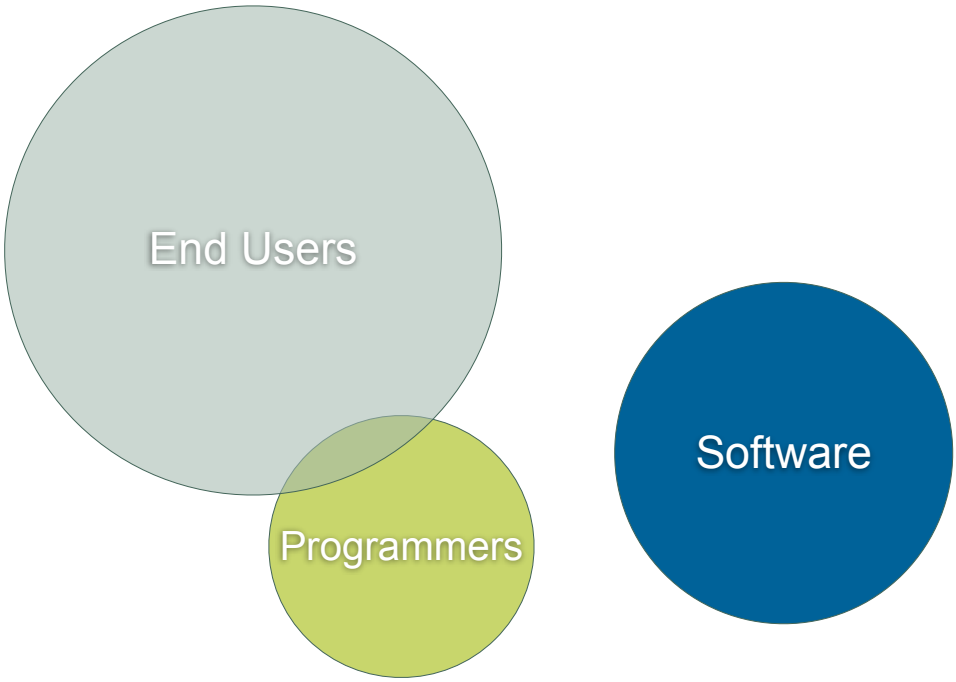
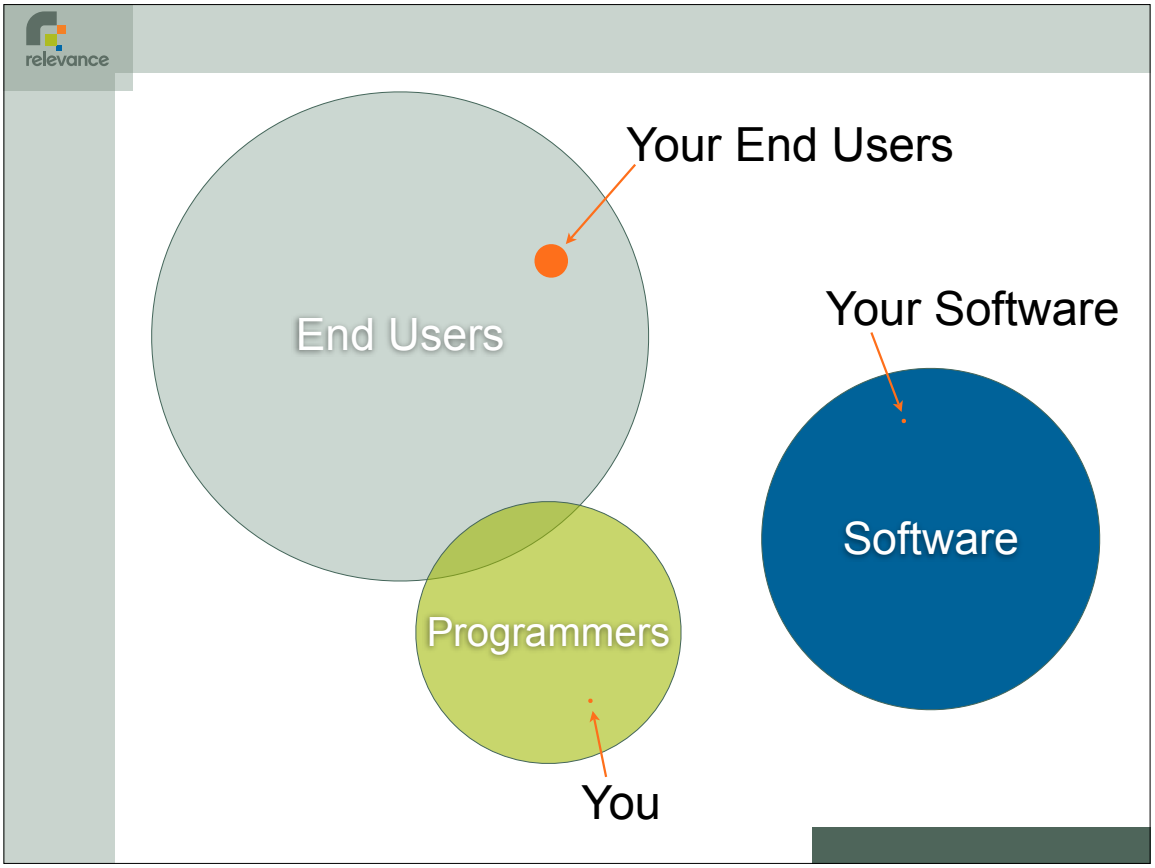
relevance

