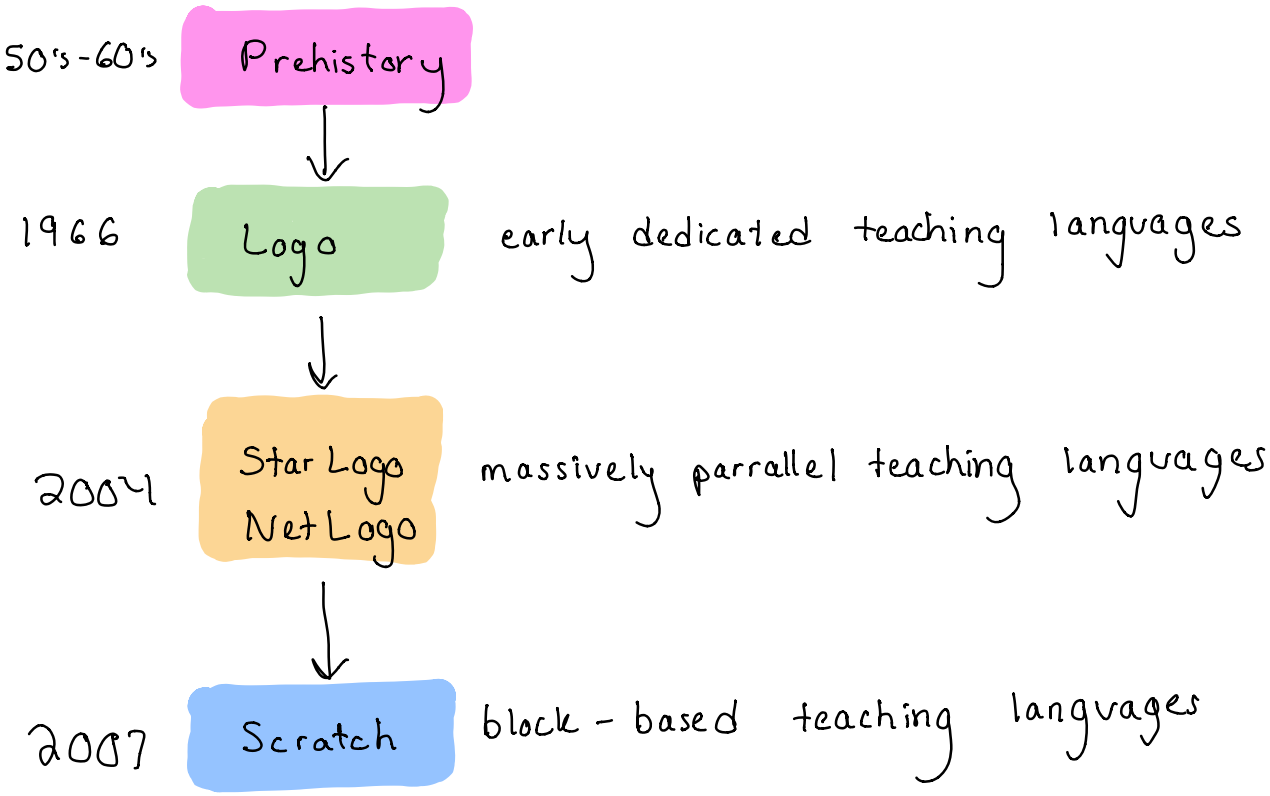
Logo-style teaching languages


50's-60's
Prehistory
1966
Logo
early dedicated teaching languages
2004
Star Logo
Net Logo
massively parrallel teaching languages
2007
Scratch
block-based teaching languages

Prehistory

Sputnik

- first official satellite
- 1957
- launched by Soviet Union
- triggered American Sputnik crisis
- new push for math and science in US schools
- schools trying new methods of teaching during the 60's

Constructivism

- pioneered by Jean Piaget in 50's - 60's

learning is a construction of knowledge rather than a transmission of knowledge

learning is particularly effective when embedded in an activity that the learner feels will construct a meaningful product

Logo

- developed in 1966 by Seymour Papert

manual: Abelson 1974

book: Mindstorms: Children, Computers, and Powerful Ideas by Seymour Papert 1980

history of Logo: Solomon et al. 2020

Seymour Papert

- Jean Piaget's PhD student
- did research on "the theory of learning"
  - ↳ developed Logo in 1966
  - ↳ founded the predecessor to the MIT Media Lab
  - ↳ published research on neural networks in 1969
  - ↳ formalized "constructivism" in 80's

