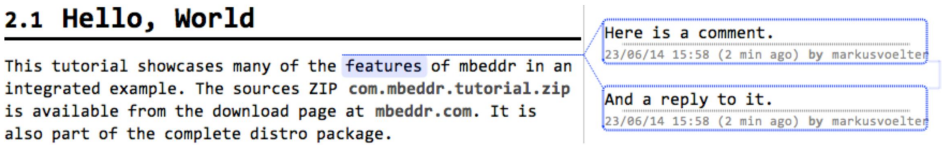


Fig. 6. Margin cells used to support Word-like comments in MPS; other contents can be projected into the margin as well.

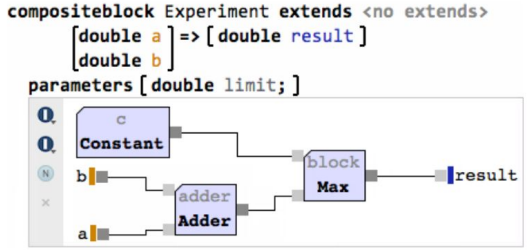


Fig. 7. A graphical editor embedded in a regular text editor.

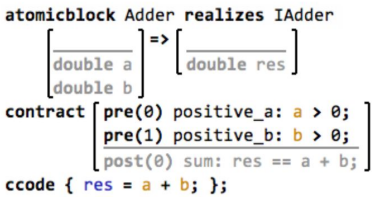


Fig. 9. A querylist is used to project the ports and contracts inherited from the interface realized by this block (in grey). New nodes ports or contracts can be entered above the grey lines.

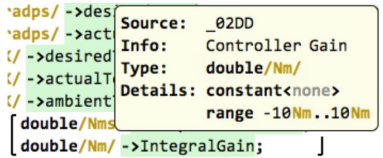


Fig. 10. This tooltip shows the definition of the quantity referenced via the -> notation: it shows its type and various additional details. The tooltip uses the querylist to project derived nodes.

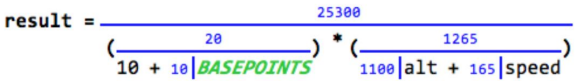
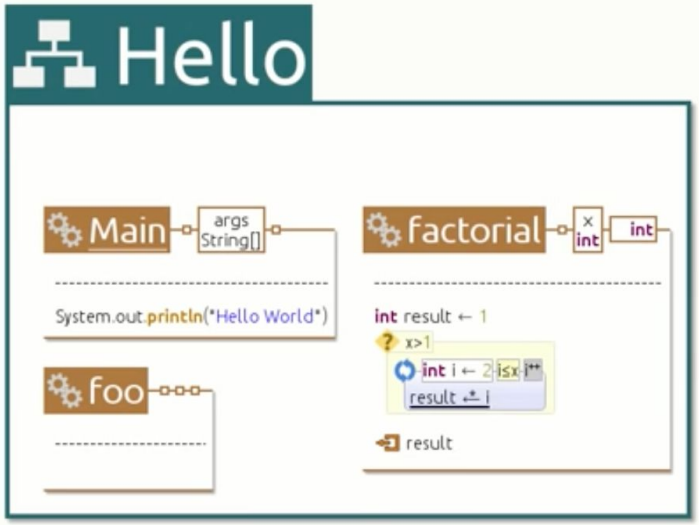
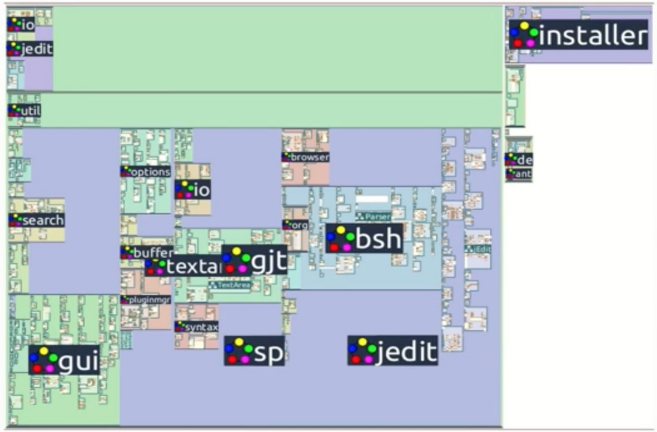
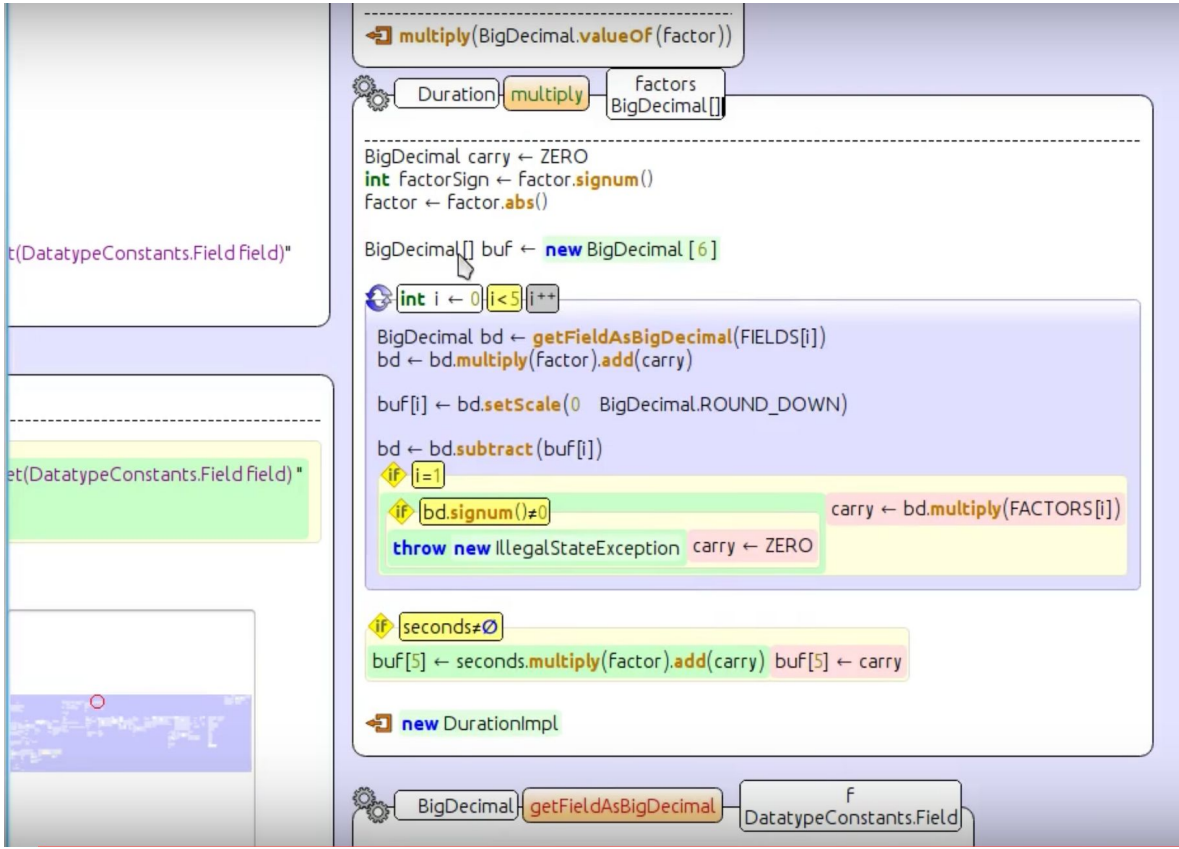


