UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI INFORMATICA

Visual programming languages

D. Malchiodi

Visual programming

Myers (1986): «*Visual Programming* (VP) refers to any system that allows the user to specify a program in a two (or more) dimensional fashion. Conventional textual languages are not considered two dimensional since the compiler or interpreter processes it as a long, one-dimensional stream. Visual Programming includes conventional flow charts and graphical programming languages. It does not include systems that use conventional (linear) programming languages to define pictures. This eliminates most graphics editors, like Sketchpad [Sutherland 63].»

Visual programming

Burnett (1999): «*Visual programming* is programming in which more than one dimension is used to convey semantics. Examples of such additional dimensions are the use of multidimensional objects, the use of spatial relationships, or the use of the time dimension to specify “before-after” semantic relationships.»

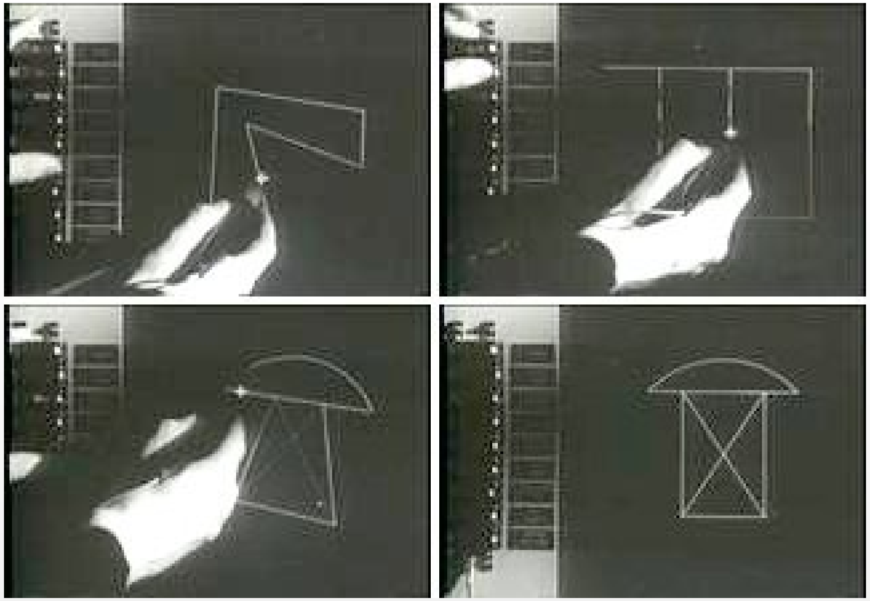
Visual programming

Wikipedia (2008): «A *visual programming language* (VPL) is any programming language that lets users **specify programs by manipulating program elements graphically rather than by specifying them textually**. A VPL allows programming with visual expressions, spatial arrangements of text and graphic symbols. Most VPLs are based on the idea of “boxes and arrows,” that is, boxes or circles or bubbles, treated as screen objects, connected by arrows, lines or arcs.»

1963: Sketchpad

The first computer system with a GUI, using an X-Y plotter and a light pen to construct 2D graphics

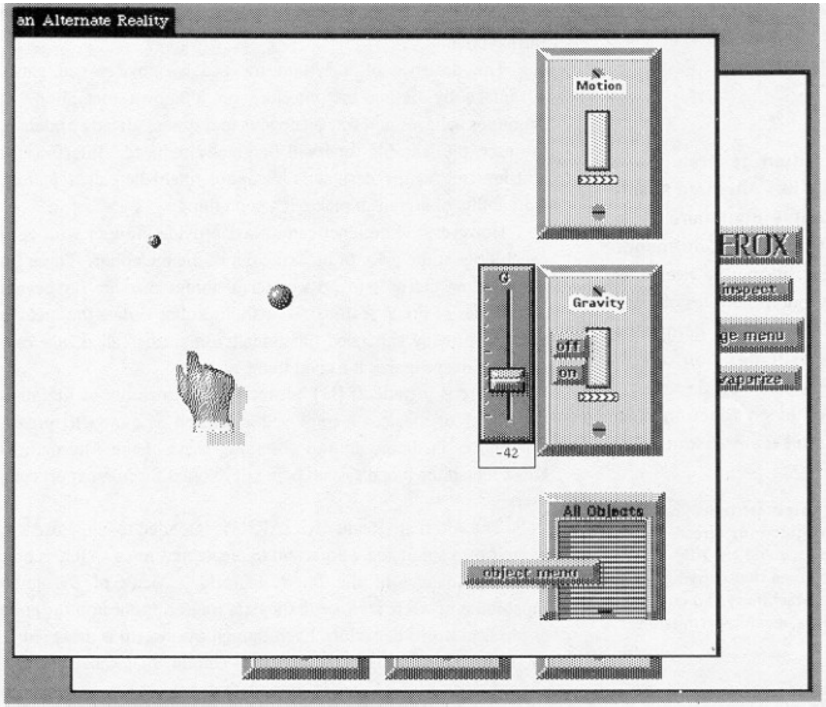
(NB: not a VPL)



Ivan Edward Sutherland, *Sketchpad: A man-machine graphical communication system*, Ph.D. thesis, MIT, January 1963. www.cl.cam.ac.uk/techreports/UCAM-CL-TR-574.pdf

ARK – The Alternate Reality Kit

A 2D environment for creating interactive simulations implemented in Smalltalk-80



Randall B. Smith, *“Experiences with the alternate reality kit: an example of the tension between literalism and magic,” 1987.* http://dx.doi.org/10.1145/30851.30861