Logo came about to:

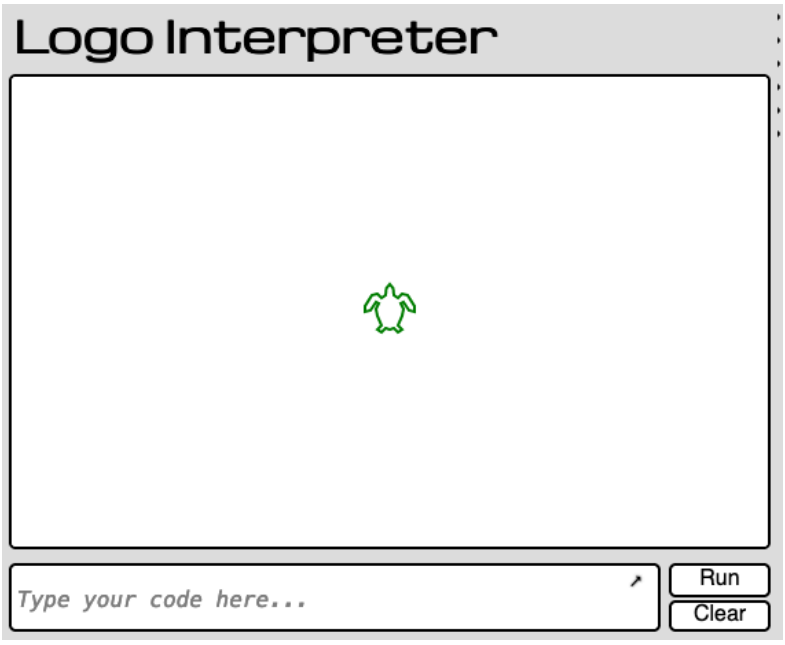
1) teach Lisp-related programming concepts
2) improve the way children think and solve problems
3) create a learning environment where children explore mathematics through projects of their own design

Sputnik

constructivism

What is Logo?

- not just a language ... but a learning environment
- composed of:
  1) turtles (turtle-shaped robots or graphical)
  2) programming language for interfacing with turtles
- no one standard implementation → 300 derivations!
- inspired by early Lisp
- Pascal and BASIC were competitors in education
- Logo was designed to act as a microworld that students could interact with naturally through text
  i.e. a space for thinking about a particular problem



How does Logo compare to Lisp?
How do you program in Logo?

	Lisp	Logo
imperative procedures and commands	✗	✓

- Logo understands primitives
- primitive TO allows you to create procedures
- primitives and procedures take inputs
- imperative commands do stuff
- integers, floating point, and strings are all considered words

```
PRINT "HELLO%WORLD
```

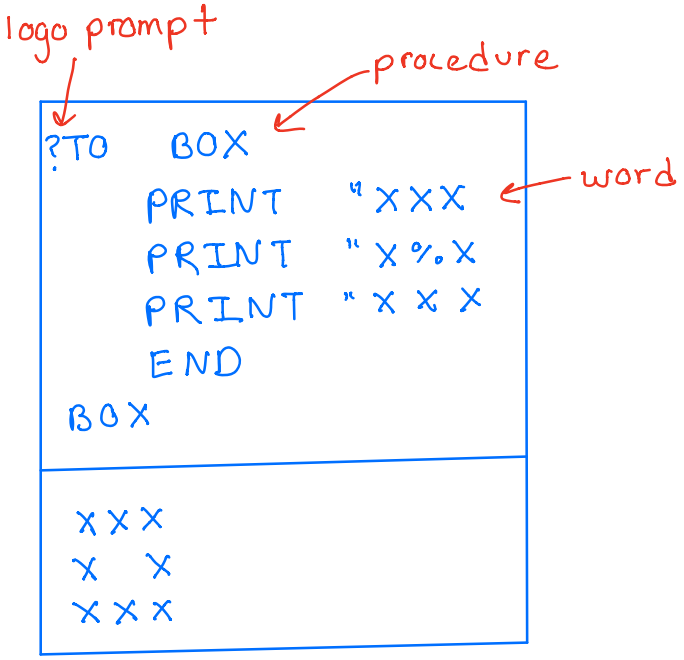
command

input, word

	Lisp	Logo
interactive development	✓	✓

constructivism

- users interact with Logo prompt to see immediate effects, unless they are defining procedures
- primitive TO enters an editing mode where a multi-line procedure can be entered