Fig. 11. This expression debugger renders the values of all subexpression over or to the left of the expression itself. The original expression (without the debug info) is `(10 + BASEPOINTS) * (alt + speed)`.

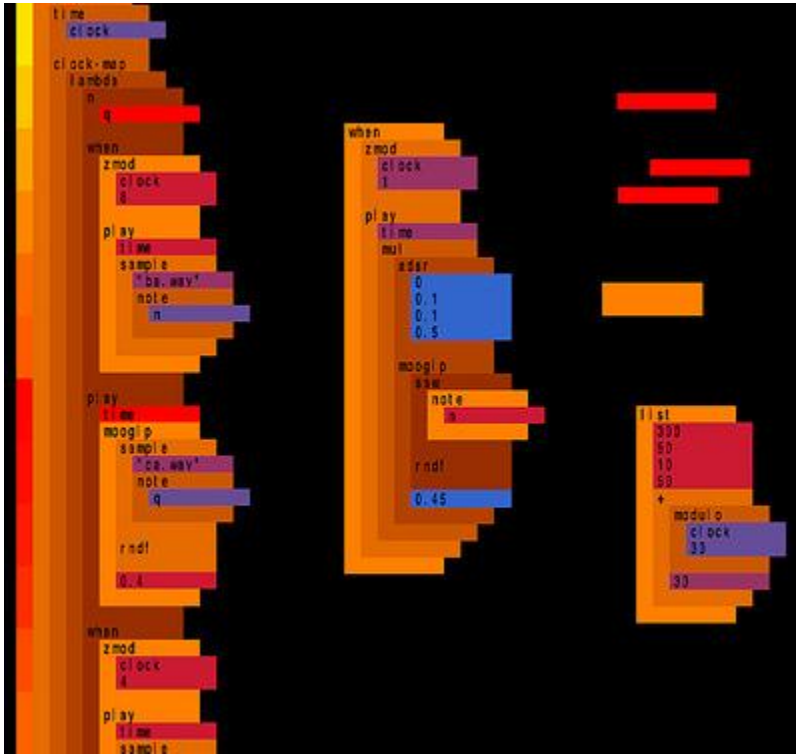
Envision 2016





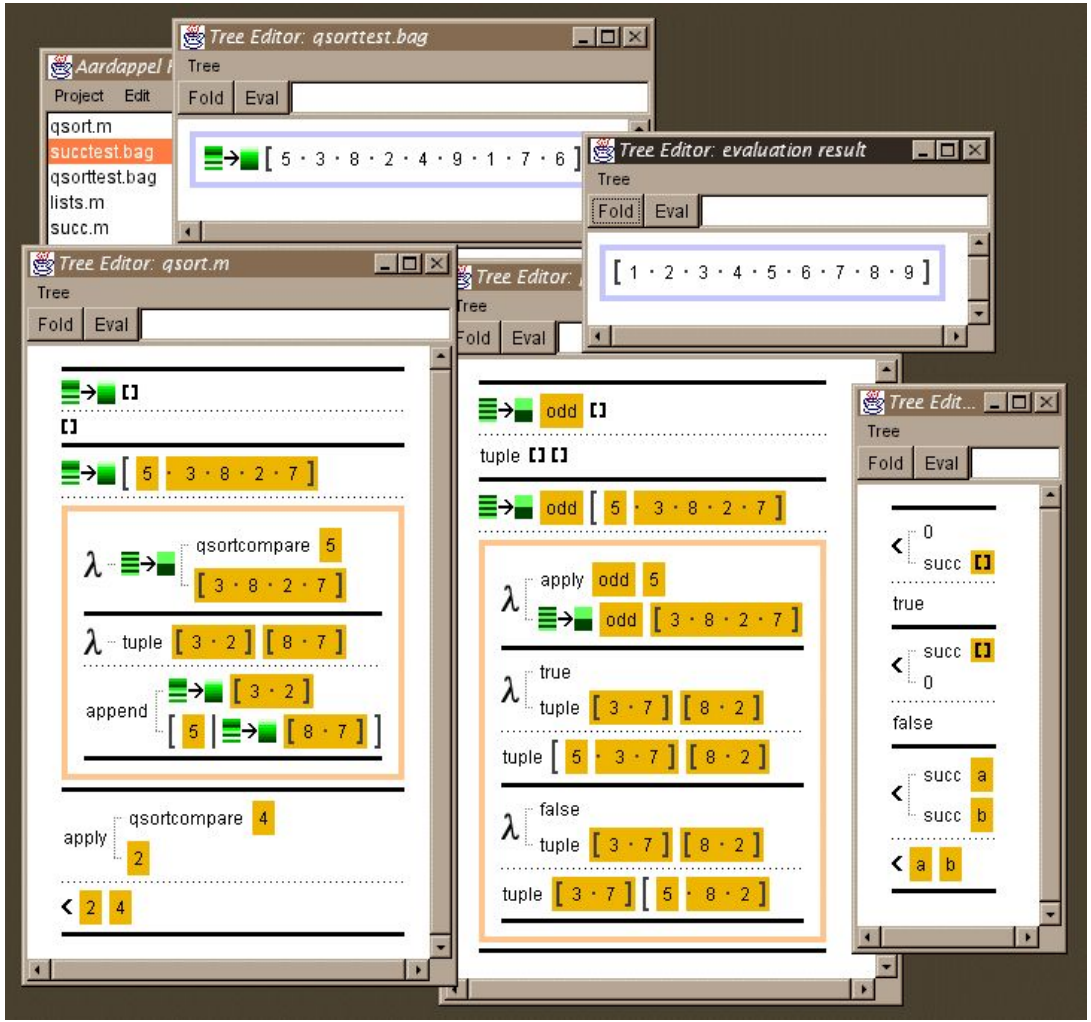


Scheme Bricks 2008



Aardappel

2000



Lamdu 2015

```
⋙ sum map 1 .. 100000
        step 4
  mapping x → 
```

Num

x
length
negate
runMut
_ - _
...
Num

```
⋙ sum map 1 .. 100000
        step 4
```

NonEmpty head 1 tail Fn

```
  mapping x → 8 / x / (x + 2)
```

1 | 8 | 3 | 2.6666666666666665

NonEmpty head 2.6666666666666665 tail Fn

3.141572653589808

```
partition array cond =
  Array a31   a31 → Bool
newMutArray « Empty ;
    mutTrue →
MutArray BI:val a31
         s i67
newMutArray « Empty ;
    mutFalse →
MutArray BI:val a31
         s i67
sequence_ map fromArray array ;
    mapping x → appendMutArray cond x »:
      a31                 True ◗ mutTrue
                          False ◗ mutFalse
                        val x
◗ freezeMutArray mutTrue ; trues → freezeMutArray mutFalse ; falses → return onTrue trues
                  Array a31                      Array a31      onFalse falses
```

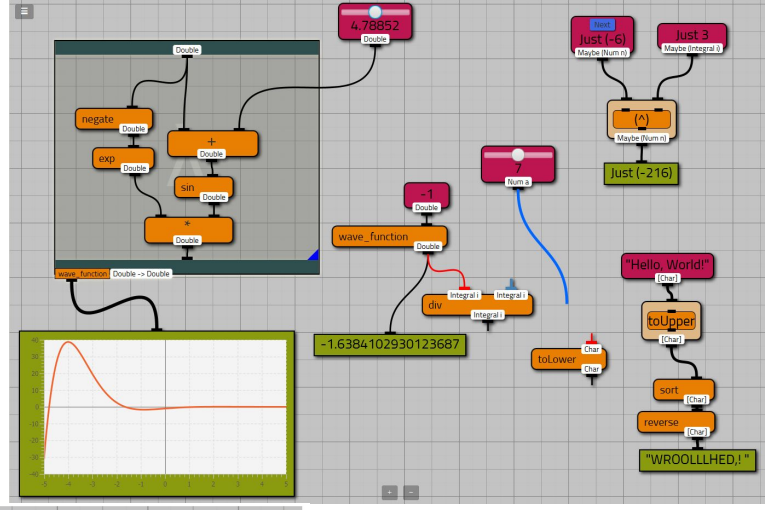
```
(x == 0) »:
  True ◗ "Hello„
  False ◗ 5
          Num
  Text
```

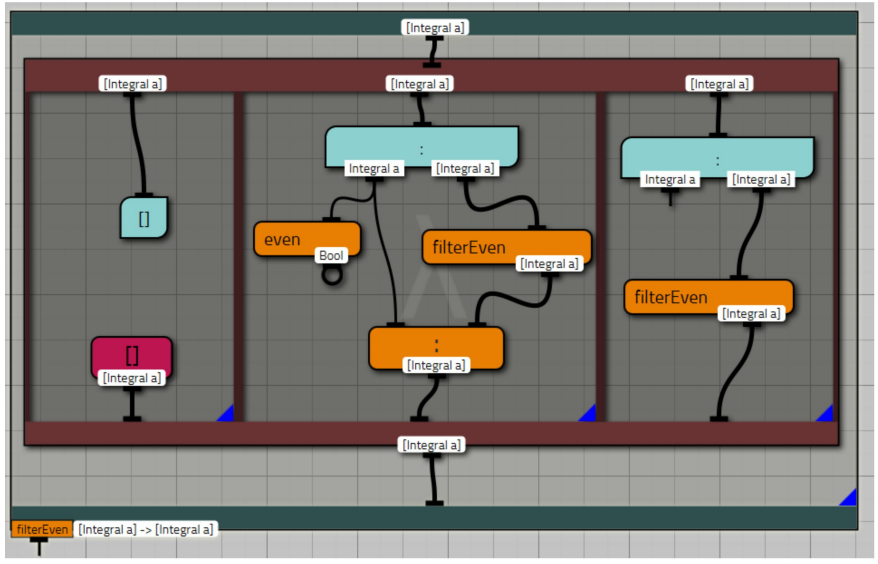
```
(x == 0) »:
  True ◗ "Hello„
         Text
  Num
  False ◗ 5
```