Architectural Description Languages

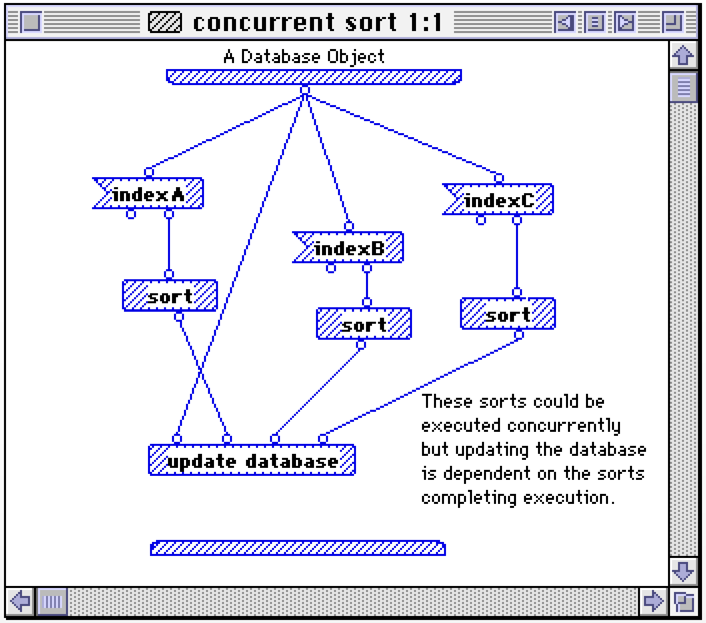
Shaw and Garlan (1996): «Architectural Description Languages (ADLs) model systems in terms of

- **components** that offer services,
- **connectors** that bind services, and
- architectural **constraints** that must be respected.

As a consequence, certain system properties are obtained.»

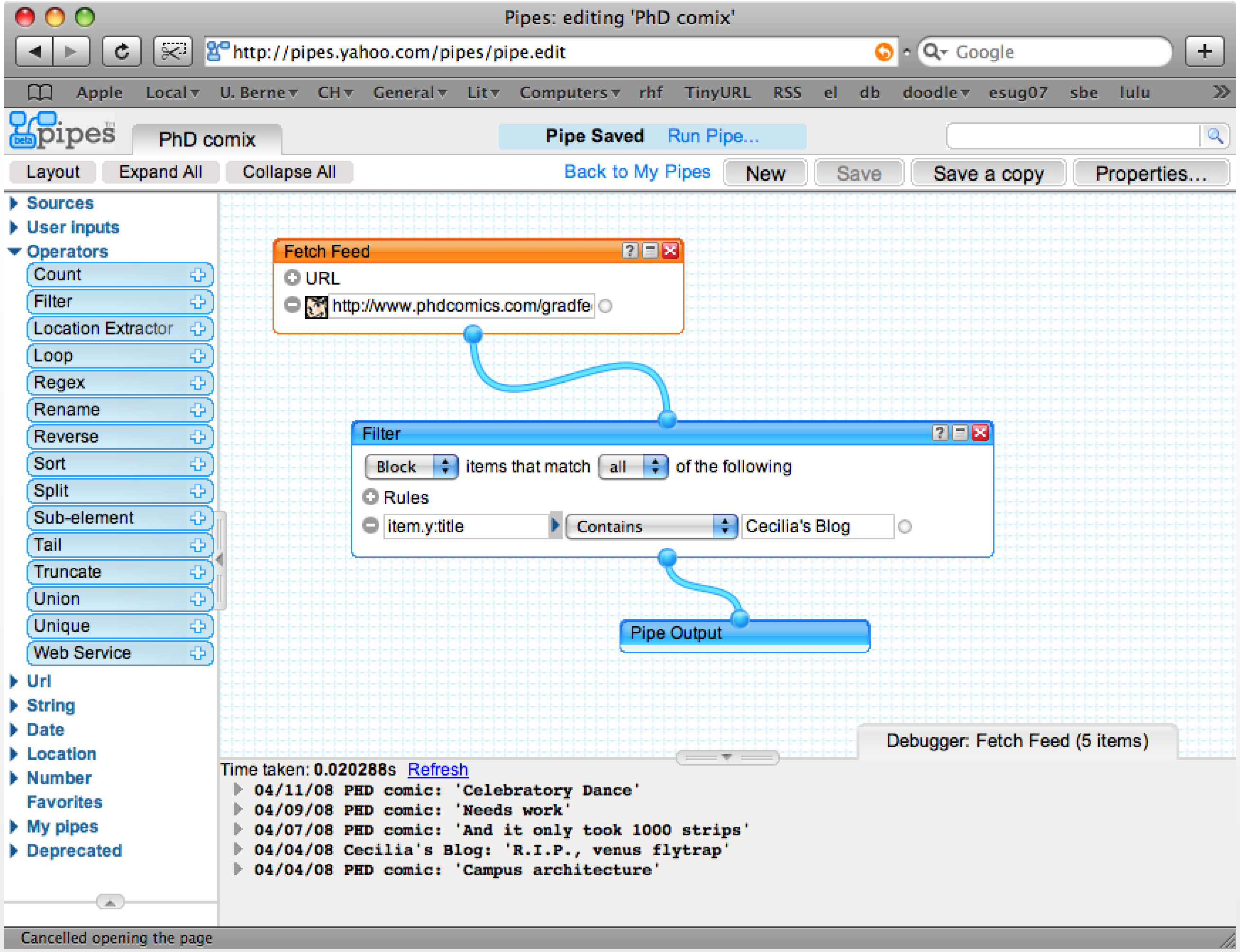
Prograph – dataflow graphs

A visual, object-oriented dataflow language (1990, recently revamped)



These sorts could be executed concurrently but updating the database is dependent on the sorts completing execution.