constructivism

	Lisp	Logo
late name binding	✓	✓

- can reference procedures not yet written

	Lisp	Logo
list - based	✓	✓

- ordered collections of words are lists

```
PRINT (I AM A LIST)
I AM A LIST
```

list

- arrays are collections of integers, floating point, or pointers in one, two, or three dimensions

```
DEFAR "B 2 2 0
```

name

dimension 1

element type (integer)

dimension 2

```
PRINT (GET "B 1 1)
0
```

```
STORE "B 1 1 7
PRINT (GET "B 1 1)
7
```

- set values with make

```
MAKE "X 27
PRINT :X
27
```

	Lisp	Logo
recursion and iteration	✓	✓

```
TO CIRCLE
   FORWARD 5
   RIGHT 1
→  CIRCLE
   END
```

FD == Forward

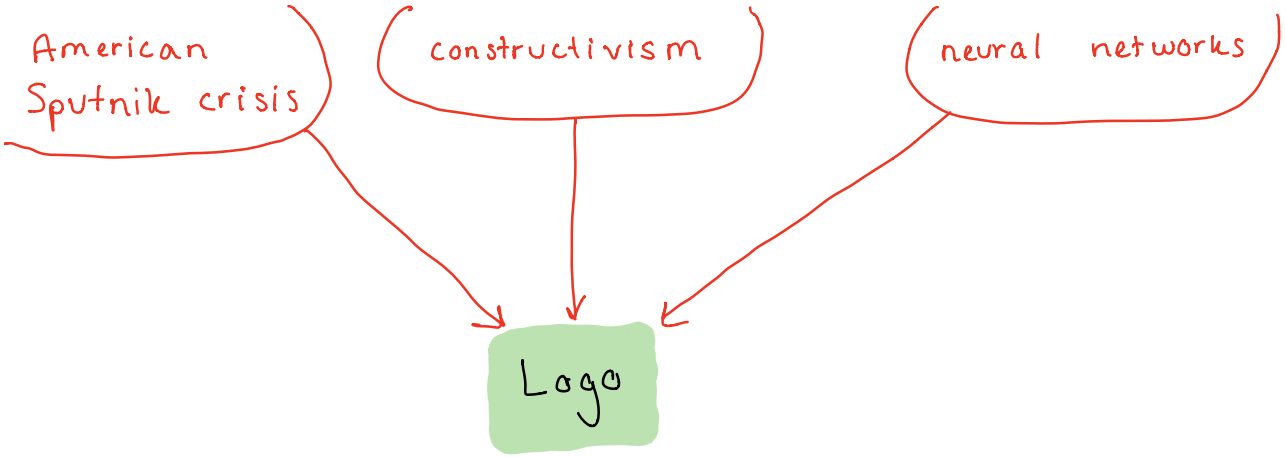
```
TO STAR
→REPEAT 5 [FD 100 RT 144]
  END
```

RT == right

	Lisp	Logo
parenthesis - based	✓	✗
infixed arithmetic	✗	✓
dynamic scope	✓	✓
automatic memory management	✓	✓

pedagogically - focused

Demo


American Sputnik crisis
constructivism
neural networks
Logo

StarLogo

in the late 80's there are developments in:
- neural networks
- cellular automata
- architecture

all of which have to do with **decentralized systems**

decentralized system = system in which the exist no single point of control

For example ... research at the time suggested that bird flocking patterns are a decentralized system (as opposed to having one central leader)

For example ... neural network = decentralized
heuristic-based AI = centralized

For example ... cellular automata = decentralized

decentralized systems can be massively parrallel

Mitchel Resnick
- Seymour Paperts PhD student
- shared constructivist ideas
- **theorized about the centralized mindset**

  people are drawn to imposing centralized control where none is needed
- wanted to test the centralized mindset theory with children
- created StarLogo

Resnick 1992

- StarLogo attempts to provide a clear conceptual model of massive parrallelism and allow users to explore the model
- programming environment similar to Logo

3 classes of objects

1) turtles
   the main inhabitants of the world
2) patches
   "pieces" of the world that turtles live in
3) observer
   looks down on turtles and patches

