15-494/694: Cognitive Robotics

Dave Touretzky

Lecture 15:

Code Lab (Scratch 3.0)

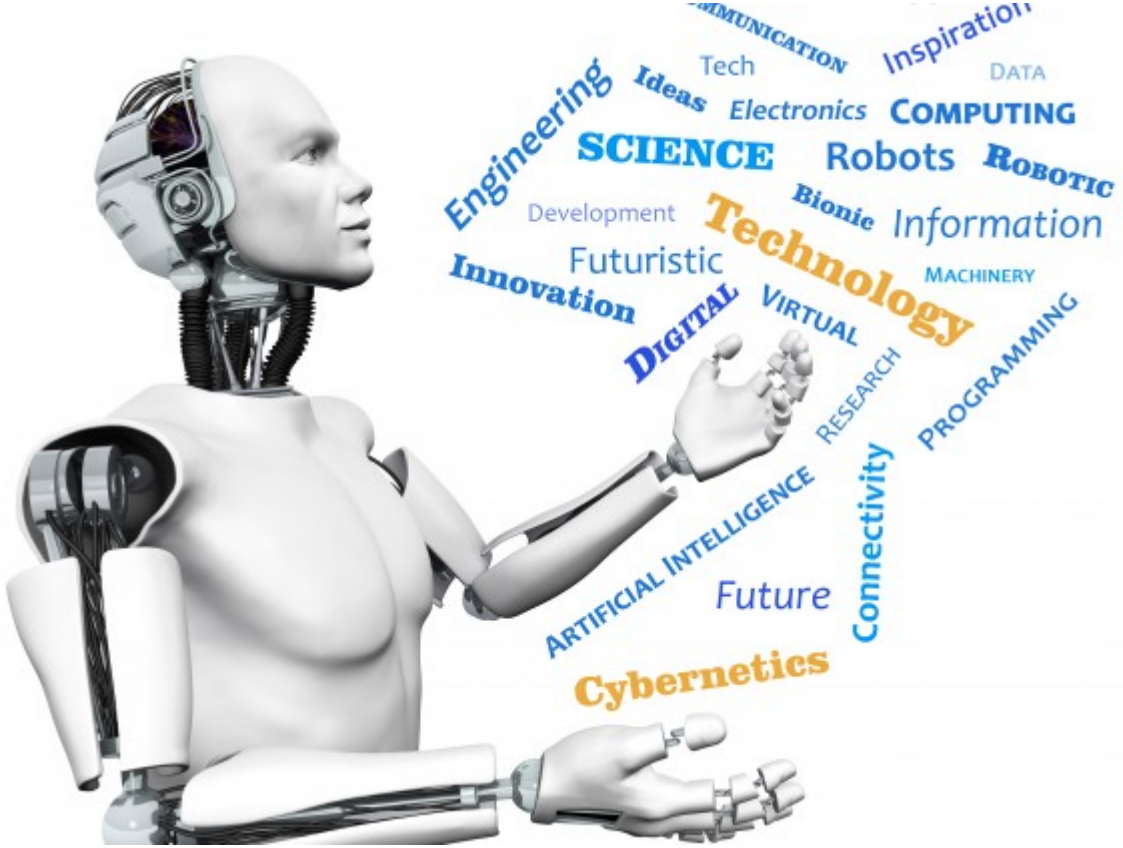


Image from http://www.futuristgerd.com/2015/09/10

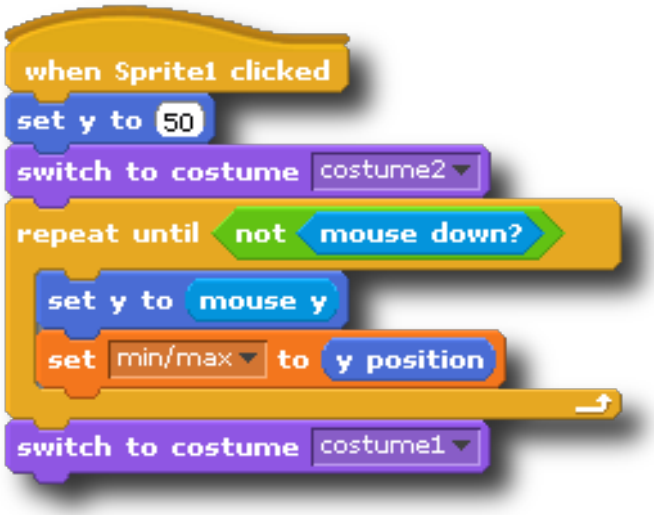
Scratch

- Visual programming language for K-12.