End User Programming

Glenn Vanderburg
Relevance, Inc.

relevance





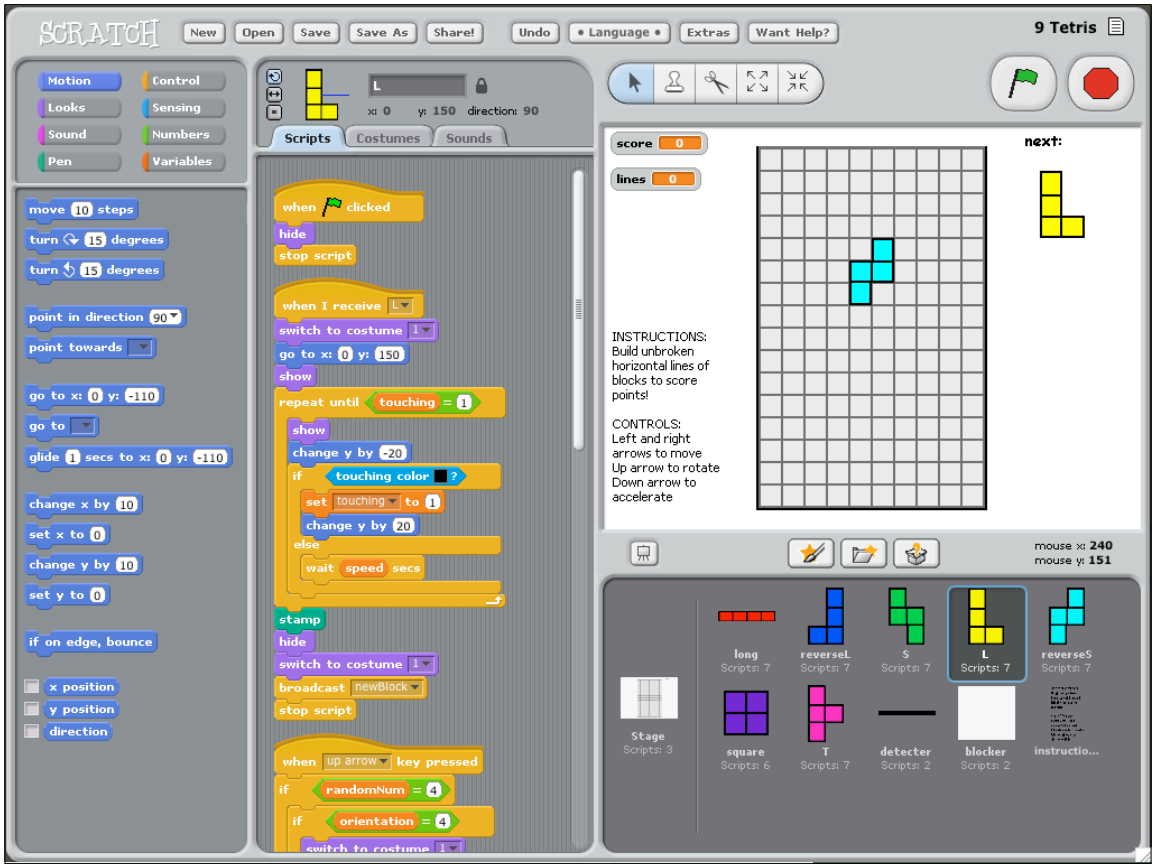
But How?