Yahoo pipes



Fetch Feed
URL: http://www.phdcomics.com/gradfe

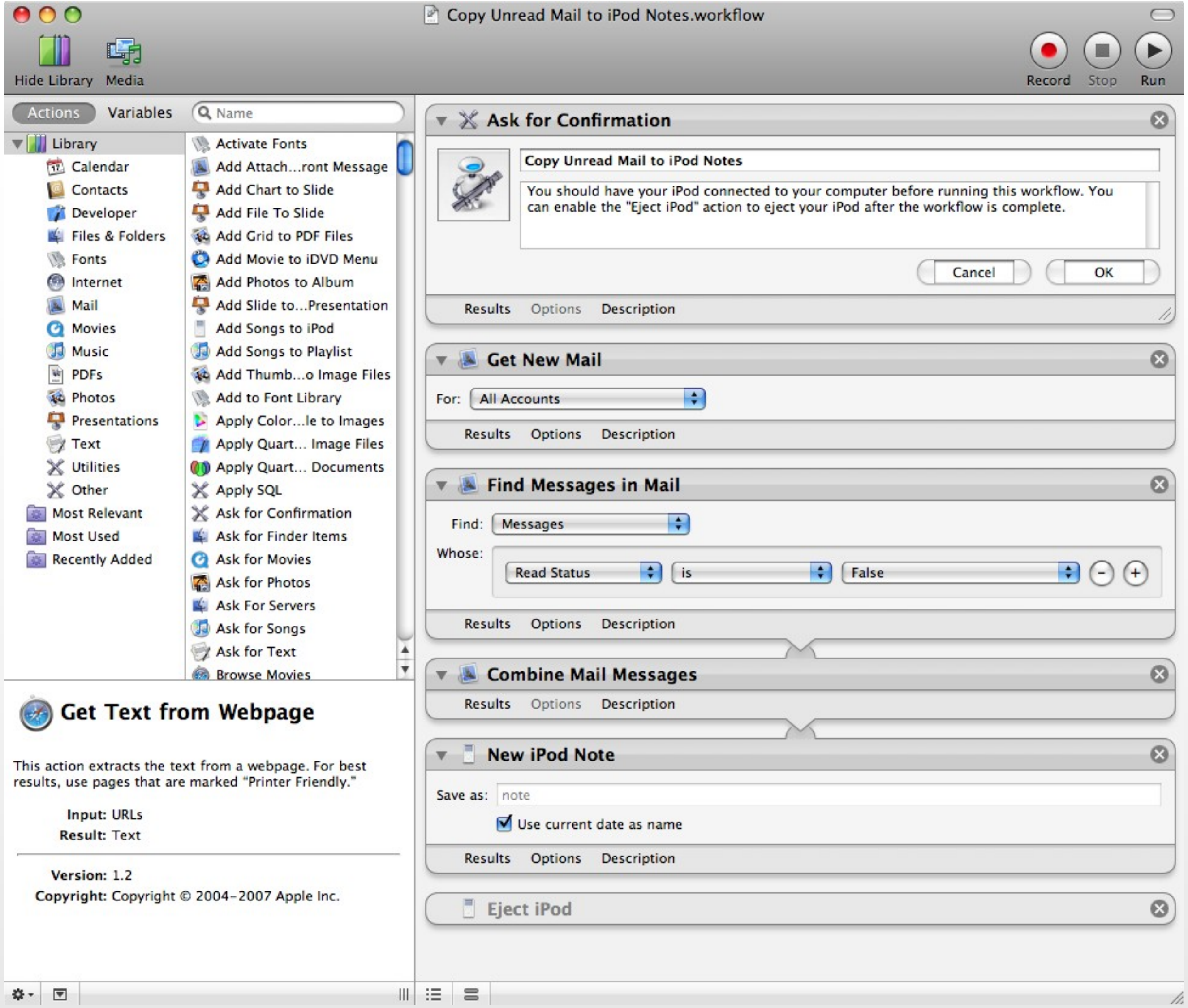
Filter
Block items that match all of the following
Rules
item.y:title Contains Cecilia's Blog

```
Time taken: 0.020288s Refresh
04/11/08 PHD comic: 'Celebratory Dance'
04/09/08 PHD comic: 'Needs work'
04/07/08 PHD comic: 'And it only took 1000 strips'
04/04/08 Cecilia's Blog: 'R.I.P., venus flytrap'
04/04/08 PHD comic: 'Campus architecture'
```

Mashup Internet resources by composing pipes and filters

OSX automator

A built-in tool for scripting common actions as «workflows»



Copy Unread Mail to iPod Notes

You should have your iPod connected to your computer before running this workflow. You can enable the "Eject iPod" action to eject your iPod after the workflow is complete.

For: All Accounts

Find: Messages

Whose: Read Status is False

Save as: note

☑ Use current date as name

Get Text from Webpage

This action extracts the text from a webpage. For best results, use pages that are marked "Printer Friendly."