- Created by Mitchel Resnick at MIT and Brian Silverman and Paula Bonta of the Playful Invention Company (Montreal).
- First released in 2003; still evolving.

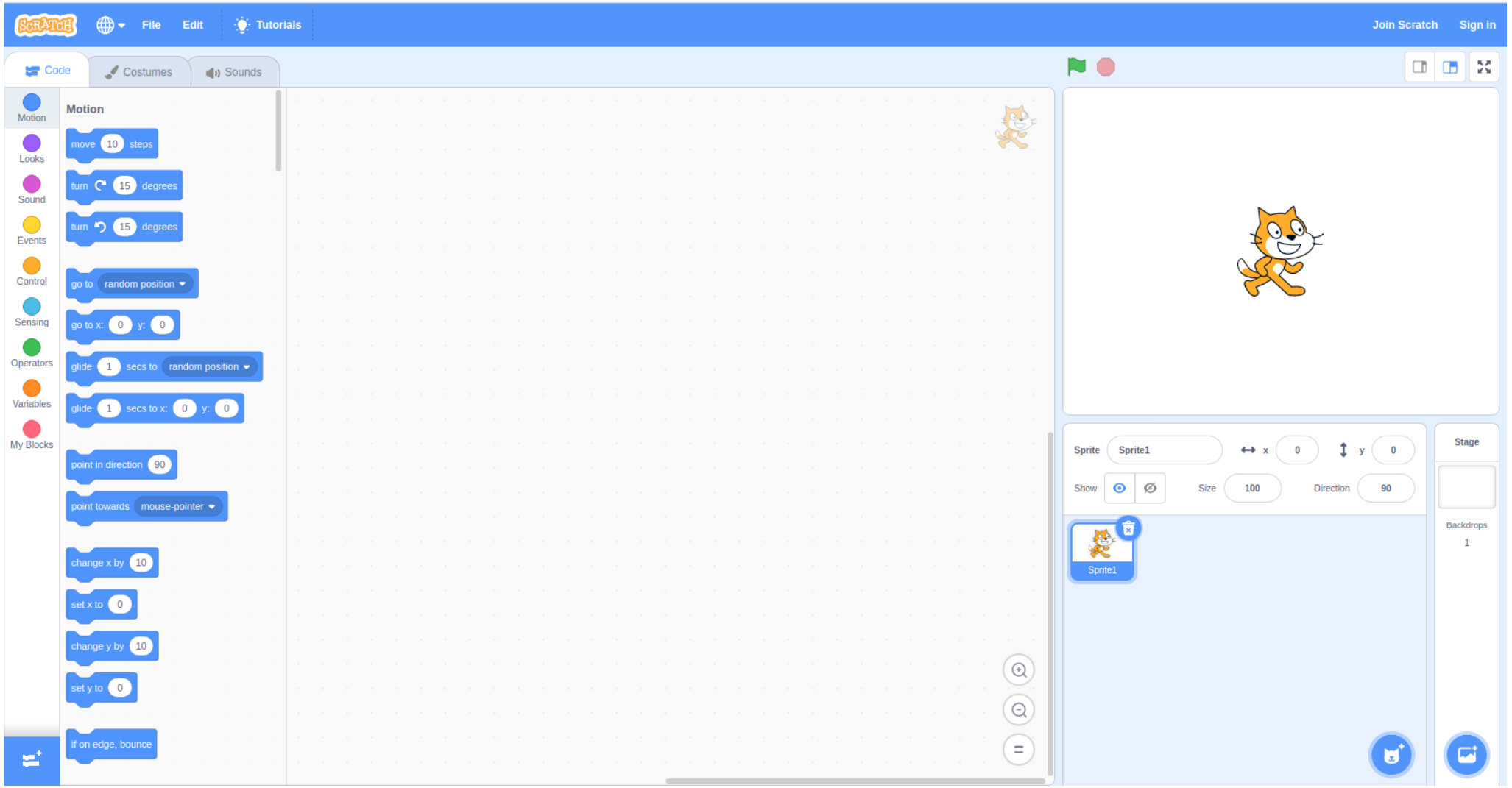
Mitchel Resnick, head of the Lifelong Kindergarten Group at MIT.

