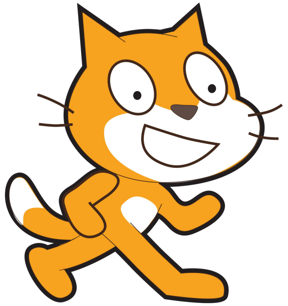
scratch on transFORM: a tangible programming environment

Anthony Baker
Scott Penman
Andrew Ringler



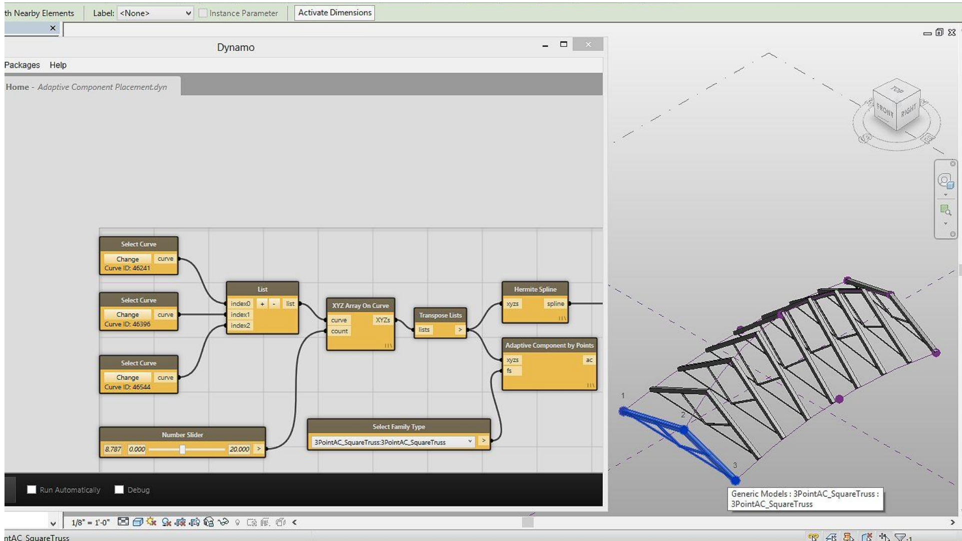
algorithmic thinking

“We’ve got to have our kids in math and science, and it can’t just be a handful of kids. It’s got to be everybody. Everybody’s got to learn how to code early.”

- POTUS #44

visual coding

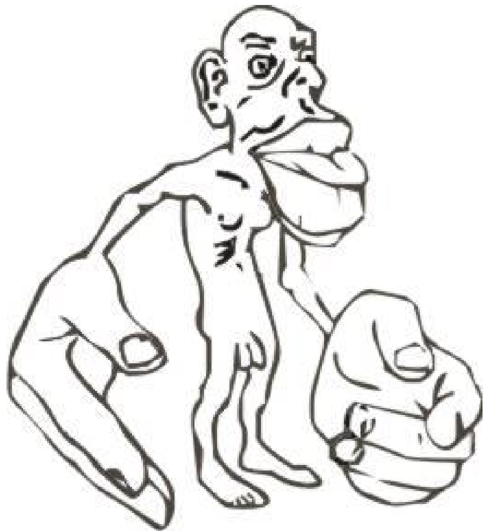
i.e., blockly, scratch, pencil code, nodebox, grasshopper, dynamo



“object to think with”

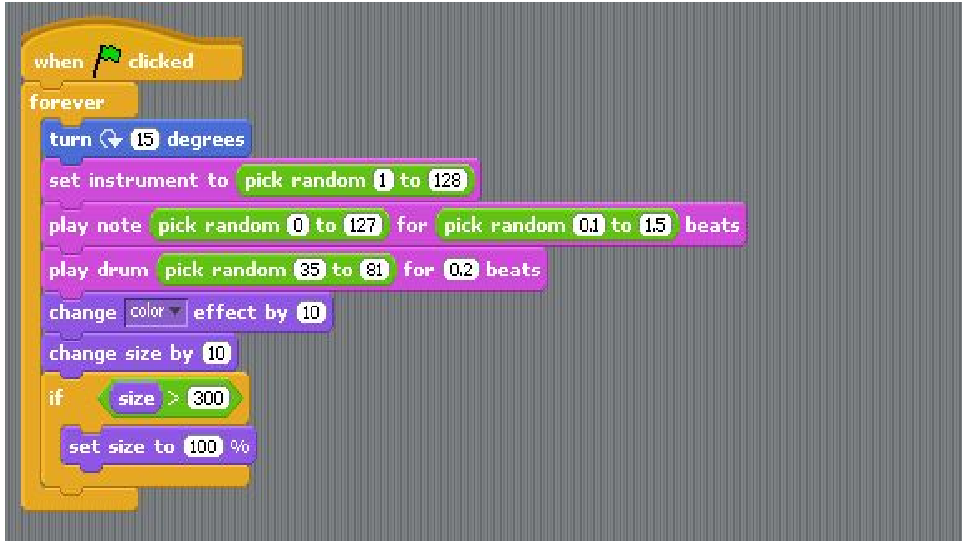
there is “a dynamic relationship between things and thinking. We tie a knot and find ourselves in a partnership with string in our exploration of space. Objects are able to catalyze self creation.”