2 types of communications

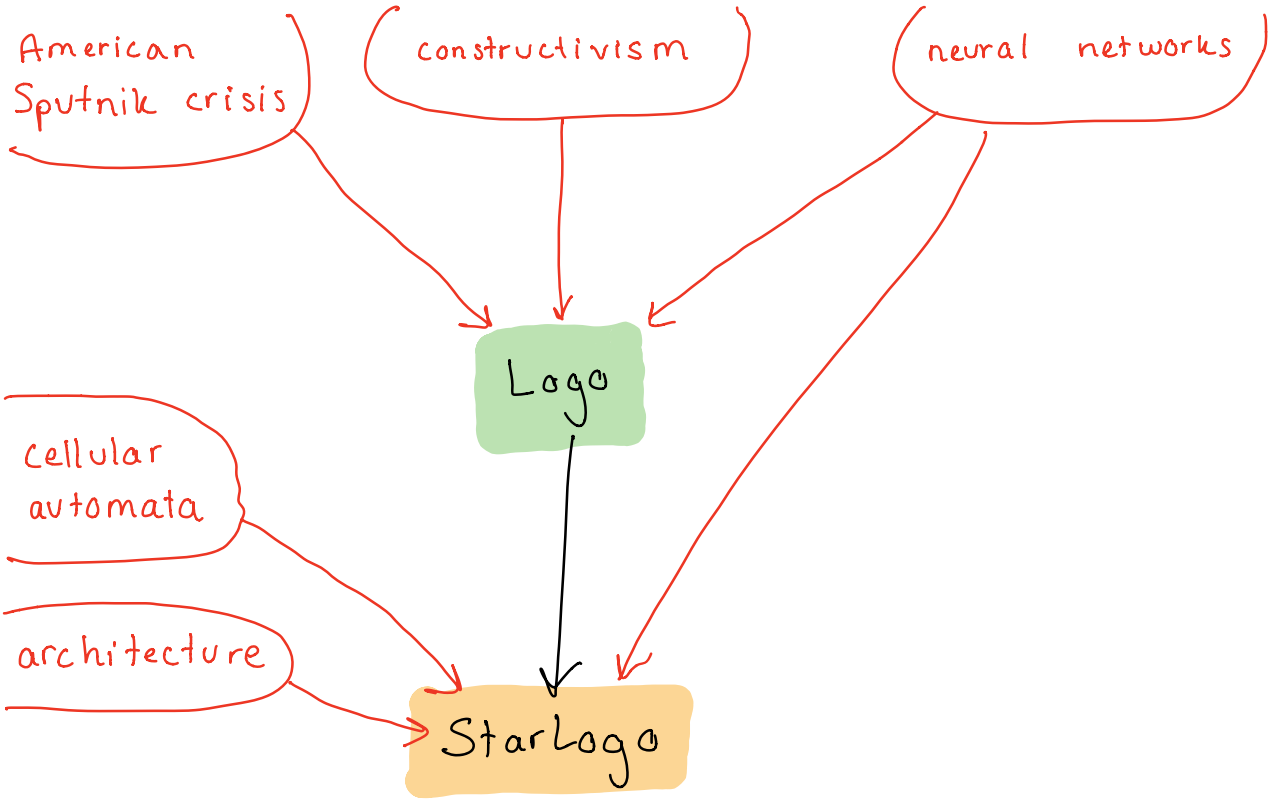
1) demand
   an object can demand another object to perform an action
2) ask
   an object can ask another object for info

Like Logo except:

- StarLogo has a lot more turtles
  5 turtles versus 1000's
  StarLogo is massively parrallel
  this is done to support the goal of supporting massively parrallel simulations
- StarLogo turtles have better "senses"
  Logo turtles are "drawing turtles"
  StarLogo turtles are "behavioral turtles"
  StarLogo turtles can detect turtles and patches nearby
  they can "sniff scents" and "follow the gradient" of a scent
  this is essential to model self-organizing phenomena
- StarLogo reifies the turtles world
  the world is divided into patches that can hold information (as opposed to just being for drawing)
  patches can execute commands

- how does this happen decentralized?
  StarLogo uses background programs that run continuously and simultaneously

Demo? StarLogo may be lost


American Sputnik crisis
constructivism
neural networks
Logo
cellular automata
architecture
StarLogo

NetLogo

- StarLogo was only intended to teach massively-parrallel concepts but it gained popularity in the simulation community

- StarLogo was limited in that it only ran on Macs
- it was designed to be lightweight and subsequently could only handle a limited number of turtles and patches
- math in StarLogo was fixed point, not floating point
- the language design of turtles, patches, and observer ended up being too complicated for users