- **We will:**
  - **Briefly discuss the renaissance in end-user programming research**
  - **Examine notable successes and failures**
  - **Establish some principles for success**
  - **Create a plan of action**

End-User Programming Renaissance

Current Efforts

- Scratch
- Hackety Hack
- OLPC
- Processing
- Lego Mindstorms
- and more ...


SCRATCH
New
Open
Save
Save As
Share!
Undo
• Language •
Extras
Want Help?
9 Tetris
Motion
Control
Looks
Sensing
Sound
Numbers
Pen
Variables
move 10 steps
turn 15 degrees
turn 15 degrees
point in direction 90
point towards
go to x: 0 y: -110
go to
glide 1 secs to x: 0 y: -110
change x by 10
set x to 0
change y by 10
set y to 0
if on edge, bounce
x position
y position
direction
L
x: 0 y: 150 direction: 90
Scripts
Costumes
Sounds
when clicked
hide
stop script
when I receive L
switch to costume 1
go to x: 0 y: 150
show
repeat until touching = 1
show
change y by -20
if touching color ?
set touching to 1
change y by 20
else
wait speed secs
stamp
hide
switch to costume 1
broadcast newBlock
stop script
when up arrow key pressed
if randomNum = 4
if orientation = 4
score 0
lines 0
next:
INSTRUCTIONS:
Build unbroken horizontal lines of blocks to score points!
CONTROLS:
Left and right arrows to move
Up arrow to rotate
Down arrow to accelerate
mouse x: 240
mouse y: 151
long
Scripts: 7
reverseL
Scripts: 7
S
Scripts: 7
L
Scripts: 7
reverseS
Scripts: 7
Stage
Scripts: 3
square
Scripts: 6
T
Scripts: 7
detecter
Scripts: 2
blocker
Scripts: 2
instructio...


relevance
HACKETY HACK!

relevance

Hackety Hack

Controls | Skip Around

Lesson 2A

Downloading

Start a new program. (Click on the paper icon in the toolbar. It's the second one from the left.)

```
# A simple file downloader.
url = ask("Enter a Web address:")
save_as = ask("Save the file as:")
Web.download(url, save_as)
```

Save the program as **Downloader**.

Continue →

HACKETY HACK

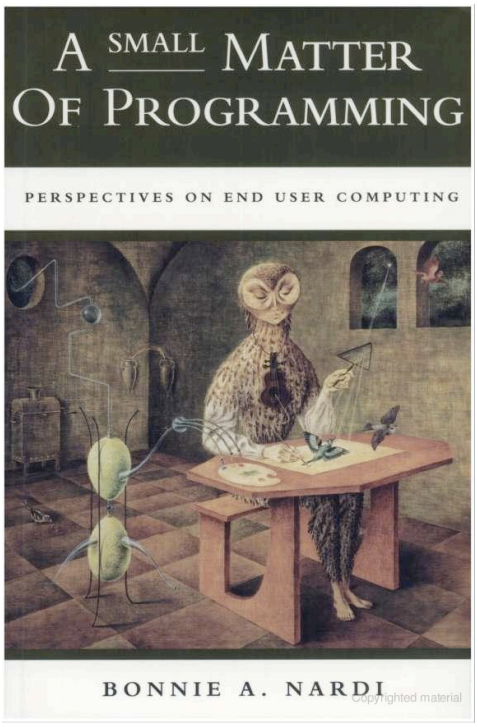
Downloader

A blank program, started on September 29th, 2008 at 7:50 AM.

```
# A simple file downloader.
url = ask("Enter a Web address:")
save_as = ask("Save the file as:")
Web.download(url, save_as)
```

Program will run below. Copy Save Run

relevance



Successes and Failures

Success: Spreadsheets

- VisiCalc
- Lotus 1-2-3
- Microsoft Excel
  - Probably the most popular programming platform ever

relevance

Car Loan Comparison6

Standard Car Loan Analysis

This worksheet can help you analyze a standard car loan. Enter values into the yellow boxes, writing over the sample values. Results will be shown in the green boxes.

