

# Non-linguistic Thinking as an Effective Tool for Innovation in Education

Leon Brânzan, Technical University of Moldova, <https://fcim.utm.md/>

# Is software getting worse?

## The Thirty Million Line Problem

174K views • 4 years ago



Molly Rocket

A historical argument for creating a stable instruction set architecture (ISA) for entire system-on-a-chip (SoC) packages. Pleas



An SoC is a Super-powered 1980s Home Cor | Requirements for Direct Coding | A Collaborative Effor... 4 momen

## Where Does Bad Code Come From?

94K views • 11 months ago



Molly Rocket

<https://www.kickstarter.com/projects/annarettberg/meow-the-infinite-book-two> Q&A: [https://youtu.be/cOcaj\\_cRBvE](https://youtu.be/cOcaj_cRBvE) Rel



Two Where Does Bad Code Come from | Experienced Programmers | Bad Code Where Does It Come... 5 m

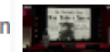
## Preventing the Collapse of Civilization / Jonathan Blow (Thekla, Inc)

136K views • 3 years ago



DevGAMM

A discussion about how they make software and what this means by Jonathan Blow, Thekla, Inc. DevGAMM is the biggest game ...



The Space Race | Antikythera Mechanism | Technology Goes Backward | The Late Bronze Age | Gma... 19 chapters

## "Stop Writing Dead Programs" by Jack Risher (Strange Loop 2022)

70K views • 9 days ago



Strange Loop Conference

Most new programming languages are accidentally designed to be backwards compatible with punchcards. This talk argues that it ...

“We are in the process of digging ourselves into an anachronism by preserving practices that have no rational basis beyond their historical roots in an earlier period of technological and theoretical development.”

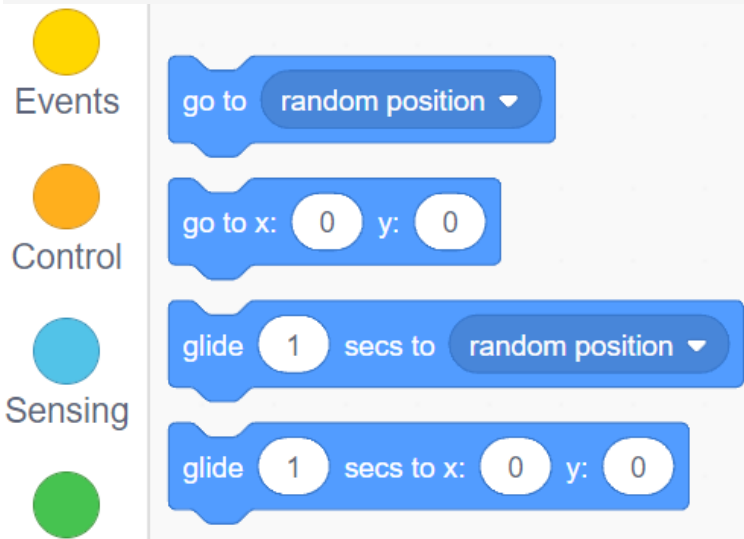
Seymour Papert, 1980

# Is software getting worse?

---

- Are programmers getting worse?
- Are the tools getting worse?
- Are we teaching programming the wrong way?

# How programming is taught



```
#include <stdio.h>
```

```
void main(int argc, char *argv[], char *envp[])  
{  
    printf("hello, world!");  
}
```

```
.LC0:
```

```
    .string "hello, world!"
```

```
main:
```

```
    pushq %rbp
```

```
    movq %rsp, %rbp
```

```
    subq $32, %rsp
```

```
    movl %edi, -4(%rbp)
```

```
    movq %rsi, -16(%rbp)
```

```
    movq %rdx, -24(%rbp)
```

```
    movl $.LC0, %edi
```

```
    movl $0, %eax
```

```
    call printf
```

```
    nop
```

```
    leave
```

```
    ret
```