Tisue and Wilensky 2004

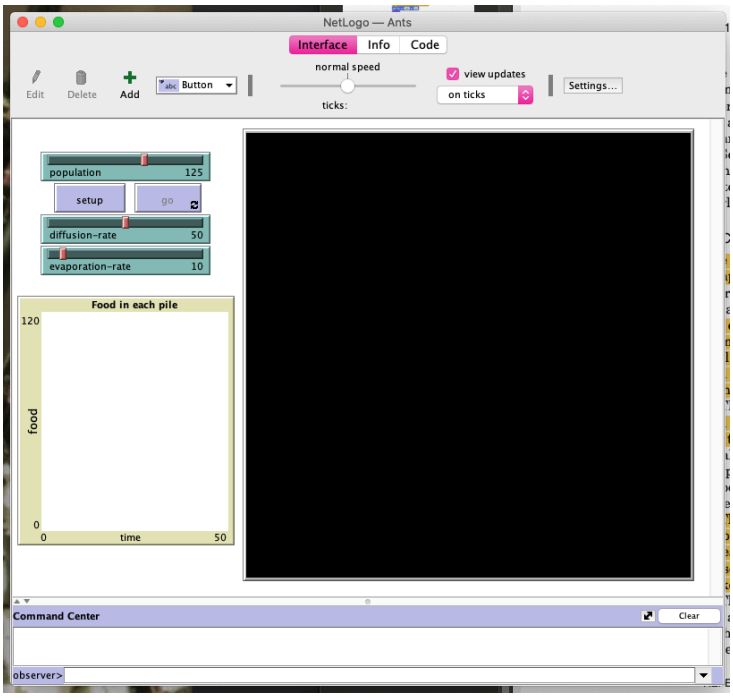
Starting over with NetLogo
- Java was chosen to be cross-platform
- meaning models could be embedded in web pages
- language was redesigned to meet "low threshold" and "no ceiling" goals

"low threshold, no ceiling"
- new users should find it easy to get started
- the language shouldnt be limiting for advanced users (as StarLogo was)

- meant to be equally a research tool and a teaching tool
- provided features encompass StarLogo
- supports massively parrallel simulations

equally a research tool and a teaching tool
- code is easier to write and read
- allows for university overlap between teaching and research

integrated GUI and language
- supports "low threshold"
- smooth transition between exploring models and programming