Once you set the duration of the loan, you may need to extend the Amortization Table. If you decrease the duration, you may need to start again from a new worksheet.

This worksheet is locked to preserve the formulas that calculate your results. To unlock the worksheet, go to the Tools menu, select Protection, then choose Unprotect Sheet. This sheet does not use a

Analysis	
Amount financed	$12,500.00
Annual interest (e.g., 8.25)	8.25%
Duration of loan (in years)	3
Start date of loan	8/1/00
Monthly payments	$393.15
Total number of payments	=I7*12
Principal amount	$12,500.00
Finance charges	$1,653.32
Total cost	$14,153.32

Amortization Table

Pmt No.	Payment Date	Beginning Balance	Interest	Principal	Balance	Accumulative Interest	Accumulative Principal
1	8/1/00	12,500.00	85.94	307.21	12,192.79	85.94	307.21
2	9/1/00	12,192.79	83.83	309.32	11,883.47	169.76	616.53
3	10/1/00	11,883.47	81.70	311.45	11,572.02	251.46	927.98
4	11/1/00	11,572.02	79.56	313.59	11,258.43	331.02	1,241.57
5	12/1/00	11,258.43	77.40	315.75	10,942.68	408.42	1,557.32
6	1/1/01	10,942.68	75.23	317.92	10,624.77	483.65	1,875.23
7	2/1/01	10,624.77	73.05	320.10	10,304.66	556.70	2,195.34
8	3/1/01	10,304.66	70.84	322.30	9,982.36	627.54	2,517.64
9	4/1/01	9,982.36	68.63	324.52	9,657.84	696.17	2,842.16

Standard Car Loan Analysis

relevance

Microsoft Excel

- Informal handling of data
- Very loosely structured
  - You can put many “tables” on a sheet
  - Lone cells acting as variables
  - Excel gives everything a name for you
- Rich expression language

relevance

Failure: Lotus Improv

Lotus Improv - [Worksheet1 - View1 - Untitled1]

File Edit Create Worksheet Tools Window Help

Sales Persor

Mike

Months

			Nov	Dec	Year Total
Toaster	Tuesday	2002		10	10
		2003	0	11	11
	Wednesday	2002	100	50	150
		2003	110	55	165
	Thursday	2002	200	100	300
		2003	220	110	330

Products Days Year

1 Year Total = sum(Jan .. Dec)

2 :2003 = :2002 * 110%

Overlaps formula 1

Arial 9 General

File Edit Create Worksheet Tools Window Help

				Period1	Period2	Period3	Period4
Balance Sheet	Assets	Current Assets	Cash	200,000	10,326	74,863	188,791
			Accounts Receivable	0	60,000	50,000	40,000
			Inventories	0	27,600	33,120	39,744
			Vendor Deposits Made	0	10,000	10,000	0
			Total	200,000	107,926	167,983	268,535
		Noncurrent Assets	Plant	0	250,000	250,000	250,000
			Property	0	100,000	100,000	100,000
			Equipment	0	75,000	75,000	75,000
			Other	0	0	0	0
			Depreciation	0	(40,000)	(80,000)	(120,000)
			Total	0	385,000	345,000	305,000
		Total		200,000	492,926	512,983	573,535
	Liabilities and Equity	Liabilities	Accounts payable	0	50,000	55,173	65,541
			Customer Deposits	0	2,500	5,000	7,500
			Amortized Debt	0	150,000	141,730	132,411
			Credit Line used	0	50,000	40,000	30,000
			Total	0	252,500	241,903	235,452
		Equity	Contributed Capital	200,000	200,000	200,000	200,000
			Dividends Paid	0	(10,000)	(25,000)	(50,000)
			Retained Earnings		40,426	71,080	138,083
			Net	200,000	240,426	271,080	338,083
		Total		200,000	492,926	512,983	573,535
Income Statement	Revenue	Product Sales			650,000	795,000	993,200
		Service Fees			100,000	125,000	150,000
		Other			30,000	46,000	30,000
		Total		0	780,000	966,000	1,173,200
	Cost of Goods Sold	Inventory	Beginning	0	0	27,600	33,120
			Additions	0	276,000	331,200	397,440
			Ending	0	27,600	33,120	39,744
			Cost	0	248,400	325,680	390,816
		Factory and Labor		0	192,000	230,400	276,480
		Depreciation			40,000	40,000	40,000
		Total		0	480,400	596,080	707,296
	Gross Margin			0	299,600	369,920	465,904
	Operating Expenses	General and Admin			100,000	120,000	130,000
		Research and Development			29,000	33,800	39,560
		Sales and Marketing			113,000	134,600	160,520
		Total		0	242,000	288,400	330,080
	Operating Income			0	57,600	81,520	135,824
	Interest Income				826	5,989	15,103
	Interest Expense				6,000	22,355	18,924