- Sherry Turkle



scratch(.mit.edu)

block-based programming language
that allows children to create interactive media



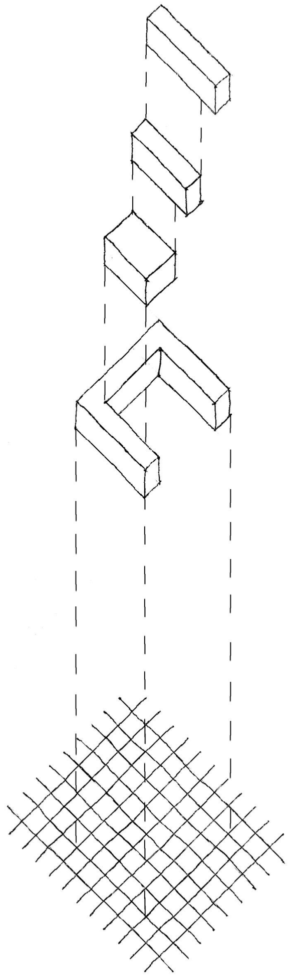
```
when 🏴 clicked
forever
    turn ↻ (15) degrees
    set instrument to (pick random (1) to (128))
    play note (pick random (0) to (127)) for (pick random (0.1) to (1.5)) beats
    play drum (pick random (35) to (81)) for (0.2) beats
    change [color ▾] effect by (10)
    change size by (10)
    if <(size) > (300)>
        set size to (100) %
```

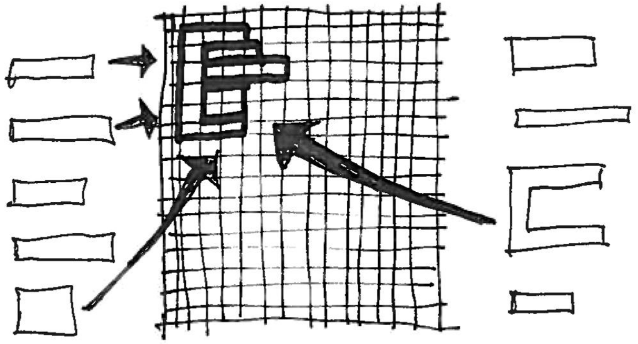
implementation idea #1

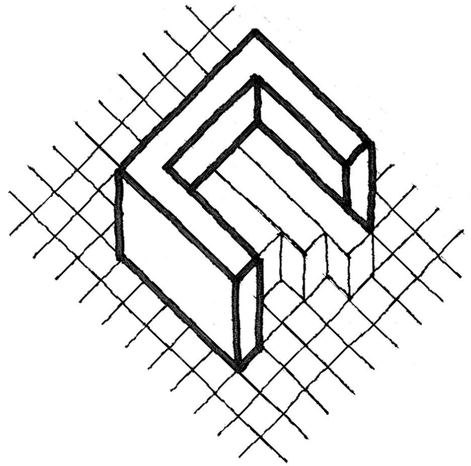
physical blocks as tokens on TransFORM

transform can recognize block shapes by pins

display output on another transform surface







implementation idea #2

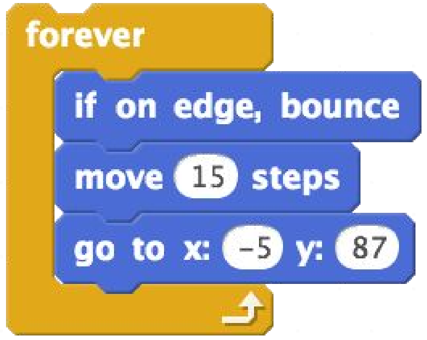
blocks are displayed virtually to the side of board

blocks are gestured onto the board, at which point they become “real” (3D)

blocks can be resized, reordered, removed, etc.

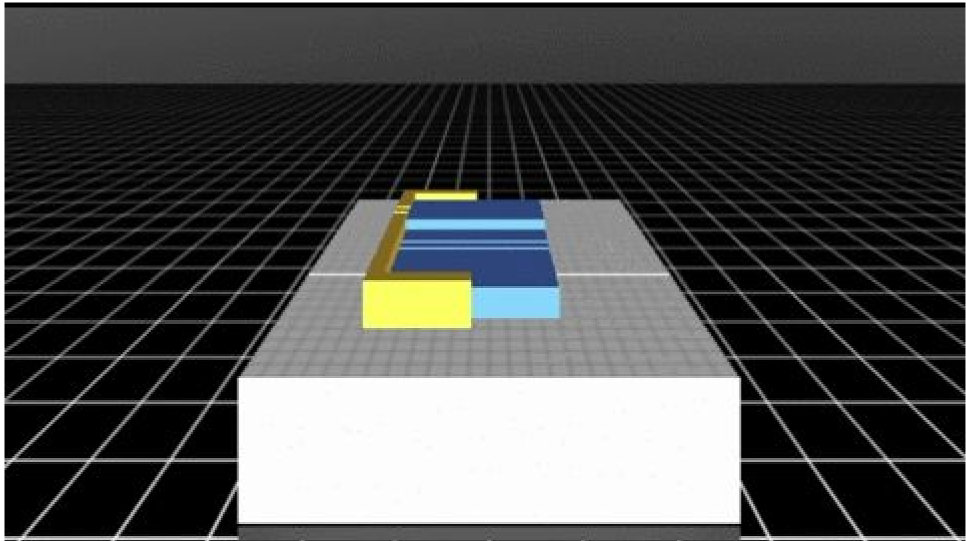
computational concepts

looping
drawing
abstraction
arrays
parameterization
variables
math
conditionals



```
forever
  if on edge, bounce
  move (15) steps
  go to x: (-5) y: (87)
```

looping



tactile benefits

"building" code - without always staring at a screen

running code - dynamic, physical representation (with adjustable speed) of steps in program

debugging code - physical interaction with pins brings up specific portions of code for further analysis

radical benefits

abstraction/scalability

holistic programming environment:
source code + working surface + output

tangible programming environment:
physical manipulation of code/state