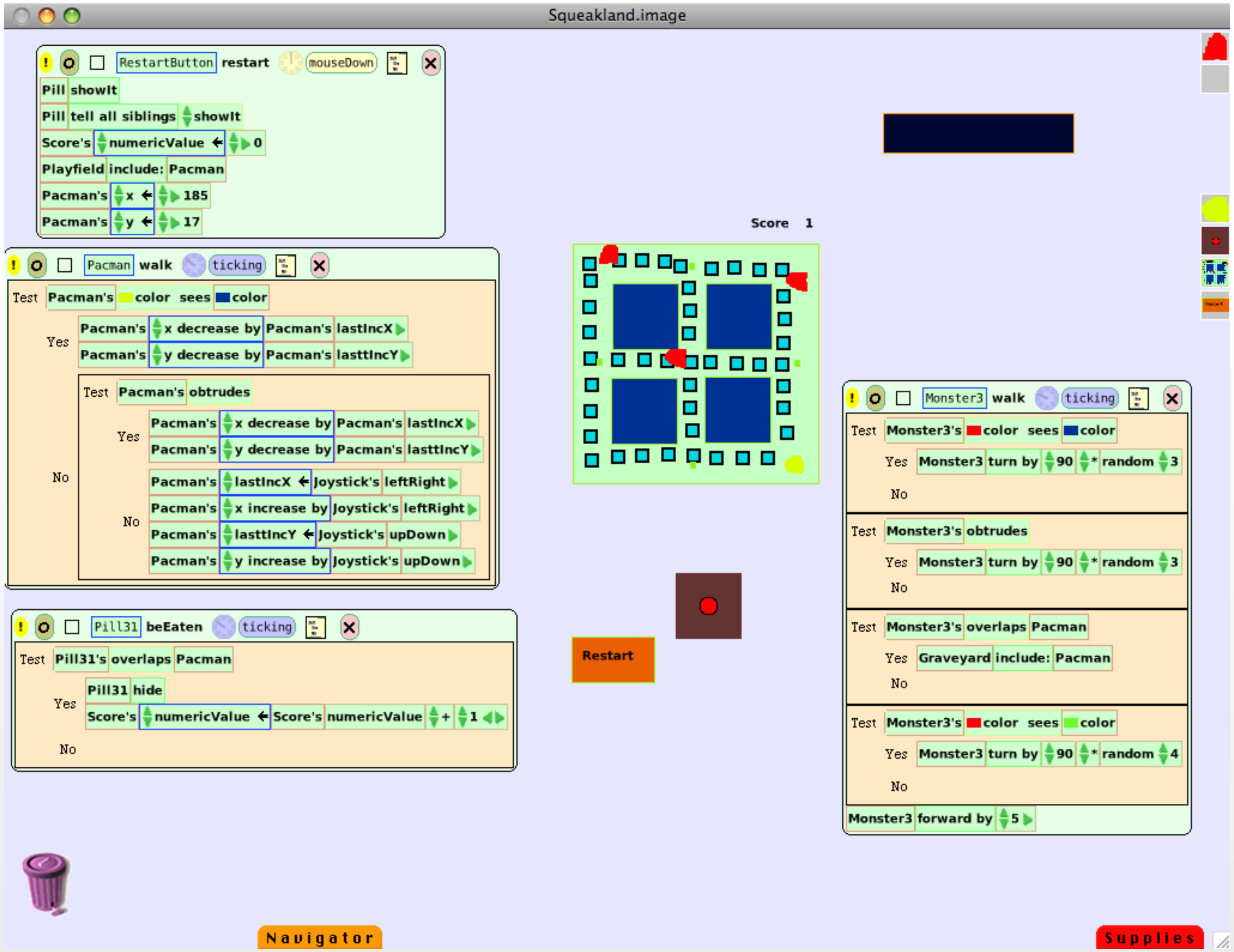
Input: URLs
Result: Text

Version: 1.2
Copyright: Copyright © 2004-2007 Apple Inc.

Etoys (1997)

Programming = composition of “tiles”

Execution = evolution of (predefined) objects



```
RestartButton restart mouseDown
Pill showIt
Pill tell all siblings showIt
Score's numericValue ← 0
Playfield include: Pacman
Pacman's x ← 185
Pacman's y ← 17
```