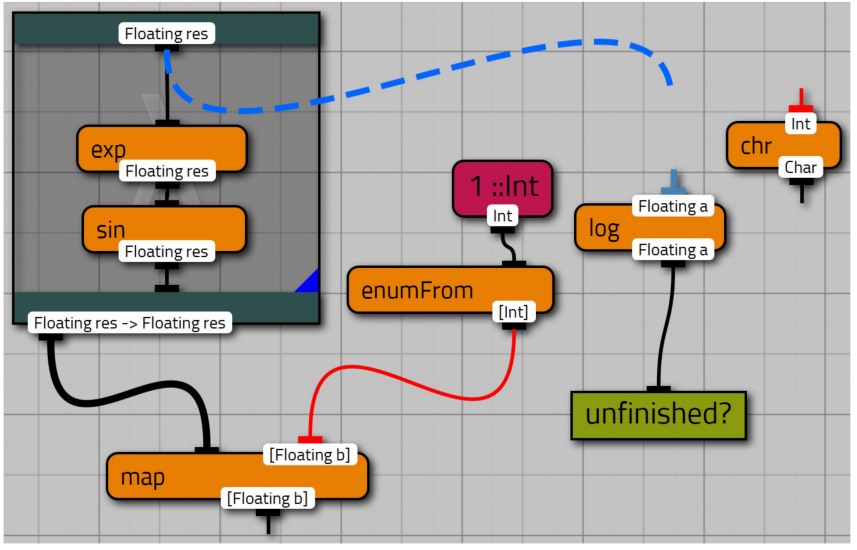
Blame assignment

Viskell

2016







Roly Perera

2012

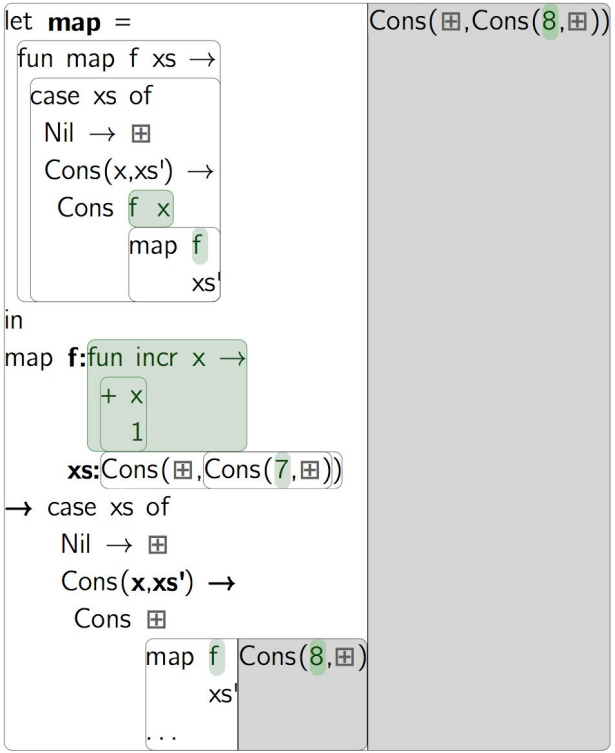
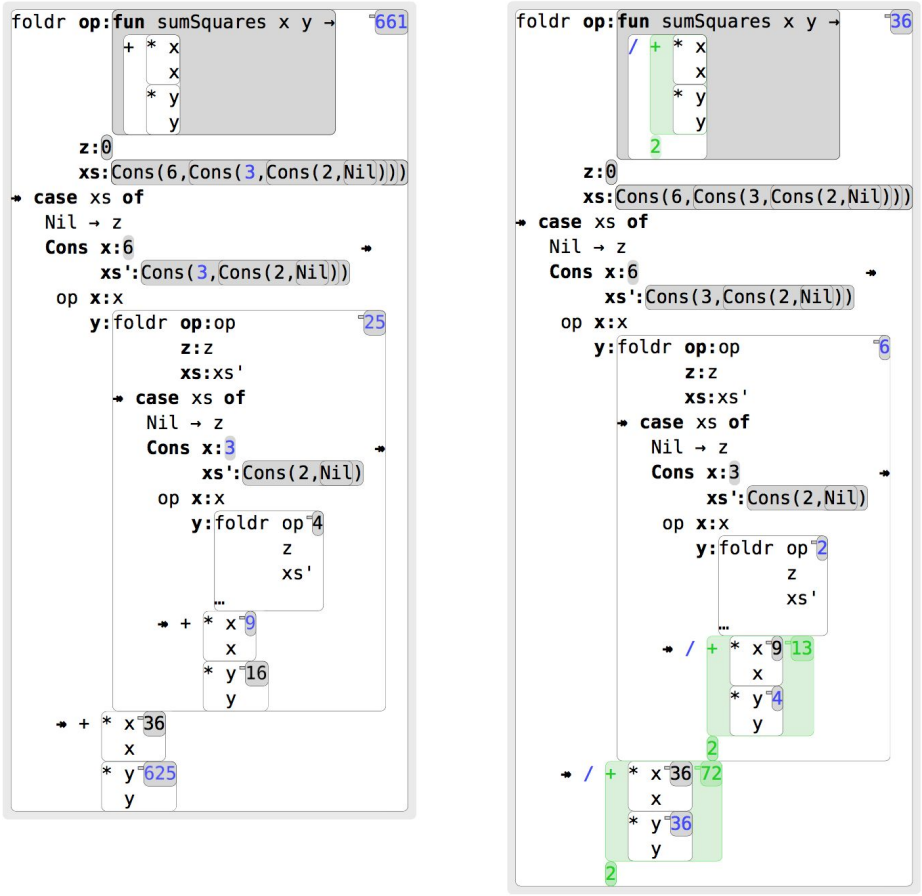


Figure 2. Slice of `map incr [6,7,2]`

Figure 3. Outermost merge phase of mergesort.