Scratch Language

- Drag and drop GUI editor:
  - Blocks organized into semantic categories.
  - Shapes indicate syntactic constraints
  - Prevents syntax errors



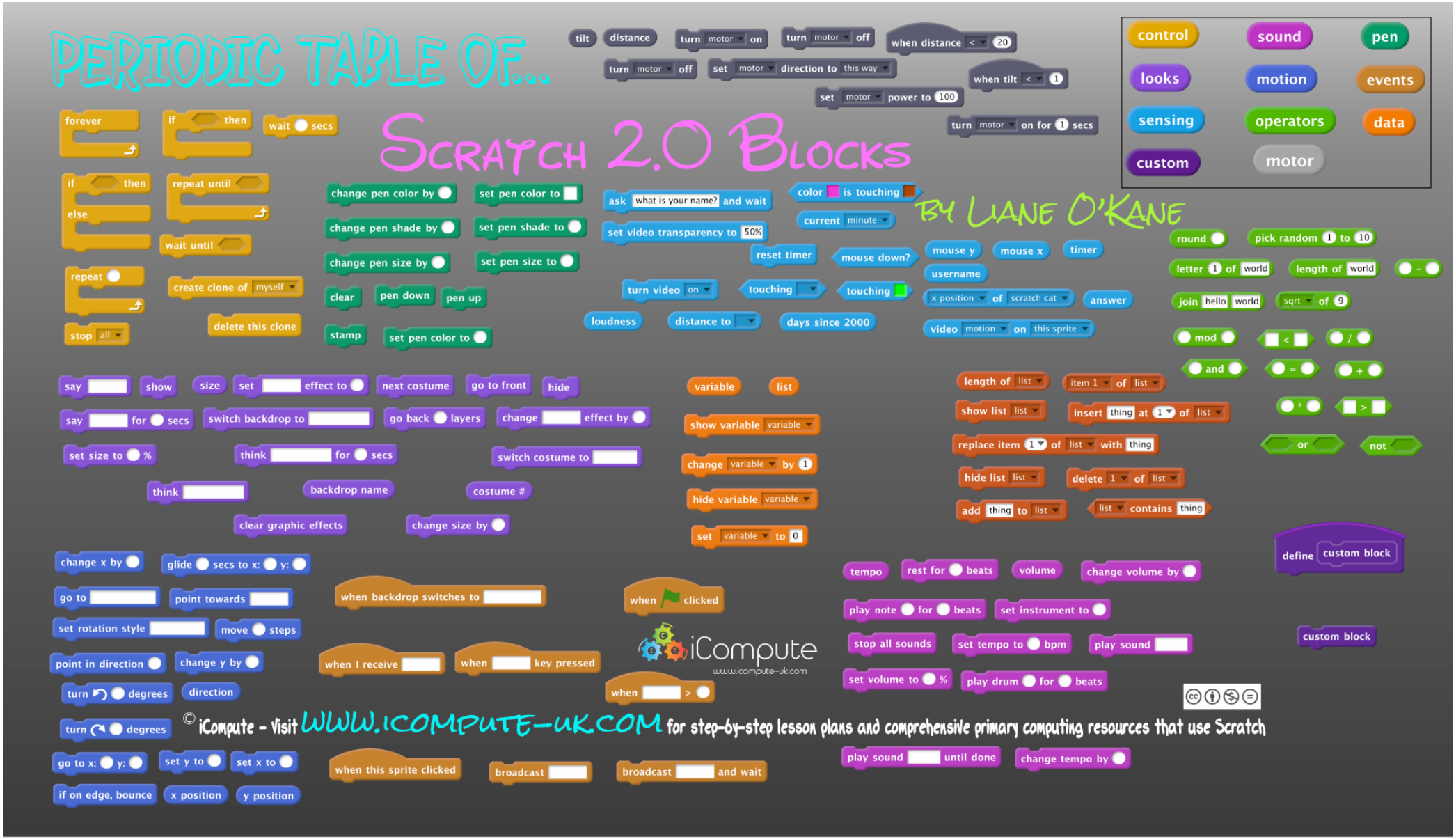
Scratch User Interface



Scratch Semantics

- A graphics programming language
- Characters are called “sprites”:
  - Move in a 2D world
  - Can change appearance (costumes)
  - Can detect contact with other sprites
- Also has turtle graphics drawing functions
  - Pen down, forward, turn, etc.
- Can play sounds, do image manipulations
- Detect “events” such as user input

Scratch Blocks



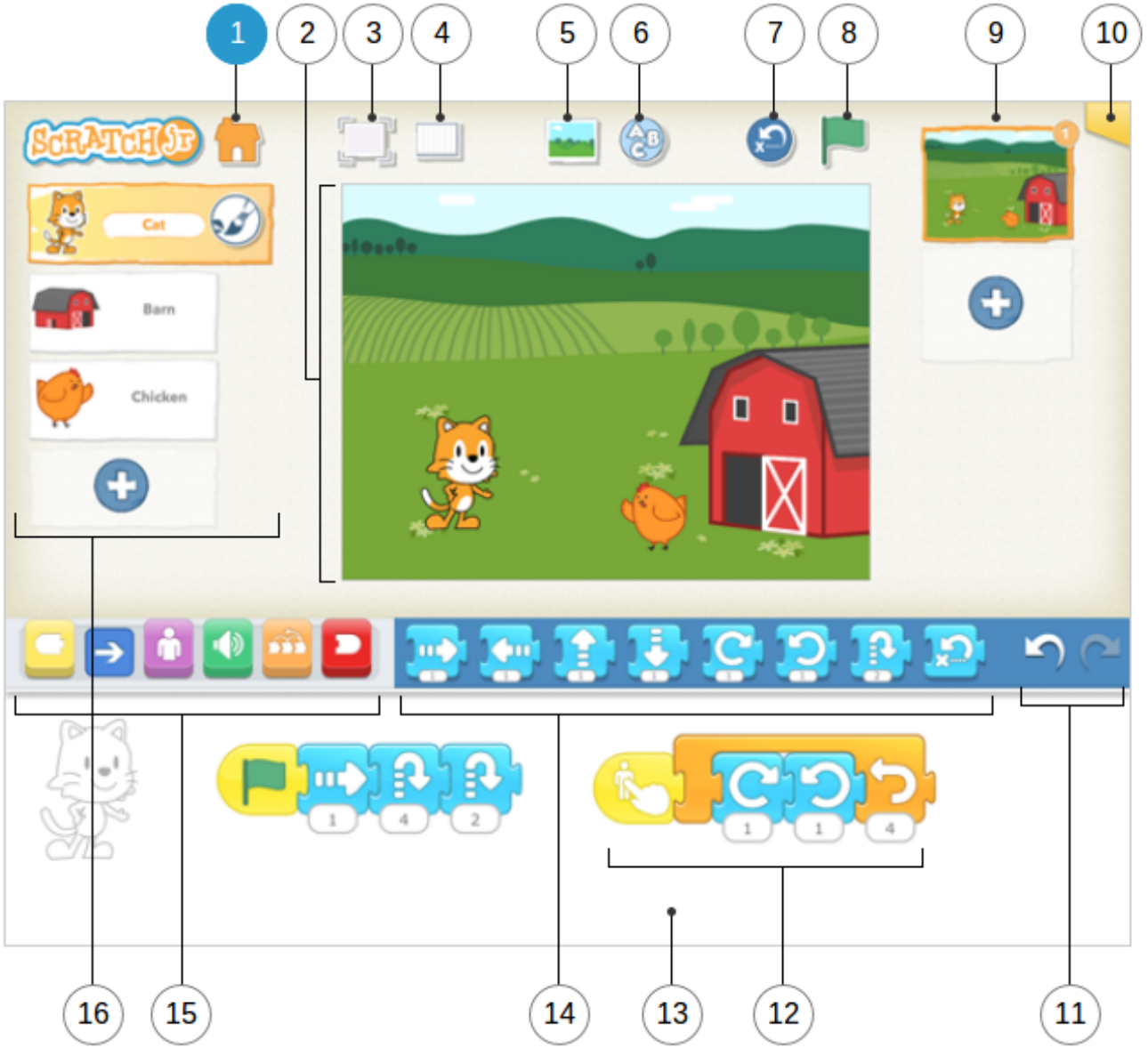
Scratch Jr.