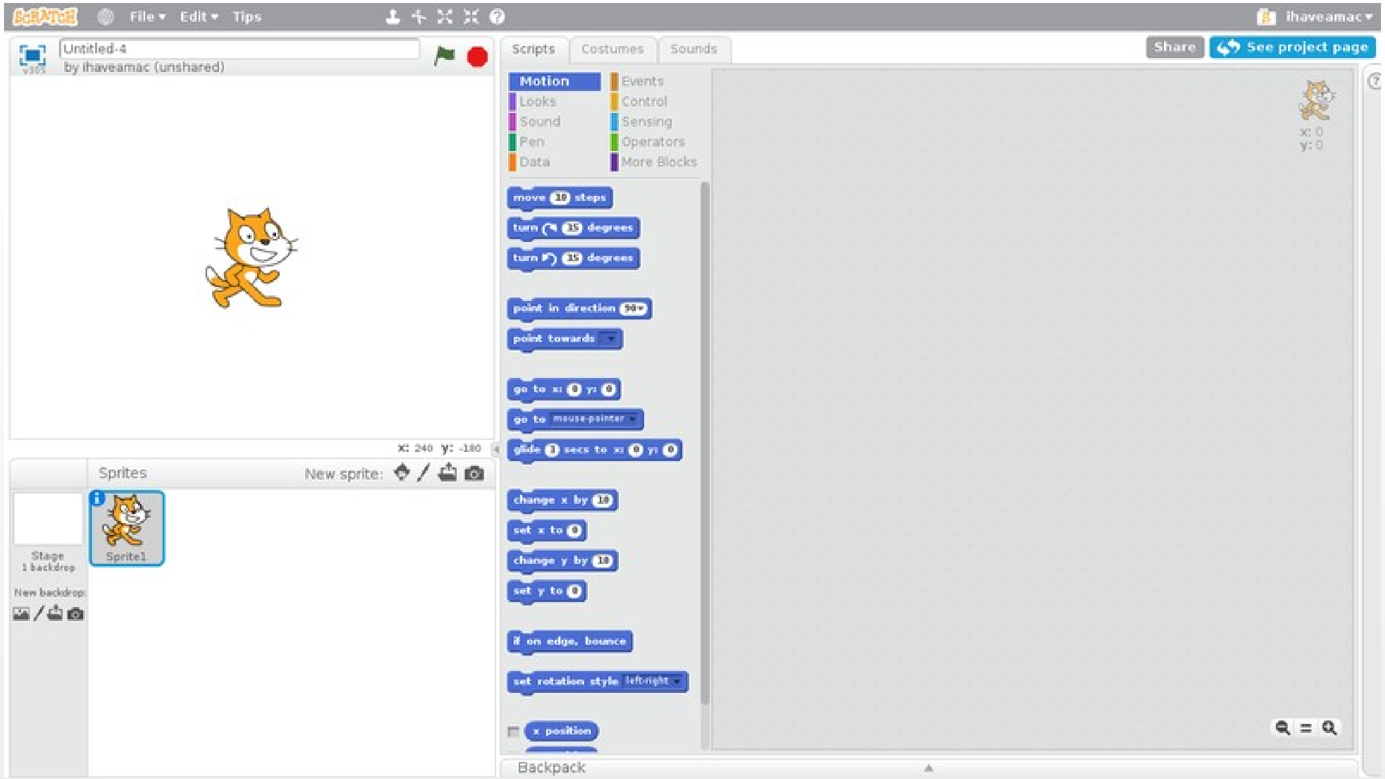
```
Pacman walk ticking
Test Pacman's color sees color
  Yes Pacman's x decrease by Pacman's lastIncX
      Pacman's y decrease by Pacman's lastIncY
  No  Test Pacman's obtrudes
        Yes Pacman's x decrease by Pacman's lastIncX
            Pacman's y decrease by Pacman's lastIncY
        No  Pacman's lastIncX ← Joystick's leftRight
            Pacman's x increase by Joystick's leftRight
            Pacman's lastIncY ← Joystick's upDown
            Pacman's y increase by Joystick's upDown
```

```
Pill31 beEaten ticking
Test Pill31's overlaps Pacman
  Yes Pill31 hide
      Score's numericValue ← Score's numericValue + 1
  No
```

```
Monster3 walk ticking
Test Monster3's color sees color
  Yes Monster3 turn by 90 * random 3
  No
Test Monster3's obtrudes
  Yes Monster3 turn by 90 * random 3
  No
Test Monster3's overlaps Pacman
  Yes Graveyard include: Pacman
  No
Test Monster3's color sees color
  Yes Monster3 turn by 90 * random 4
  No
Monster3 forward by 5
```

Scratch (2007, MIT media lab)