a. Expand nested `op` call

b. Change `sumSquares`

Figure 2.4 Structural editing with delta-highlighting

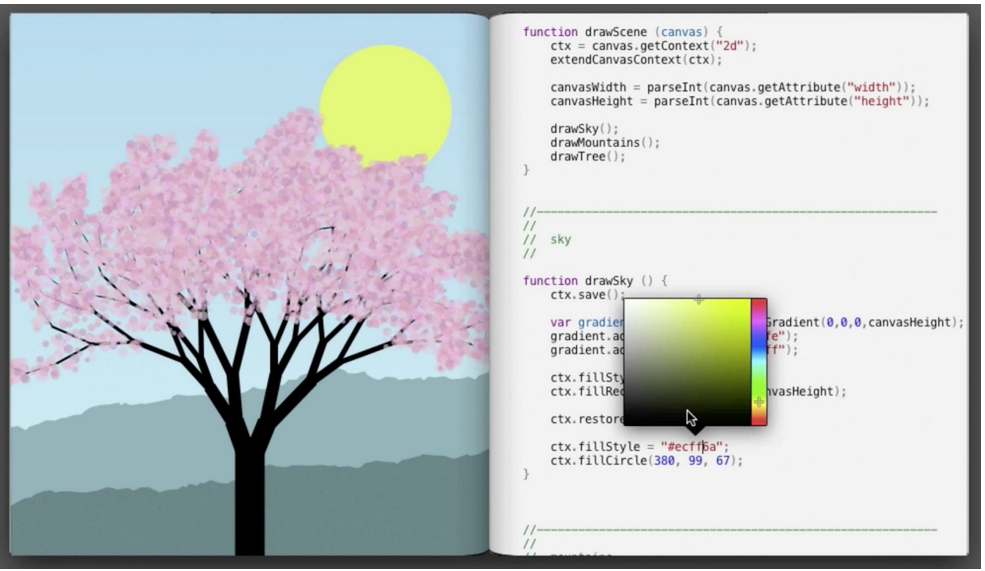
(a) Key 3 is not found

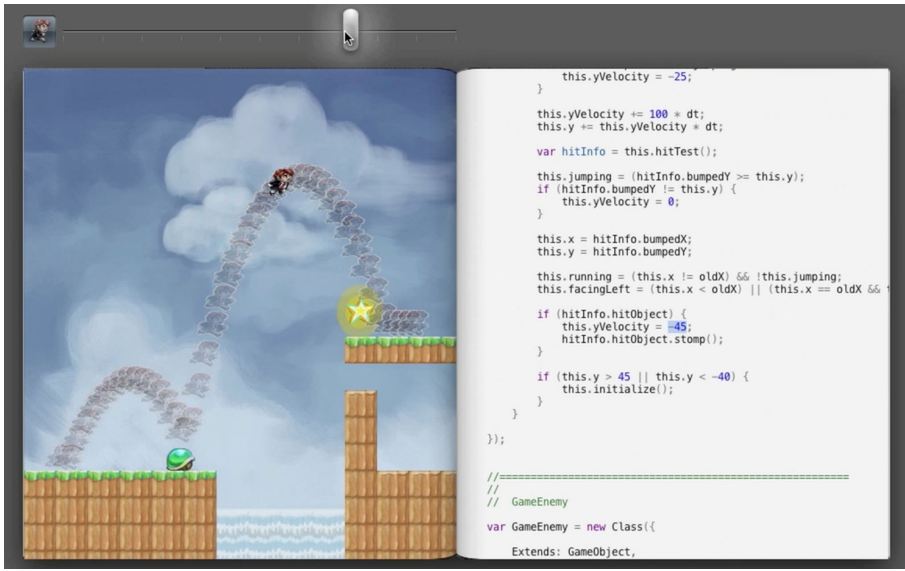
(b) Debug into test case

(c) Accept delta, then re-expand unused nodes

Figure 2.16 Finding and analysing bug in `lookup`

Bret Victor: Inventing on Principle 2012

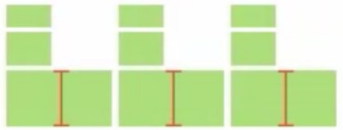




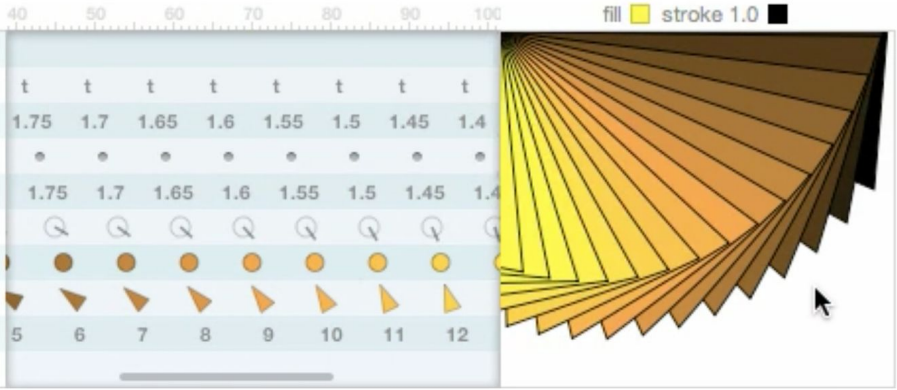
Bret Victor: Learnable Programming 2012

```
fill(161, 219, 114);
for (var x = 40; x < 150; x += 50) {
    rect(x, 33, 20, 10);
    rect(x, 45, 20, 15);
    rect(x, 62, 47, 25);
}
```

height of the rectangle



```
var i = 0;
while (i < 20) {
    var scaleFactor = 1 + (20 - i)/20;
    resetMatrix();
    scale(scaleFactor);
    rotate(i * 6);
    fill(i * 30, i * 18, 0);
    triangle(0,0, 100,-20, 95,40);
    i += 1;
}
```



Abstract

var

function

Flow

if

for

while

```
var x = 80;
rect(x, 80, 40, 25);
triangle(x, 80, 20, 60, 120, 80);
```

Abstract

var

function

Flow

if

for

while

```
var x = 80;
rect(x, 80, 40, 25);
triangle(x, 80, 20 + x, 60, 120, 80);
```

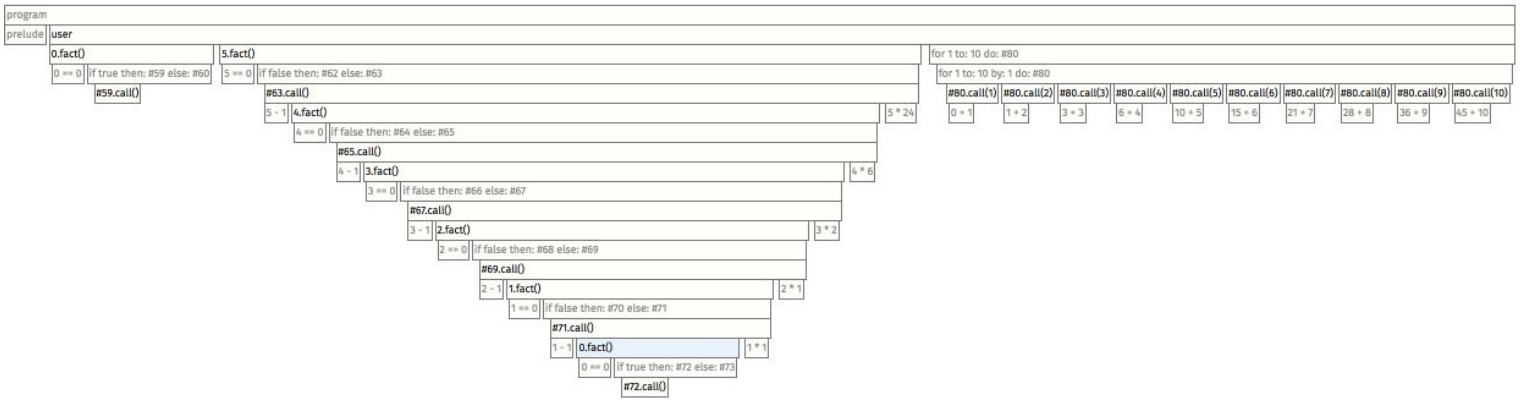
```
20 + x
180 - x
1.25 * x
8000 / x
```