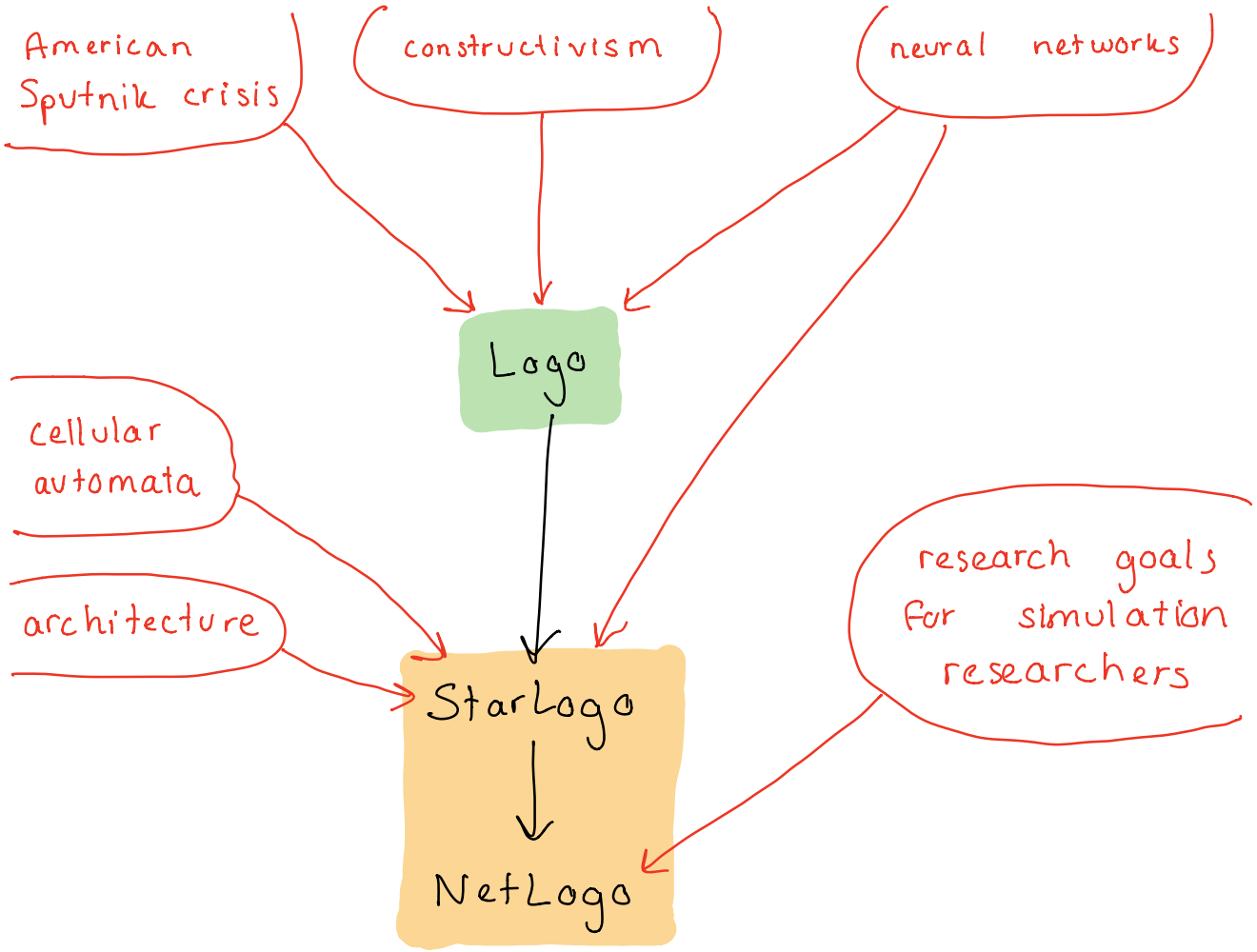
- language embedded in a GUI application
- interface tab = GUI controlling model execution
- procedures tab = holding backend code for model
- expanded the StarLogo environment with...
- includes a tool called BehaviorSpace that does "parameter sweeping" with models
- supports "participatory simulations" in which students can act as agents in the simulation

Language details
  - turtles and patches are the agents
  - all agents can interact with one another
  - there is one observer, who is also an agent
  - the observer "gets the ball rolling" by giving commands to the turtles and patches
  - different "breeds" of turtles may be defined, with different behaviors
- patches are square sections of continuous 2D space
- turtles are floating-point value coordinates

- some models use the patch world as the whole world
- others use both patches and turtles
- others use just turtles

Demo

- NetLogo is widely used in simulation research


American Sputnik crisis
constructivism
neural networks
Logo
cellular automata
architecture
StarLogo
NetLogo
research goals for simulation researchers

Scratch

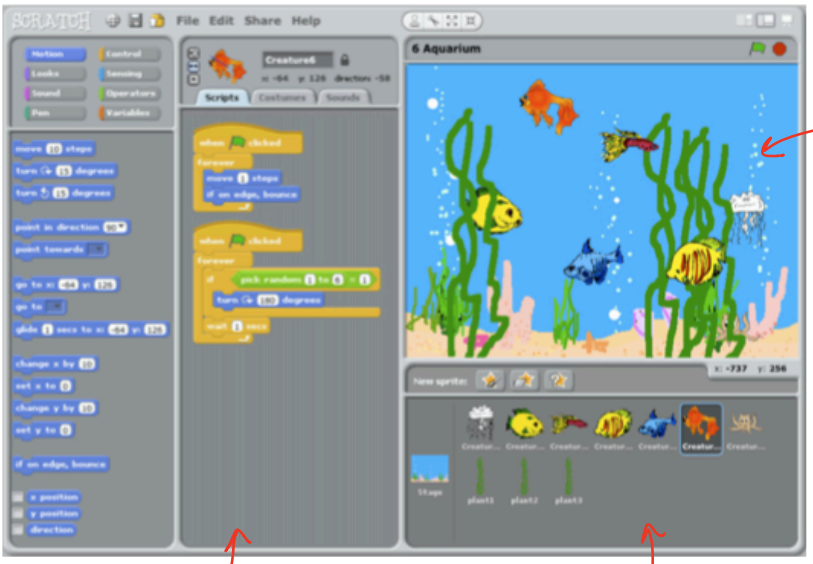
Maloney et al. 2010

- launched in 2007
- at a time when schools trying to get more people into computer science and also starting to require it more in curriculum
- builds on constructivist ideas of Logo
- tries to make projects personally engaging to students by offering the ability to define projects
- goal: introduce programming to those with no previous programming experience
  - ↳ visual blocks language
  - ↳ single window user interface layout
  - ↳ minimal command set
- designed to encourage self directed learning
  - ↳ invite scripting
  - ↳ provide immediate feedback for script execution
  - ↳ make execution and data visible
- aimed at people ages 8-16