FINANCE.IMP - C:\IMPROV\MODELS\EXAMPLES

Financials - Balance Sheet Report
Financials - Cash Flow Report
Financials - Complete
Financials - Income Statement Report

This view isolates Balance Sheet data.

It is formatted and prepared as a printed report.

```
1  //----------------------------BALANCE SHEET FORMULAS-----------------------------------\\
2  Assets.Current Assets.Cash = Cash Flow.Ending Cash
3  Period1:Assets.Current Assets.Cash = Liabilities and Equity.Equity.Contributed Capital:Period1
4  Assets.Current Assets.Inventories = Income Statement.Cost of Goods Sold.Inventory.Ending
5  Current Assets.Total = groupsum(Current Assets)
6  Noncurrent Assets.Total = groupsum(Noncurrent Assets)
7  Assets.Total = groupsum(Assets)
```

Failure: Lotus Improv

- By most standards, much better than Excel
  - Inherently multidimensional
  - More structured and sophisticated
- But it failed.
  - Easier to do sophisticated things
  - Harder to do simple things
  - Harder to *explore* your problem

What Went Wrong?

Improv set out "to fix all this". It was an auditors dream. It provided rarified heights of abstraction, formalisms for rows and columns, and in short was truly comprehensible. It failed utterly, not because it failed in its ambitions but because it succeeded.

—Adam Bosworth

In the end it didn't go anywhere, probably because in setting out to improve on spreadsheets, Improv lost the essence of a spreadsheet …

—Pito Salas, inventor of Improv

Success: Ruby DSLs

- Rich Kilmer, JAOO 2007
- USAF system for managing mid-air refueling network.
- Core of system described in Ruby code.
- Non-programmer domain experts reading, correcting, and even writing new Ruby code for the system.
- That code formed the core of the running system.

Ruby DSL Example

```
# I'm guessing at what the real thing
# looks like -- Glenn
coronet :grand_forks do
  base     'Grand Forks AFB'
  tankers :long_range   8
  tankers :short_range 15
  location [47.964296, -97.394829]
end
```

Failure: AppleScript

```
on get_header_from_message(desiredHeader, theMessage)
  tell application "Mail"
    set hdrs to (headers of theMessage)
    repeat with hdr in hdrs
      if name of hdr is desiredHeader then
        return contents of hdr
      end if
    end repeat
    return ""
  end tell
end get_header_from_message
```

Failure: AppleScript

- Scripting system for MacOS
- Looks just like English!
  - Which is the problem.
  - It doesn't *act* like English.
- People don't have a big problem with formal languages.
  - They just want to have clear rules
  - And sensible behavior in the face of mistakes

relevance

What Went Wrong?

The experiment in designing a language that resembled natural languages was not successful. […] In the end the syntactic variations and flexibility did more to confuse programmers than help them out.

The main problem is that AppleScript only appears to be a natural language on the surface. In fact is an artificial language, like any other programming language […] even small changes to the script may introduce subtle syntactic errors which baffle users. It is very easy to read AppleScript, but quite hard to write it.

—William Cook, designer of AppleScript

relevance

Success: DabbleDB

- Web application for managing data
  - You build your own apps to suit your data
  - Goal: be the platform for every system that's written in Excel but shouldn't be
- Different model from Excel, but similar lessons:
  - Do sensible things with no direction from user
  - Allow user to add structure and metadata gradually
  - Programmable using formulas.

relevance

Dabble DB: Expected value by month

DABBLE DB | Help | Search All | Applications | User Account | Admin | Logout

Sale | More... | Import Data | Sharing

+ New entry

91 entries

Sale

Search

Close Date is

Send feedback to the Dabble DB team

Show all tips

Expected value by month (Modified: Save, Revert, Save as...)