Seymour 2017

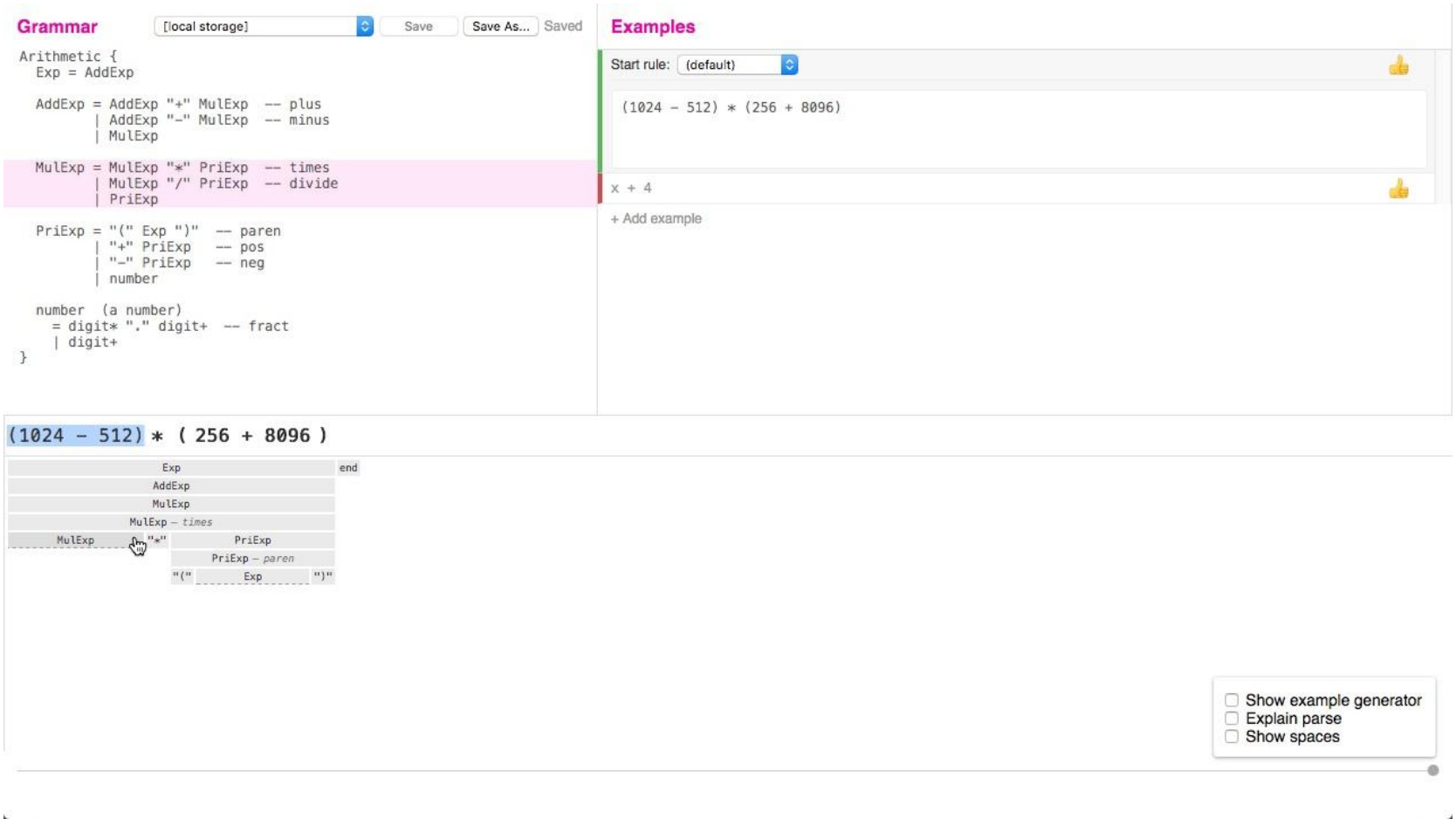
```
1  def Number.fact() {
2    if this == 0 then: {
3      return 1;
4    } else: {
5      var that = (this - 1).fact();
6      return this * that;
7    };
8  }
9
10 var f0 = 0.fact();
11 var f5 = 5.fact();
12
13 var sum = 0;
14 for 1 to: 10 do: {x |
15   sum = sum + x;
16 };
```

```
this = 0

return 1

 f0 = 1
 f5 = 120

sum = 0
```

x = 1	x = 2	x = 3	x = 4	x = 5	x = 6	x = 7	x = 8	x = 9	x = 10
sum = 1	sum = 3	sum = 6	sum = 10	sum = 15	sum = 21	sum = 28	sum = 36	sum = 45	sum = 55