- Developed by Marina Bers at Tufts University, the MIT Lifelong Kindergarten Group, and Playful Invention Co.

Marina Bers

- Linear language for kids ages 5-7:
  - No nesting (except **repeat** block)
  - No conditionals
  - No variables

Scratch Jr. Interface



1 | Save

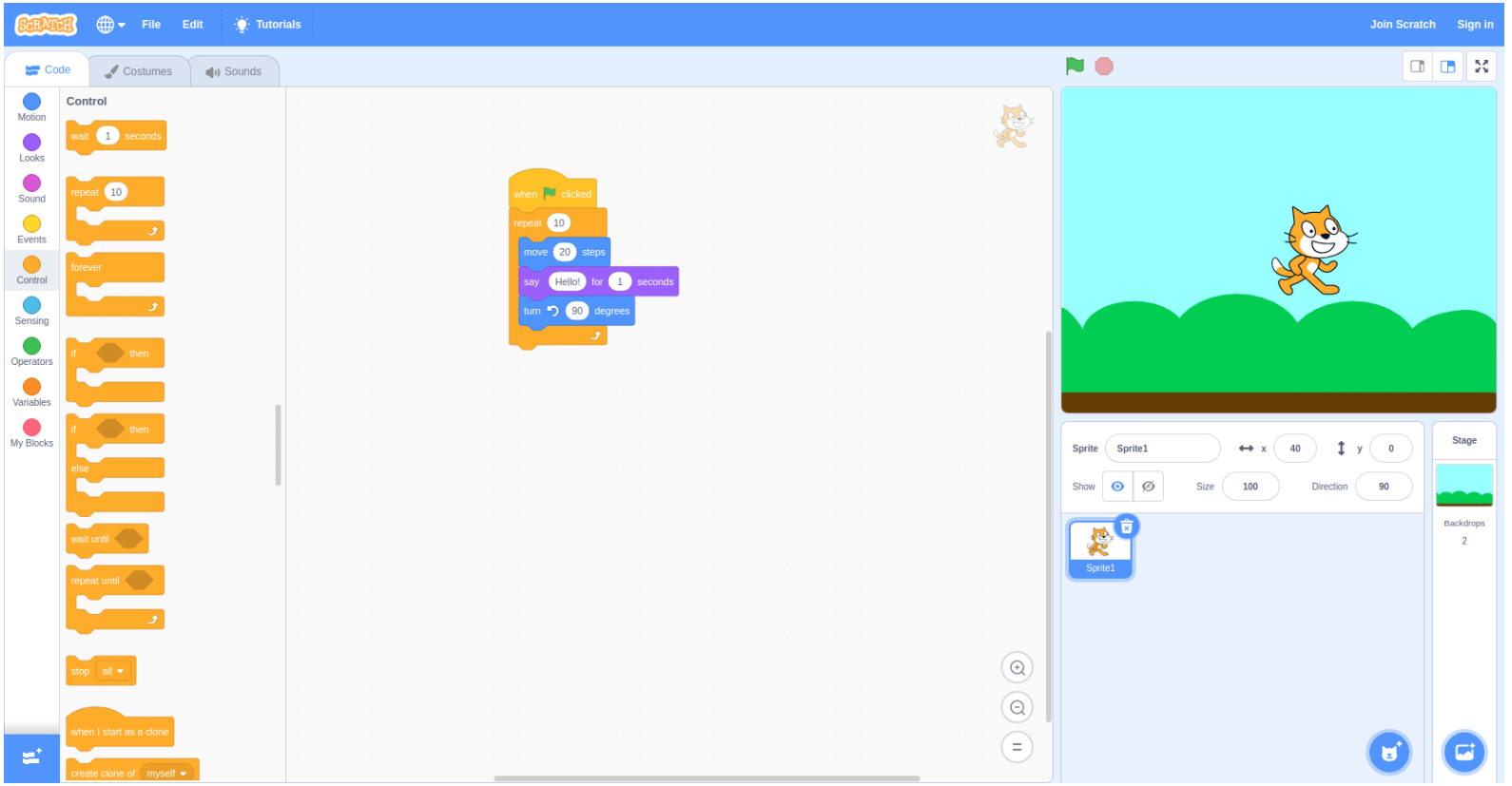
Save the current project and exit to the Home page.