# Why learning how to program is hard

- Programming languages should come last
- Text should be used less
- Specific skills and intuition should be developed early

# Programming languages

---

$$3 + x = 5$$

# Programming languages

---

**Python → C → Assembly → Machine code**

**Data-oriented > Object-oriented > Functional > Structured**

**$x = j$  if  $a > b$  else  $k$**

**$x = a > b \rightarrow j, k$**

**$\text{int } x = (a > b) ? j : k;$**

# Textual form

---

- Formal language  $\neq$  natural language
- Cognitive load theory
- Visual information processing vs. textual information processing
- Reasoning and language processing are done in different parts of the brain



# Specific set of skills

Handmade Asset Rendering

```
147 }
148 }
149
150 internal world_sim
151 BeginSim(game_assets *Assets, memory_arena *TempArena, world *World, world_position SimCenterP, rec
152 {
153     world_sim Result = {};
154
155     // TODO(casey): How big do we actually want to expand here?
156     // TODO(casey): Do we want to simulate upper floors, etc.?
157     temporary_memory SimMemory = BeginTemporaryMemory(TempArena);
158
159     sim_region *SimRegion = BeginWorldChange(Assets, TempArena, World, SimCenterP, SimBounds, dt);
160
161     Result.SimRegion = SimRegion;
162     Result.SimMemory = SimMemory;
163
164     return(Result);
165 }
166
167 internal void
168 Simulate(world_sim *WorldSim, f32 TypicalFloorHeight, random_series *GameEntropy, r32 dt,
169         v4 BackgroundColor, game_state *GameState,
170         game_assets *Assets, game_input *Input, render_group *RenderGroup,
171         particle_cache *ParticleCache, editable_hit_test *HitTest)
172 {
173     sim_region *SimRegion = WorldSim->SimRegion;
174
175     // NOTE(casey): Run all brains
176     BEGIN_BLOCK("ExecuteBrains");
177     for(u32 BrainIndex = 0;
```

Output Breakpoints Memory

0000018E7DB0000: 01 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00  
0000018E7DB0010: 40 00 DB B7 8E 01 00 00 20 00 00 00 00 00 00 00  
0000018E7DB0020: 00 00 00 00 00 00 00 00 00 00 CA B7 8E 01 00 00  
0000018E7DB0030: 00 00 EC B7 8E 01 00 00 00 00 00 00 00 00 00 00  
0000018E7DB0040: 00 00 DB B7 8E 01 00 00 00 00 10 00 00 00 00 00  
0000018E7DB0050: 01 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00  
0000018E7DB0060: 80 00 00 00 00 00 00 00 00 E2 A4 00 01 00 00 00  
0000018E7DB0070: 04 00 00 00 01 00 00 00 FF 9F FF 9F 03 03 01 80  
0000018E7DB0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB00A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB00B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB00C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB00D0: 00 00 00 00 00 00 00 00 29 09 80 04 00 00 00 00  
0000018E7DB00E0: BF E2 A4 00 00 00 00 00 DC 28 A0 00 04 00 00 00  
0000018E7DB00F0: 07 00 00 00 01 00 00 00 FF 9F FF 9F 03 03 01 80  
0000018E7DB0100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB0110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB0120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB0130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB0140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB0150: 00 00 00 00 00 00 00 00 29 09 5F 04 00 00 00 00  
0000018E7DB0160: DA 0B 45 01 00 00 00 00 0C 60 2F 00 07 00 00 00  
0000018E7DB0170: 0A 00 00 00 01 00 00 00 FF 9F FF 9F 03 03 01 80  
0000018E7DB0180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB0190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB01A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB01B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB01C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0000018E7DB01D0: 00 00 00 00 00 00 00 00 29 09 4B 01 00 00 00 00

Prev page Next page Address: SimMemory.Block->Base (0000018E7DB0040)