Ohm Editor 2016



Subtext 1 2005

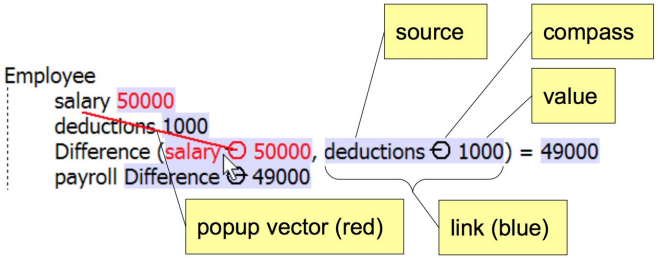


Figure 3. Linking

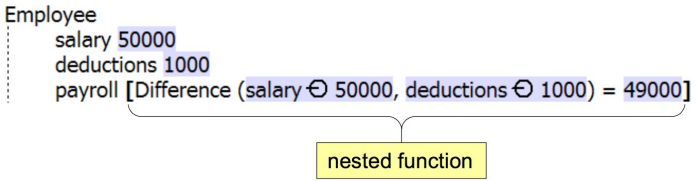


Figure 6. Nesting

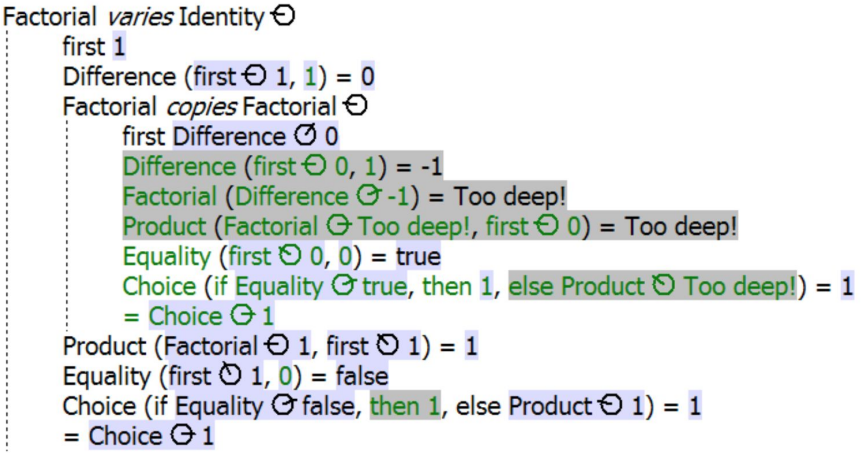
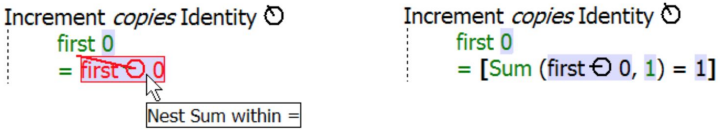