Share | Print | Export

Close Date » Month	Value	Value × Probability
November 2007	$260,000.00	$198,750.00
December 2007	$2,905,000.00	$1,792,700.00
January 2008	$2,605,000.00	$1,496,850.00
February 2008	$6,035,000.00	$2,937,100.00
March 2008	$2,665,000.00	$1,356,700.00
April 2008	$2,190,000.00	$1,247,650.00
	$16,660,000.00	$9,029,750.00

Go to "http://localhost:9090/dabble/dabbledb?view=646&_k=xYpnmFgN&x51"

relevance

Dabble DB: Expected value by month

DABBLE DB | Help | Search All | Applications | User Account | Admin | Logout

Sale | More... | Import Data | Sharing

53 entries

Sale

Search

Close Date is

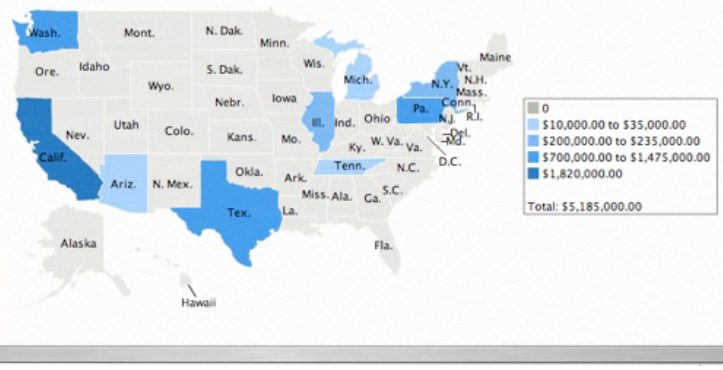
q1 2008

Send feedback to the Dabble DB team

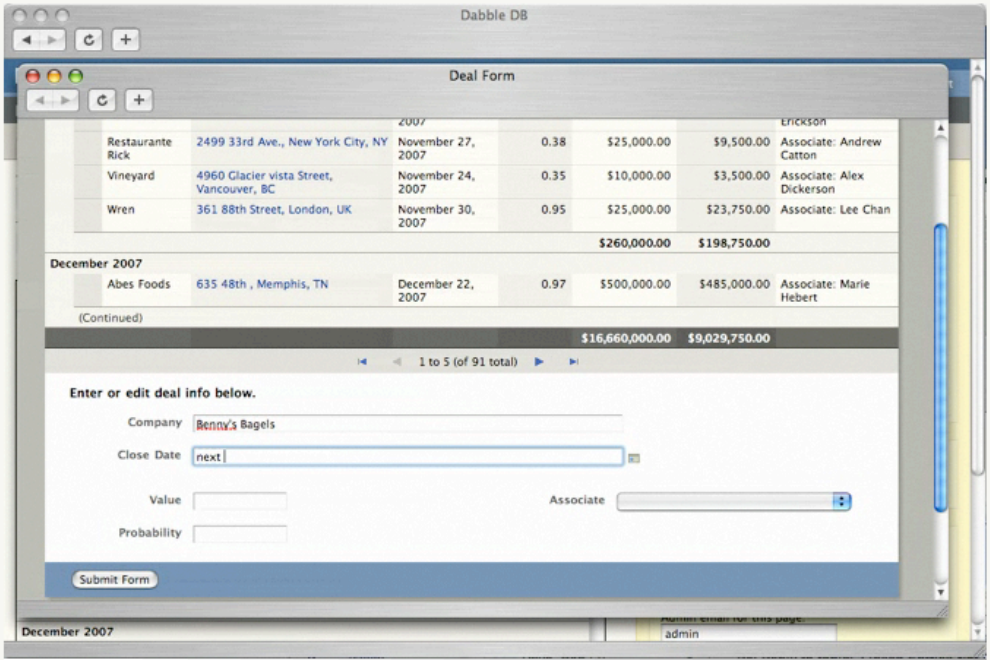
Show all tips

Expected value by month (Modified: Save, Revert, Save as...)