- visual programming environment
- users create interactive media-rich projects
- programming is done by snapping together 2D blocks
- blocks control sprites

Single Window User Interface

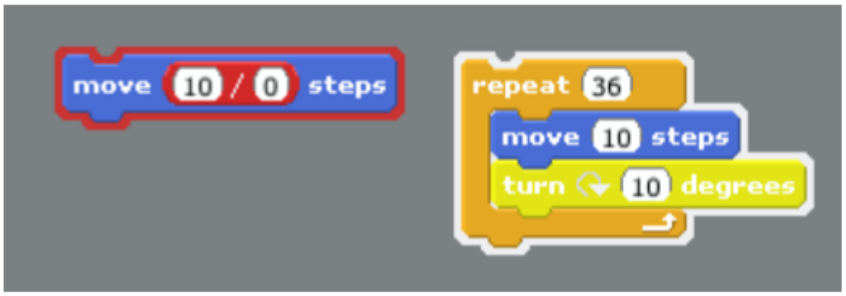
- single window, multi-pane design to ensure key components are always visible



- commands are divided into subcategories: motion, looks, sound, control, etc
- Scratch is always live
  no compilation step or edit/run mode distinction

- helps users stay engaged in testing, debugging, and improving their projects
- Scratch is tinkerable — lets users experiment with commands and code snippets similar to how one might tinker with electrical or mechanical components
- tinkerability helps users discover the functionality of blocks

making execution visible
- if a script encounters an error, it turns red
- normal execution is white



- scratch has no error messages because
  - ↳ blocks only fit together in ways that make sense
  - ↳ scratch tries to eliminate runtime errors by using failsoft = instead of failing with an error message, every block attempts to do something sensible even when presented with out-of-range inputs
- a program that runs feels closer to working than a program that doesn't run

- like Boxer, Scratch turns variables into concrete objects, making them easier to understand through tinkering and observation
- scratch tries to minimize the number of command blocks
- scratch scripts are built by snapping blocks together
- the visual grammar of block shapes and their combination rules play the role of syntax
- 4 kinds of scratch blocks

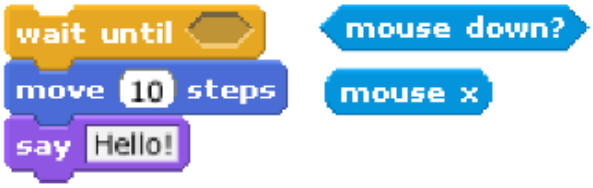
1) command blocks can be joined to create a sequence of commands called a stack

2) function blocks return a value

3) trigger blocks run the tack below it when the trigger event occurs

4) control structure blocks hold nested command sequences

- scratch only allows blocks to be combined in meaningful ways
- command block connects when dropped into command sequence, but a function block will not connect if dropped in the same place
- using the blocks editor feels natural and users often discover how to use it without being told
- 3 data types: boolean, number, and string
- the shape of a parameter slot indicates the expected data type



- sprites are objects that encapsulate state (variables) and behavior (scripts)
- sprites communicate via a one-to-many broadcast mechanism
- any sprite with the matching trigger will activate
- broadcasting is asynchronous
- this is done for a couple reasons
    1) when debugging, the only code controlling a sprite will be in the sprite itself
    2) this allows sprites to be modular and sharable, encouraging collaboration