Registers

RIP: 00007FF953AEE50F

RAX: 0000000FD28FA520	RBX: 0000000000000000	RCX: 0000000000000000	RDX: 0000018E7DB0040
RSI: 0000000FD28FA4E0	RDI: 0000000FD28FA538	R8: 0000018E7DB0040	R9: 0000018E8540040
R10: 0000000000000000	R11: 0000000000000246	R12: 0000000000000000	R13: 0000000000000000
R14: 0000000000000000	R15: 0000000000000000	RSP: 0000000FD28FA470	RBP: 0000000000000000

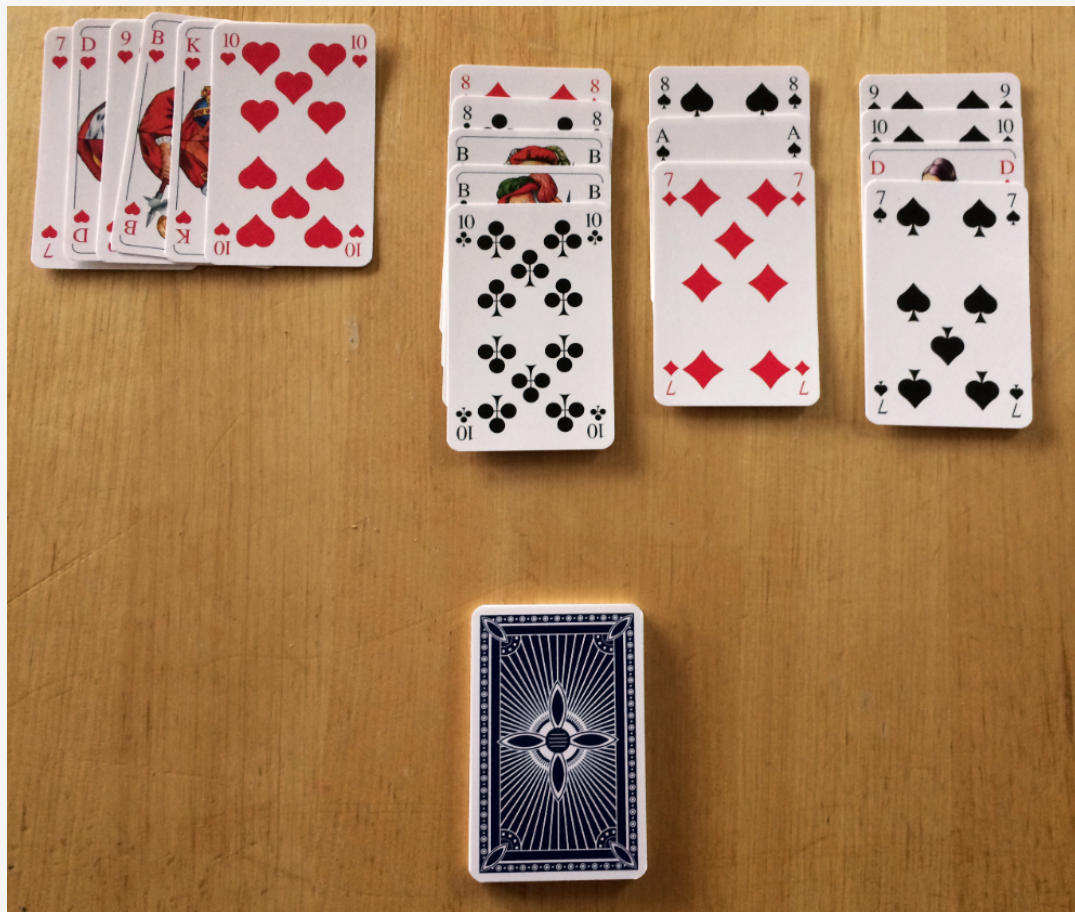
Call Stack

Name	Value	Type
SimMemory.Block->Base, dw		memory watch
0000018E7DB0040	0000 b7db 018e 0000 0000 0010 0000 0000	
0000018E7DB0050	0001 0000 0000 0000 0002 0000 0000 0000	
0000018E7DB0060	0080 0000 0000 0000 e200 00a4 0001 0000	
0000018E7DB0070	0004 0000 0001 0000 9fff 9fff 0303 8001	
0000018E7DB0080	0000 0000 0000 0000 0000 0000 0000 0000	
0000018E7DB0090	0000 0000 0000 0000 0000 0000 0000 0000	
0000018E7DB00A0	0000 0000 0000 0000 0000 0000 0000 0000	
0000018E7DB00B0	0000 0000 0000 0000 0000 0000 0000 0000	
xmm0, ps	0.000000 0.000000 0.000000	u128
ymm0, ps	0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.016667	u256
SimMemory.Block->Base, dyb /c8 /n64		memory watch
0000018E7DB0040	00000000 00000000 11011011 11101101 01110001 10000000 00000000 00000000 00	
0000018E7DB0048	00000000 00000000 00001000 00000000 00000000 00000000 00000000 00000000 00	
0000018E7DB0050	10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 01	
0000018E7DB0058	01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 02	
0000018E7DB0060	00000001 00000000 00000000 00000000 00000000 00000000 00000000 00000000 80	
0000018E7DB0068	00000000 01000111 00100101 00000000 10000000 00000000 00000000 00000000 00	
0000018E7DB0070	00100000 00000000 00000000 00000000 10000000 00000000 00000000 00000000 04	
0000018E7DB0078	11111111 11111001 11111111 11111001 11000000 11000000 10000000 00000001 ff	

# Solutions

- Guiding the learning process with visual tools to promote inference and reflection
- Choosing the right computational model
- Developing engineering thinking early by teaching fundamental concepts first