Share | Print | Export | SHOWING: Value (USD) | FOR: United States

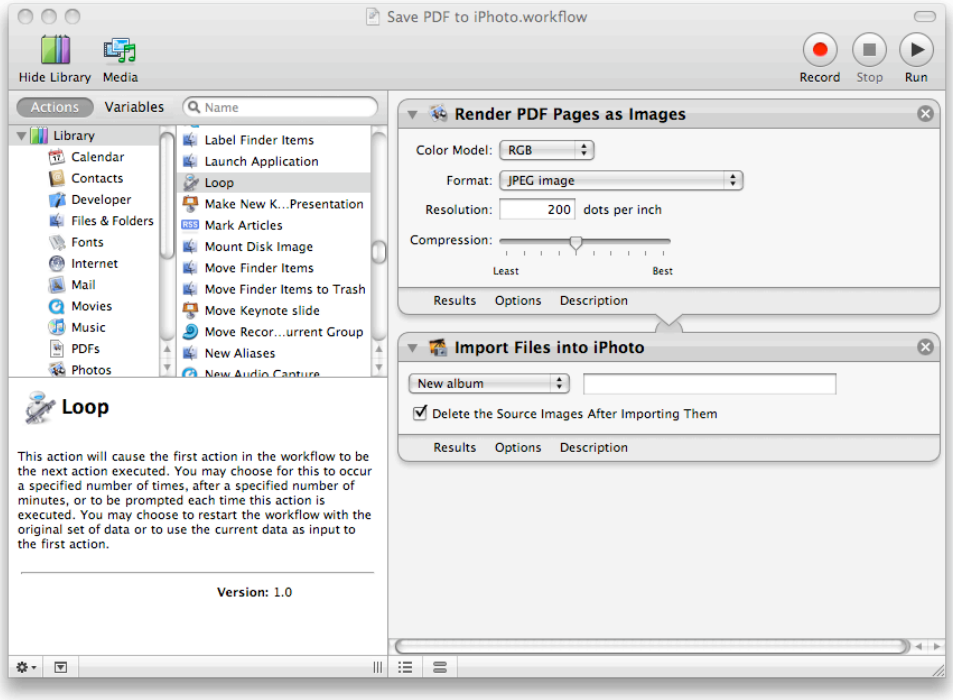


Cancelled opening the page



Failure (So Far): Automator

- Apparently an attempt to replace AppleScript
- A visual programming system
  - Follows a pipes-and-filters model
  - Configurable filters; no real runtime decisions
- Very broad; too shallow
- Still evolving



Success: Mingle

- Project collaboration and management tool
- Doesn't mandate a development process
- Teams build a system that fits their process
  - Cards, properties, formulas, transitions
  - Charts and tables
- Has been applied in unexpected ways

relevance

Select tree:
(no configured trees)

View as: List Grid

Manage trees

Group by: Status Color and Sort by: Priority

Lane headings: Count Select property...

Link to this page Add / remove lanes

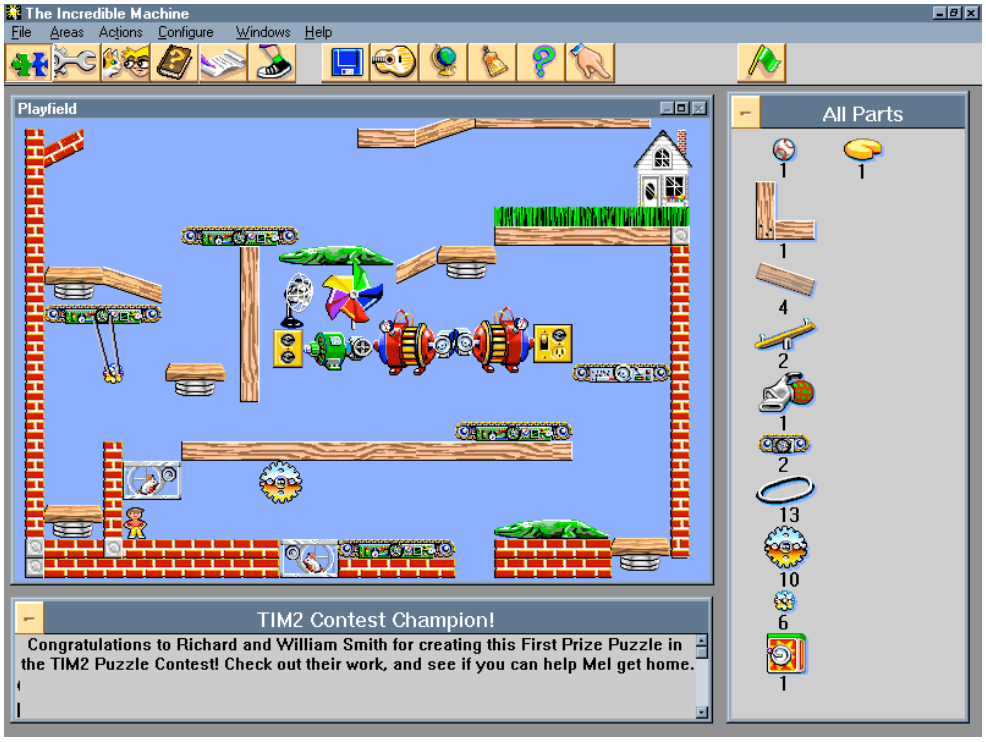
New (1)	Open (1)	Narrative started (2)	Ready for Development (6)	Ready for Testing (2)	Ready for Showcase (1)	Deleted (1)
Display #422 validation error messages in proximity to field in error	Apply #423 new UI for secondary pages	Produce #411 CSS and graphics for new secondary page look-and-	Produce #420 graphical prototype of interior static content pages	Add #399 "How Points Work" explanation	Update #404 graphical prototype of UI redesign for secondary	#400 Include/Exclude Selected Categories From User
		#316 Implement New Home Page	Update #419 graphical prototype of UI redesign for secondary	Upgrade #401 to Rails 2.0		
			Produce #418 graphics for new home page look-and-feel			
			Add #412 newsletter toggle to user profile			
			Produce #410 CSS for new home page look-and-feel			
			Produce #421 graphical prototype of *printed* catalog detail			