Good Things About Scratch

- Avoids syntax errors.
- Program state is visible:
  - The contents of the 2D “stage”
  - Variable values automatically displayed
- Immediate execution mode aids experimentation.
- “Low floor; high ceiling.”
- Scratch 3.0 runs in the browser and uses HTML5 graphics.

Try Out Scratch 3.0

https://scratch.mit.edu



Limitations of Scratch

- Primitives are too low level
  - Screen coordinates, angles
  - Difficult to reason about program behavior
- No support for state machines.
- Can't operate on collections of sprites.
- No user-defined functions.
- Non-trivial programs are just as tedious as in Python or Java.

Successors to Scratch

- Snap (from UC Berkeley) extends Scratch:
  - User-defined functions
  - Lambda expressions; closures
  - Arrays
  - Object-oriented sprite hierarchy
- Blockly
  - Generic blocks-based language created by Google
  - Used as the basis for many other blocks-based programming frameworks

Code Lab


Do more with Cozmo than ever before.
Sandbox Mode
The best place to start coding.
Simply snap together code blocks to create your first project! Complete challenges to grow your skills.
SEE HOW IT WORKS »
Constructor Mode
Take your coding skills to the next level.
With concepts like branching and variables, create complex projects with endless possibilities.
SEE SAMPLE PROJECTS »

Code Lab : Scratch

- Code Lab is built on Scratch 3.0
- Runs inside the Cozmo app
- Primitives mirror those of the Python SDK:
  - Detect cubes, faces, cube taps
  - Manipulate cubes
  - Play sounds and animations

Sandbox Mode

Drive

You control which direction Cozmo goes and how fast he gets there. Add how far with a simple drop down.

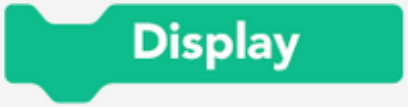
Actions

Use his lift, control his head, and light up his backpack. You can even command him to speak short phrases.

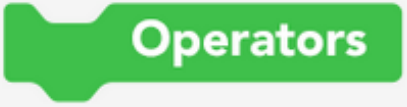
Constructor Mode

These blocks control the flow of a project in a desired fashion, for example by providing functions for looping various blocks and scripts. They control the project and enhance its running.

Sensors blocks link with Cozmo's sensors to provide information about his world. For example, they provide data about his physical location, what he's seeing, if his Cubes were moved, and even the orientation of your mobile device.