concurrency

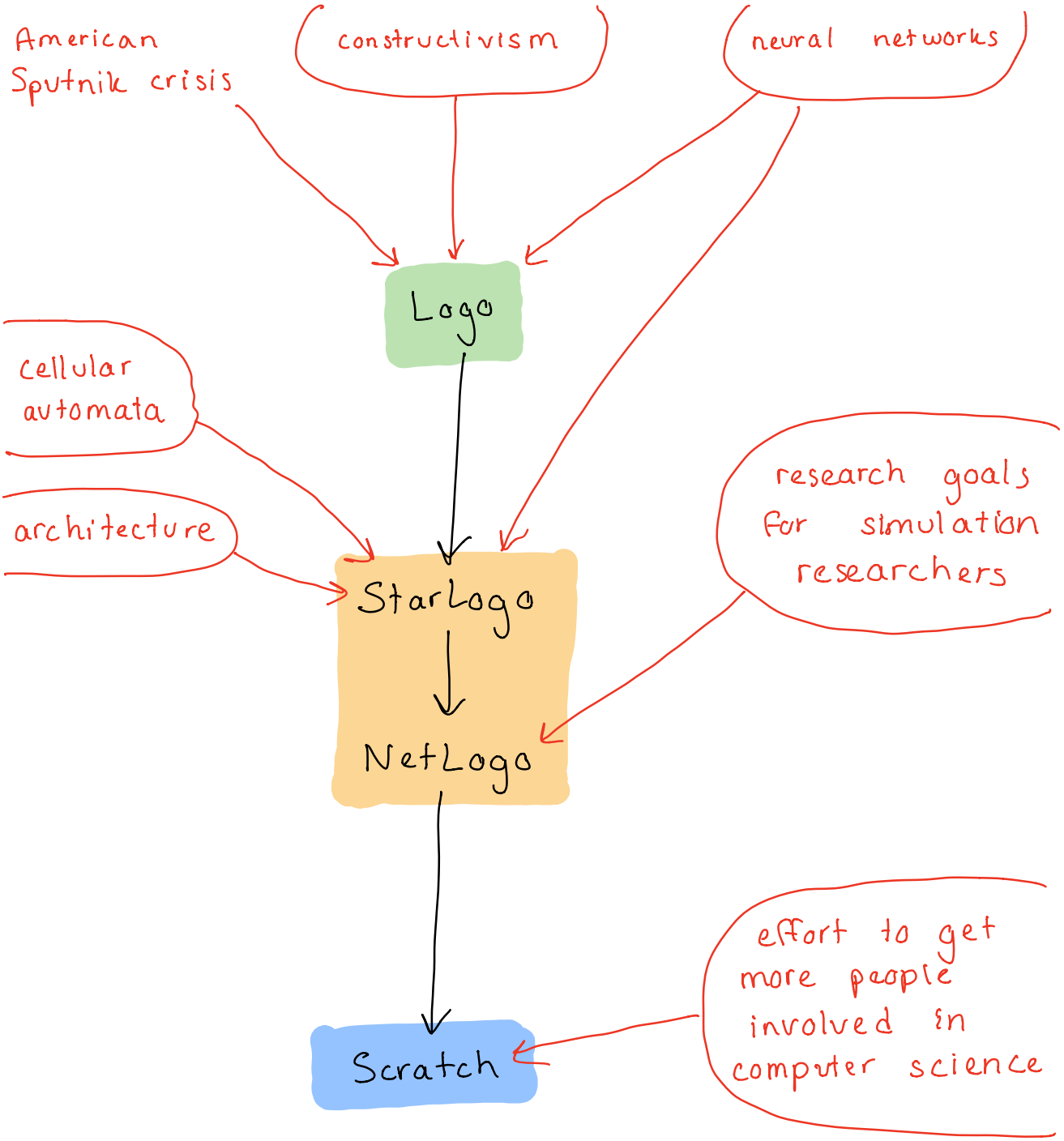
- Scratch lacks explicit concurrency control mechanisms (semaphores, locks, etc)
- builds it into its threading model in a way that avoids race conditions
- in a preemptive threading model, a thread switch can occur between any two instructions

- in the scratch model, a thread switch can occur only
  1) on a command that explicitly waits
  2) at the end of a loop
- the scratch threading model allows users to reason about a script in isolation, giving little consideration to potential side effects from the interleaved execution of other scripts
- still, when multiple scripts are triggered by one broadcast, the ordering of execution may not be what the user expects
- once users understand that scripts are triggered in arbitrary order, they understand the solution (have the event trigger events in order)

Demo

- recent effort to "raise the ceiling" of Scratch by adding higher-order functions

Harvey and Mönig 2010


American Sputnik crisis
constructivism
neural networks
Logo
cellular automata
architecture
StarLogo
NetLogo
research goals for simulation researchers
Scratch
effort to get more people involved in computer science