# Inference and reflection



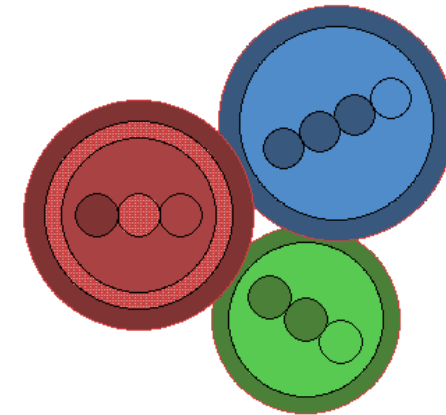
# Computational models

## Turing machine

Symbol	Write	Move
Blank	None	None
0	1	Left
1	0	Left

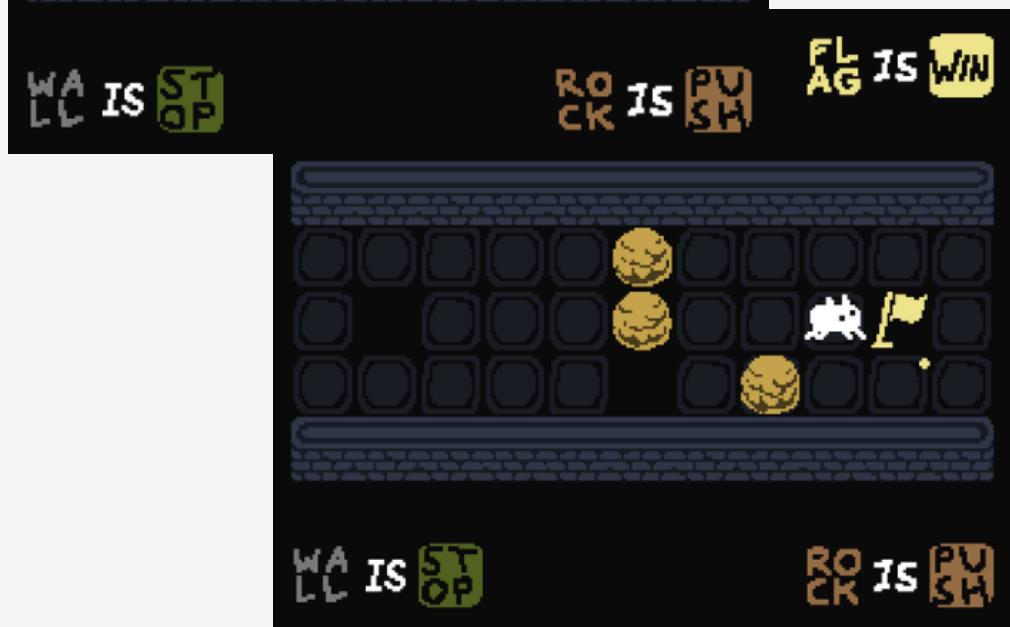
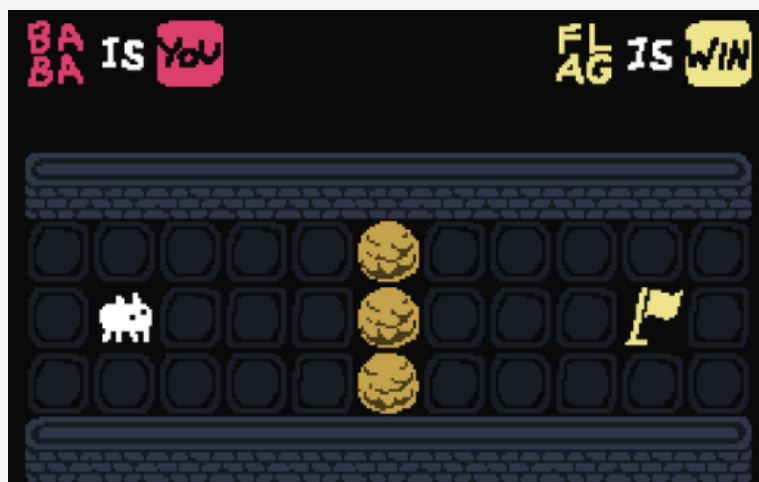


## $\lambda$ -calculus



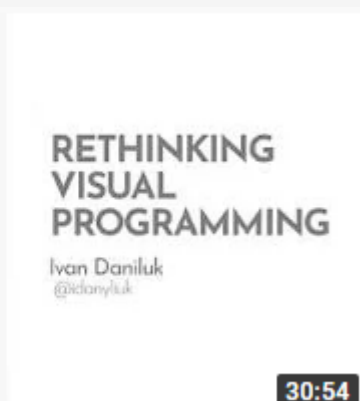


# Fundamental concepts



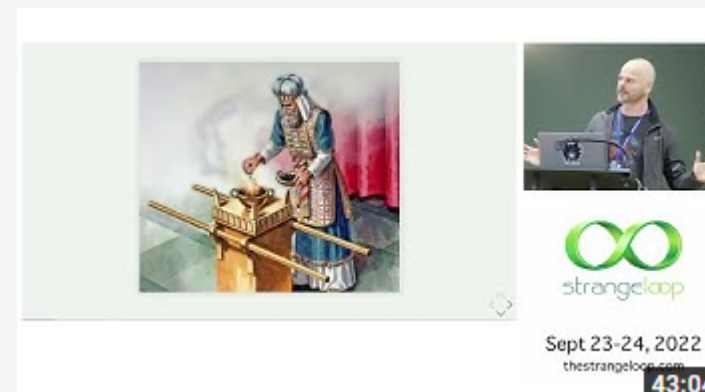
# Food for thought

Rethinking visual programming,  
Ivan Danyliuk



Programming with nothing,  
Tom Stuart

Video games and the future of education,  
Jonathan Blow



Stop writing dead programs,  
Jack Rusher

# Presentation, links and references

<https://bit.ly/case-for-non-linguistic-teaching>