relevance

Success: Puzzle Games

relevance

relevance



relevance

relevance

relevance

Success: Puzzle Games

- I think the essence of programming shows in puzzle-oriented games:
  - Lemmings
  - The Incredible Machine
  - Professor Fizzwizzle
  - Enigmo
- Popularity indicates children becoming accustomed to programming challenges.
- Wait a generation.

relevance

Principles for Success

relevance

Constrain to a Domain

- Success stories are all *domain specific!*
- Allows focus on the task
- Available facilities make sense in context
- General-purpose facilities can be present, but should be secondary

relevance

Allow, Don't *Require* Structure

- Start with expressions, declarations, and data, not programs
- Structuring mechanisms should be optional
- Optimize for easy start and exploration

relevance

Act, Don't *Look* Human

- Many people thing these are required:
  - Natural-language syntax
  - Visual programming
- What matters more:
  - Simple rules
  - Not much punctuation
  - Good error messages
  - Sensible default behavior
  - Ability to start small and explore

relevance

Imperative, OO, Functional?

- I think most people relate to *imperative* programming best.
  - Tcl's command-oriented syntax seems ideal.
- But success stories *so far* don't bear that out.
  - Excel is practically functional programming.
  - SQL (including Mingle's query language) and Rich Kilmer's Ruby DSLs are declarative.

A Plan of Action

Recipe 1 (1970s)

- Bentley, Kernighan, and the Bell Labs crowd
- Design a language and implement a processor or translator
- Examples: pic, tbl, eqn, grap, chem
- Works great if you’re the guys who invented yacc and lex

relevance

Recipe 2 (1980s)

- Alan Kay, Dan Ingalls, etc.
- Immerse the user in a sea of objects!
- Smalltalk users will modify their environment by programming.
- Still real potential here for *programmers*, but not for end users.

relevance

Recipe 3 (1990s)

- Many folks, but mostly John Ousterhout
- Build your system in two halves:
- Core domain logic implemented as a DSL
- Rest of system implemented in that DSL
- Success with this approach has been rare
  - Seems too costly up front
  - Your users might not want or need such a powerful language

relevance

Recipe 4 (2000s)

- Popping up everywhere (but Eric Evans gets special credit)
- Get the *domain language* at the core of the system right
  - Maybe involving domain-specific programming constructs
  - But the important thing is the system of objects and names
- Users will be thinking in that language already.

relevance

How Recipe 4 Works

- If you get the *domain language* right, building a *domain-specific language* is easy.
- If you are writing in a metaprogrammable language, an internal DSL will happen naturally.
- Domain-driven design helps you separate *essence* from *accident.*
- A system with good hooks for adding an external DSL, if necessary.

relevance

Summary

- Learn from the past
- Cater to your users' strengths
  - domain experts, language users
- Allow exploration and improvisation
- Focus on the domain
- Clean internal design facilitates exposing the internals