Display blocks allow you to draw images on Cozmo's face screen.

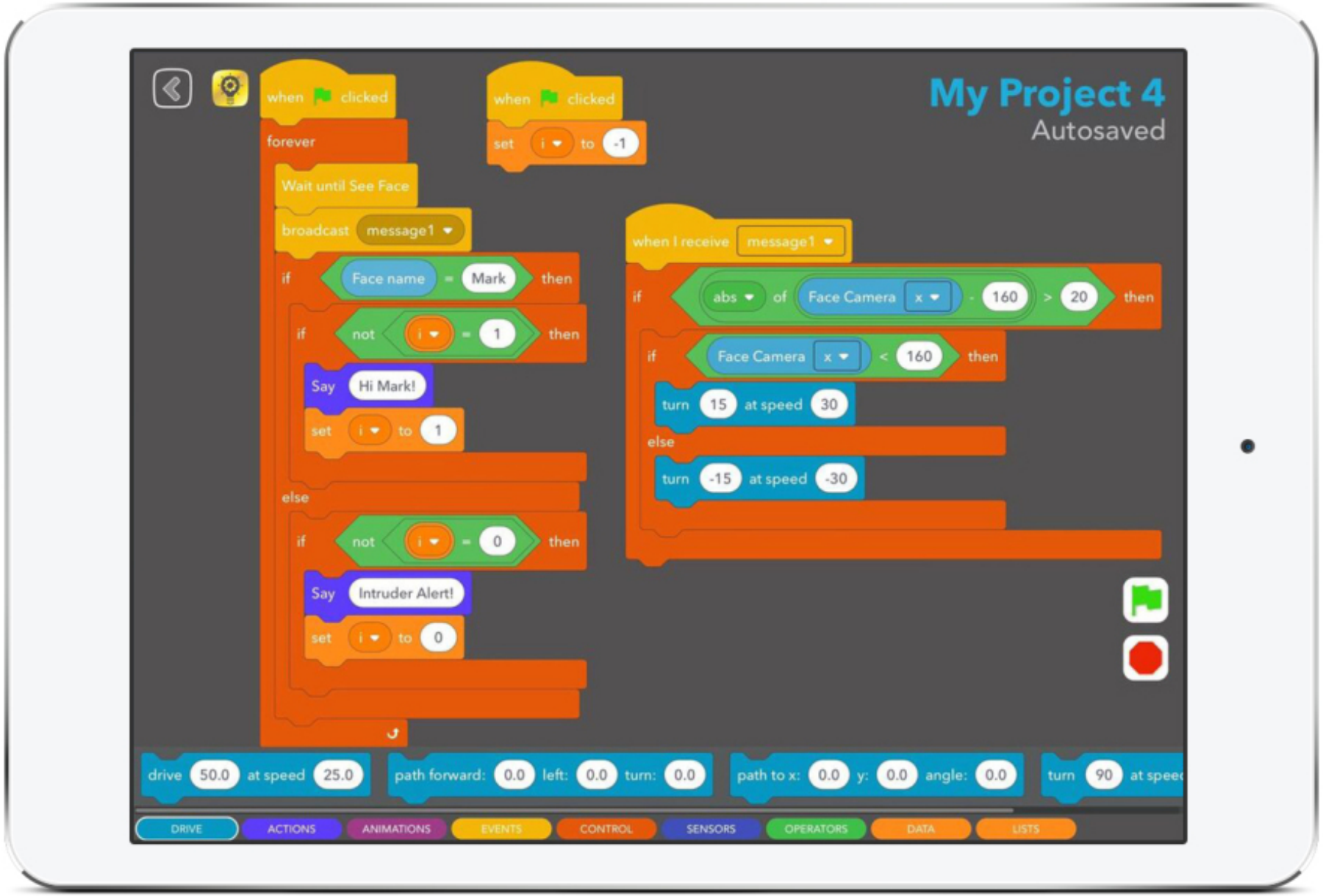


Operators blocks deal with simple and complex mathematical functions within a project. They provide capabilities of simple to complex mathematical operations, as well as allowing you to manipulate and use strings.