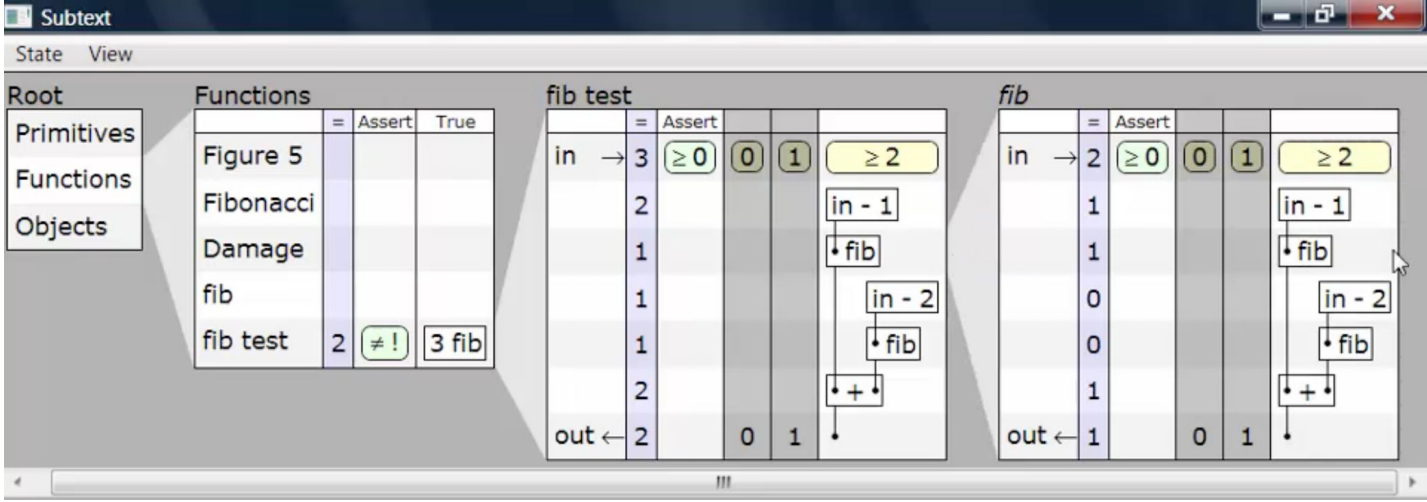
Figure 5. Factorial recursion

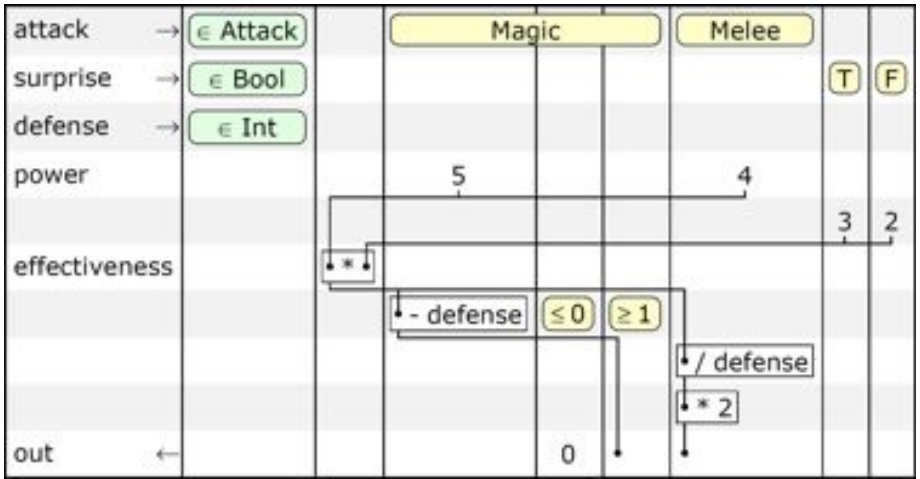


a) dragging to splice b) nested splice

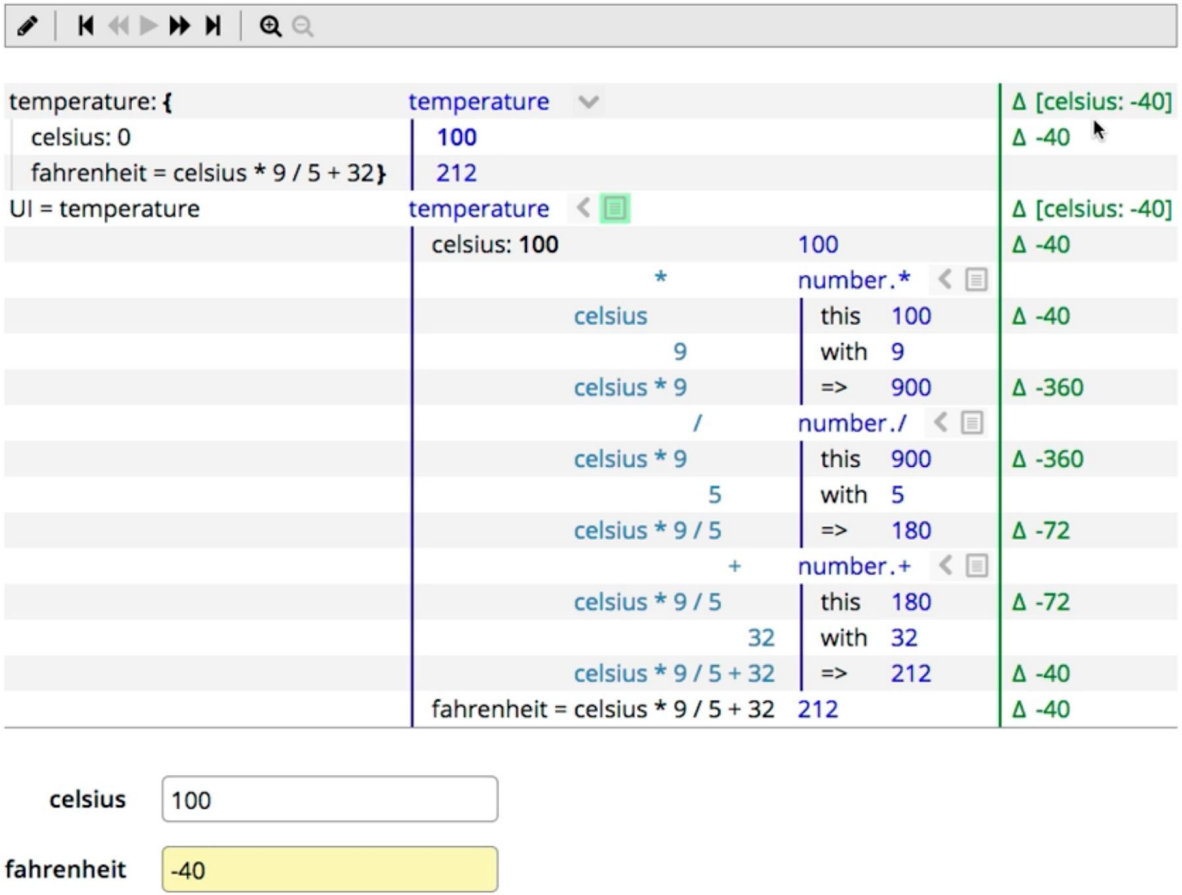
Figure 7. Splicing

[Subtext 2](#)