A generic environment with “statement” blocks

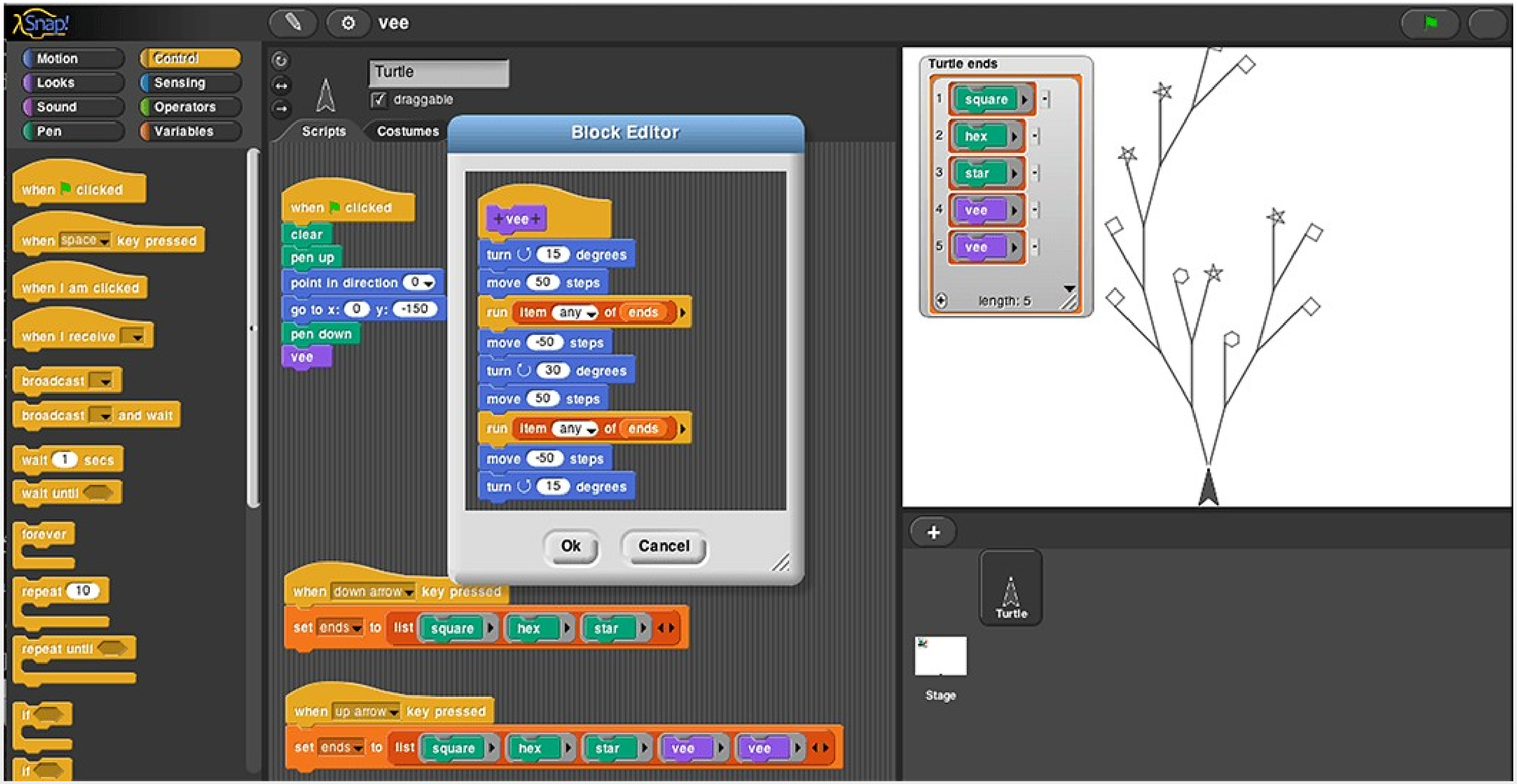


Scratch (2007)

- Probably the most used visual programming environment
- Over 29M shared scratch programs
- Lego-like blocks sticking together as counterpart of syntax rules
- Concurrency
- Event-driven programming
- Multimodality (graphics, sounds, animations, …)
- Fosters creativity
- Promotes reuse (remix)
- Inspired recent VP environments

Snap!

Originally BYOB. Actually used at Berkeley.



```
when [flag] clicked
when [space] key pressed
when I am clicked
when I receive [ ]
broadcast [ ]
broadcast [ ] and wait
wait (1) secs
wait until < >
forever
repeat (10)
repeat until < >
if < >
if < >
```

```
when [flag] clicked
clear
pen up
point in direction (0)
go to x: (0) y: (-150)
pen down
vee
```

```
+ vee +
turn ↻ (15) degrees
move (50) steps
run (item (any) of (ends))
move (-50) steps
turn ↺ (30) degrees
move (50) steps
run (item (any) of (ends))
move (-50) steps
turn ↻ (15) degrees
```

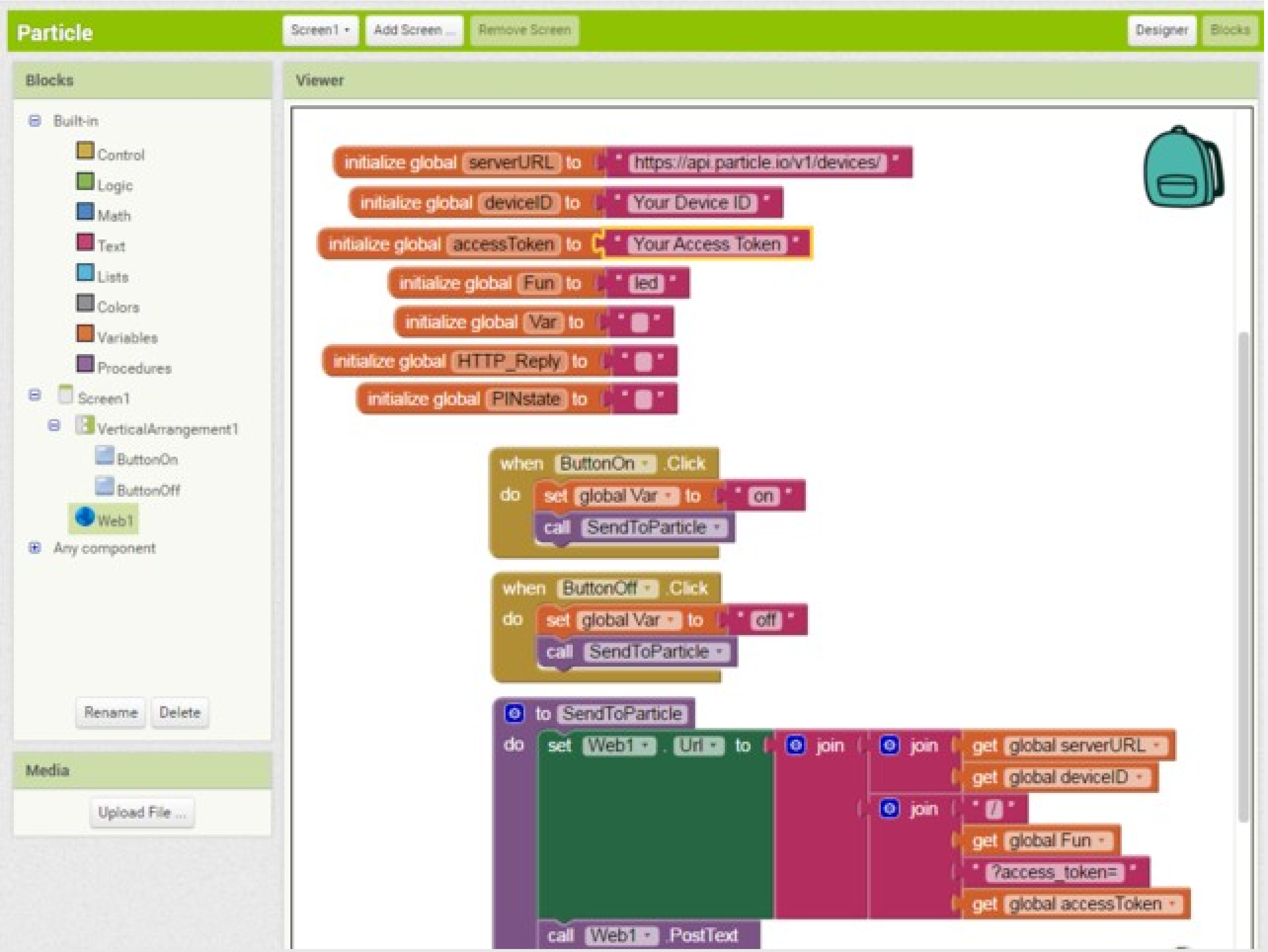
```
when [down arrow] key pressed
set [ends] to (list (square) (hex) (star))
```

```
when [up arrow] key pressed
set [ends] to (list (square) (hex) (star) (vee) (vee))
```