Data blocks are related to storing and accessing information. This category can be used to create variables, which can be used to store data, such as the score in a game, and use it to control how the program behaves.

Low-Level Control

Constructor Mode Project



Drive

- drive [X] mm at [y] mm/s
- turn [X]º at [Y] º/s
- drive left wheel [X] mm/s, right [Y] mm/s
- stop [wheels | head | lift | all]
- dock with Cube [#]
- navigate [X] forward, [Y] to side, [Z] º
- navigate in world to [X] x, [Y] y, [Z] º

Actions

- say [text]
- move lift [X] °/s
- move lift to [X] % at [Y] °/sec
- move head to [X]° at [Y] °/sec
- set backpack light [color]
- on Cube [#] set [all lights | light #] to [color]
- on cube [#] [spin | blink] lights in [color]

Animations

- play [animation name] animation
- enable: [wheels | head | lift] in animations
- disable: [wheels | head | lift] in animations
- play [sound name] sound
- play [sound name] sound and wait
- stop [sound name] sound
- play [text] SDK animation
- play [text] SDK animation group

Events

- when  clicked
- when Cube [#] tapped
- when Cube [#] moved
- when face seen
- when [happy | sad] face seen
- when [message#] received
- broadcast [message#]
- broadcast [message#] and wait to complete

Control (1/2)

- wait [X] secs
- repeat [X]
- forever
- if (condition) then [A]
- if (condition) then [A] else [B]
- wait until (condition)
- repeat until (condition)

Control (2/2)

- stop [all | this script]
- stop [driving | moving head | moving lift | animations | saying text | everything]
- wait for Cozmo to finish [...]
- enable: always wait for Cozmo to finish
- disable: always wait for Cozmo to finish

Sensors (1/3)

- Cozmo lift height %
- Cozmo head angle º
- is Cozmo picked up
- Cozmo [pitch | roll | yaw] º
- Cozmo position [X | Y | Z] in world

Sensors (2/3)

- is face visible
- face expression
- face name
- face position in camera [X | Y]
- face position in world [X | Y | Z]

Sensors (3/3)

- was Cube [#] tapped
- last tapped Cube
- is Cube [#] visible
- Cube [#] [pitch | roll | yaw] º in camera
- Cube [#] position in camera [X | Y]
- Cube [#] position in world [X | Y | Z]
- device [pitch | roll | yaw] º
- current [year | month | date | day-of-week | hour | minute | second]

Display (1/2)

- display on Cozmo's face
- clear all pixels
- draw [text] at [X],[Y]
- set text scale to [X] %
- set text alignment to [top | center | bottom] [left | center | right]
- draw line from [X],[Y] to [X2],[Y2]
- draw rectangle from [X],[Y] to [X2],[Y2]
- fill rectangle from [X],[Y] to [X2],[Y2]

Display (2/2)

- draw circle at [X],[Y] with radius [Z]
- fill circle at [X],[Y] with radius [Z]
- set drawing mode to [draw | erase] pixels

Operators (1/2)

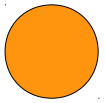
- Arithmetic: + - * /
- Comparison: < = >
- Boolean: **and** **or** **not**

Boolean valued (hexagon shape)

- pick random [X] to [Y]
- [X] mod [Y]
- round [X]
- Math: abs, floor, ceiling, sqrt, sin, cos, tan, asin, acos, atan, ln, log, e^, 10^

Operators (2/2)

- Join [text1] and [text2]
- letter [#] of [text]
- length of [text]
- [text1] contains [text2]



Data

- [Create variable ...]
- set [variable] to [X]
- change [variable] by [X]

Assessment of Code Lab

- Easier than Python SDK:
  - Runs directly in the app: no USB cables!
  - No syntax errors
- What's missing?
  - No camera viewer
  - No world map
  - No state machine support
  - No way to detect failed actions
  - No high level primitives (grab, roll, stack)