([Schematic Tables](#))
2007

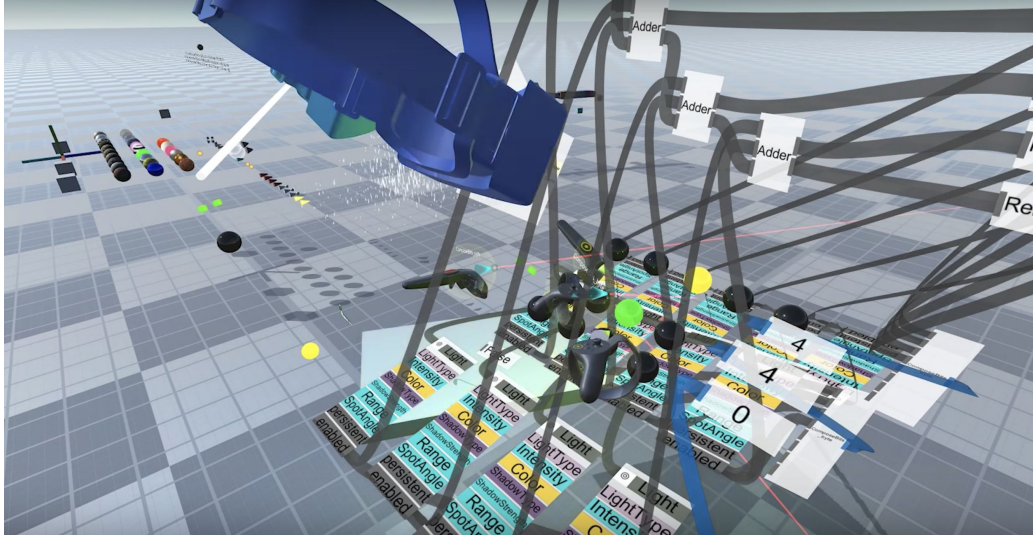


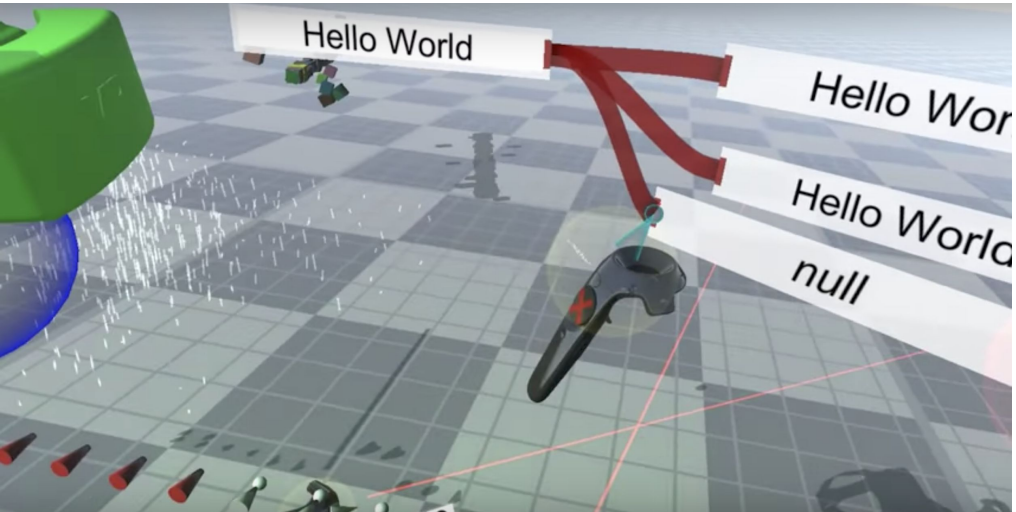


Subtext 5 (Two-way Dataflow) 2014



LogiX 2017


Adder
Adder
Adder
4
4
0


Hello World
Hello Wor
Hello World
null