	Turtle ends
1	square
2	hex
3	star
4	vee
5	vee

AppInventor

Allows direct deployment on smartphones



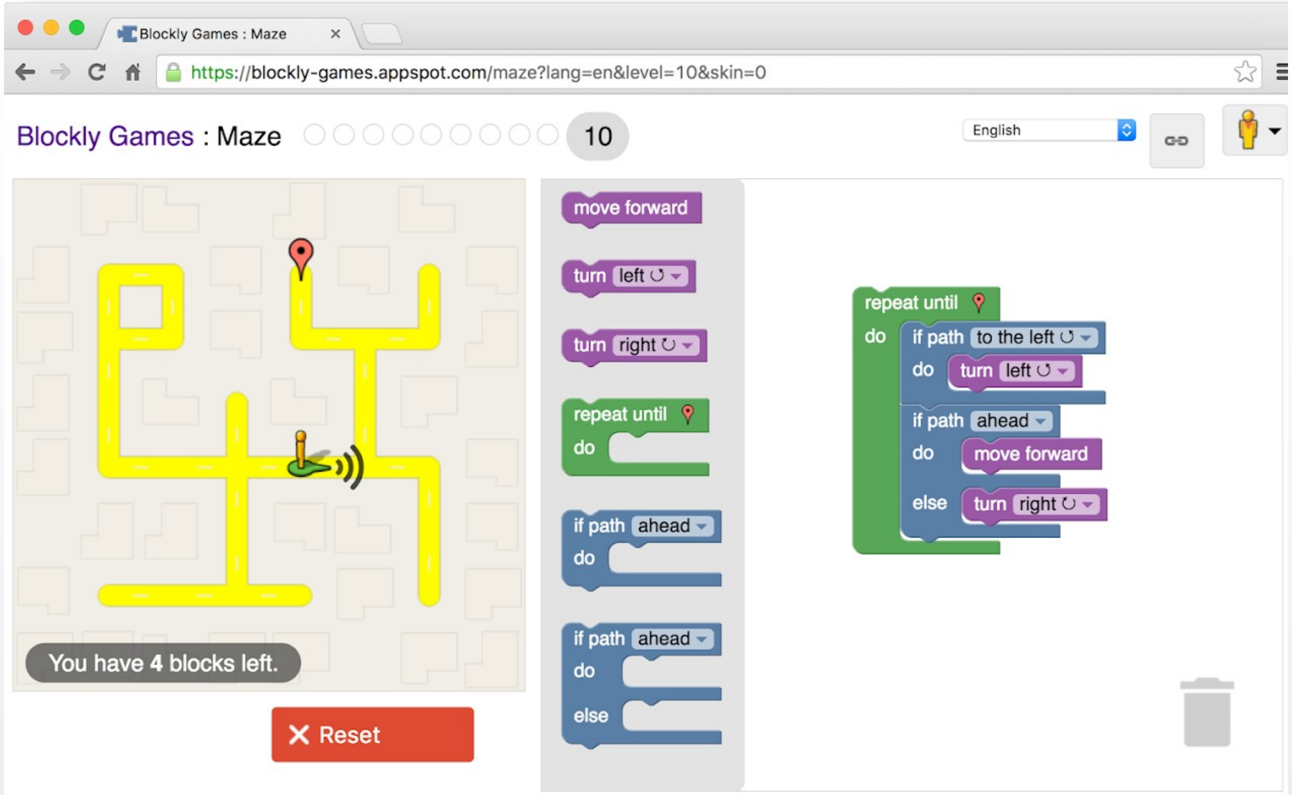
```
initialize global serverURL to " https://api.particle.io/v1/devices/ "
initialize global deviceID to " Your Device ID "
initialize global accessToken to " Your Access Token "
initialize global Fun to " led "
initialize global Var to " "
initialize global HTTP_Reply to " "
initialize global PINstate to " "

when ButtonOn .Click
do  set global Var to " on "
    call SendToParticle

when ButtonOff .Click
do  set global Var to " off "
    call SendToParticle

to SendToParticle
do  set Web1 . Url to  join  join  get global serverURL
                                   get global deviceID
                             join  " / "
                                   get global Fun
                                   " ?access_token= "
                                   get global accessToken
    call Web1 .PostText
```

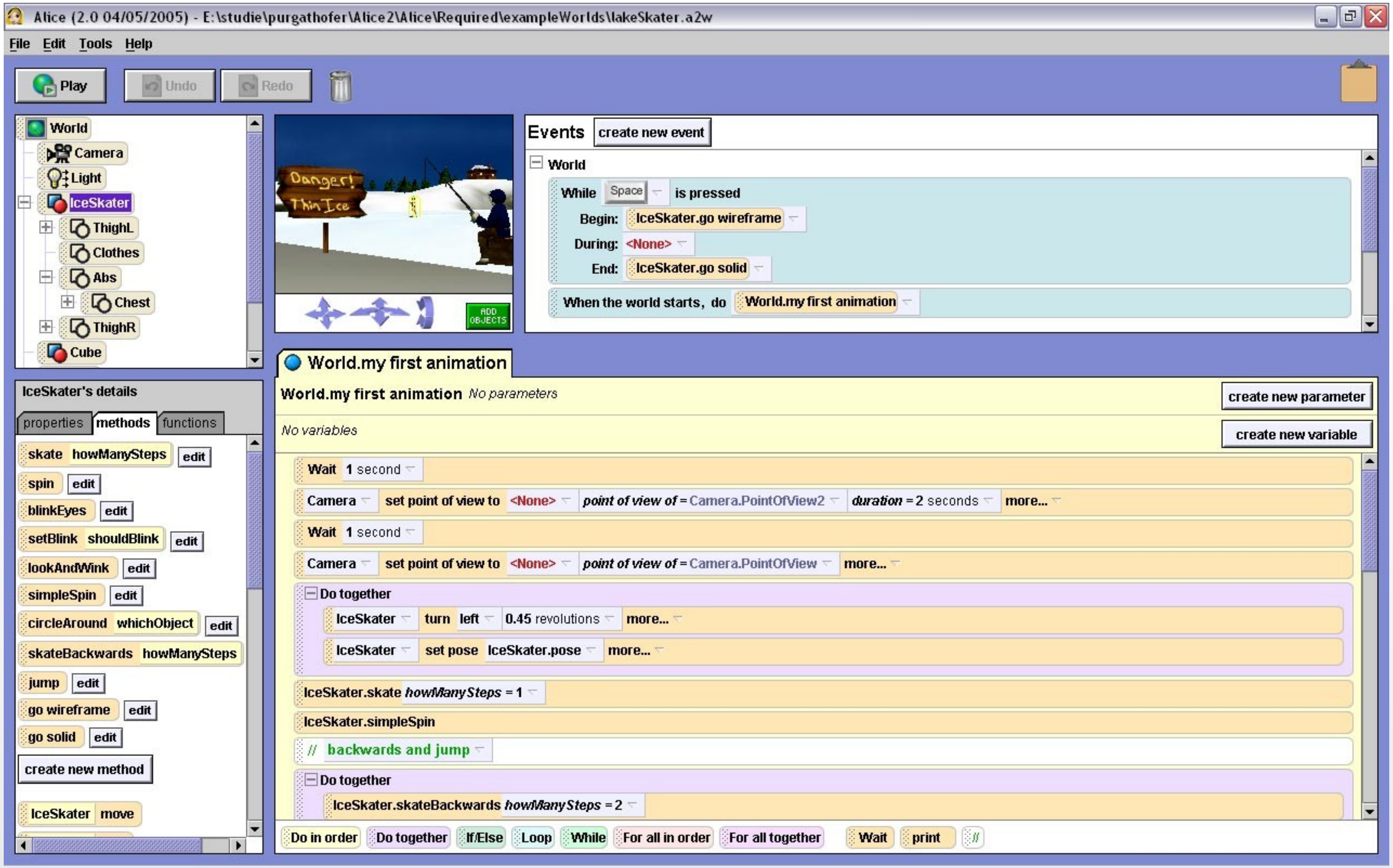
Blockly



You have 4 blocks left.

```
repeat until
do  if path to the left ↺
    do  turn left ↺
    if path ahead
    do  move forward
    else  turn right ↻
```

Alice



Alice (2.0 04/05/2005) - E:\studie\purgathofer\Alice2\Alice\Required\exampleWorlds\lakeSkater.a2w

```
World
  While Space is pressed
    Begin: IceSkater.go wireframe
    During: <None>
    End: IceSkater.go solid
  When the world starts, do World.my first animation
```

World.my first animation

```
World.my first animation No parameters
No variables
  Wait 1 second
  Camera set point of view to <None> point of view of = Camera.PointOfView2 duration = 2 seconds more...
  Wait 1 second
  Camera set point of view to <None> point of view of = Camera.PointOfView more...
  Do together
    IceSkater turn left 0.45 revolutions more...
    IceSkater set pose IceSkater.pose more...
  IceSkater.skate howManySteps = 1
  IceSkater.simpleSpin
  // backwards and jump
  Do together
    IceSkater.skateBackwards howManySteps = 2
```

Common features

- Personification
- Visualization and tracking
- Appeal

Sources

- O. Nierstrasz, Visual Programming, http://scg.unibe.ch/download/lectures/pl/PL-12VisualProgramming.pdf
- Lodi M., Malchiodi D., Monga M., Morpurgo A. e Spieler B., Constructionist Attempts at Supporting the Learning of Computer Programming: A Survey, Olympiads in Informatics 13 (2019), 99—121