Code.org curricula (Blockly-based)

Andrea Sterbini – sterbini@di.uniroma1.it

Code.org

Built with Blockly: a JavaScript library for visual languages

Code.org (and AppInventor.mit.edu)

Fine-grained activities with CONSTRAINED activities

(initially less freedom … later full environment)

Initial language

NO local variables

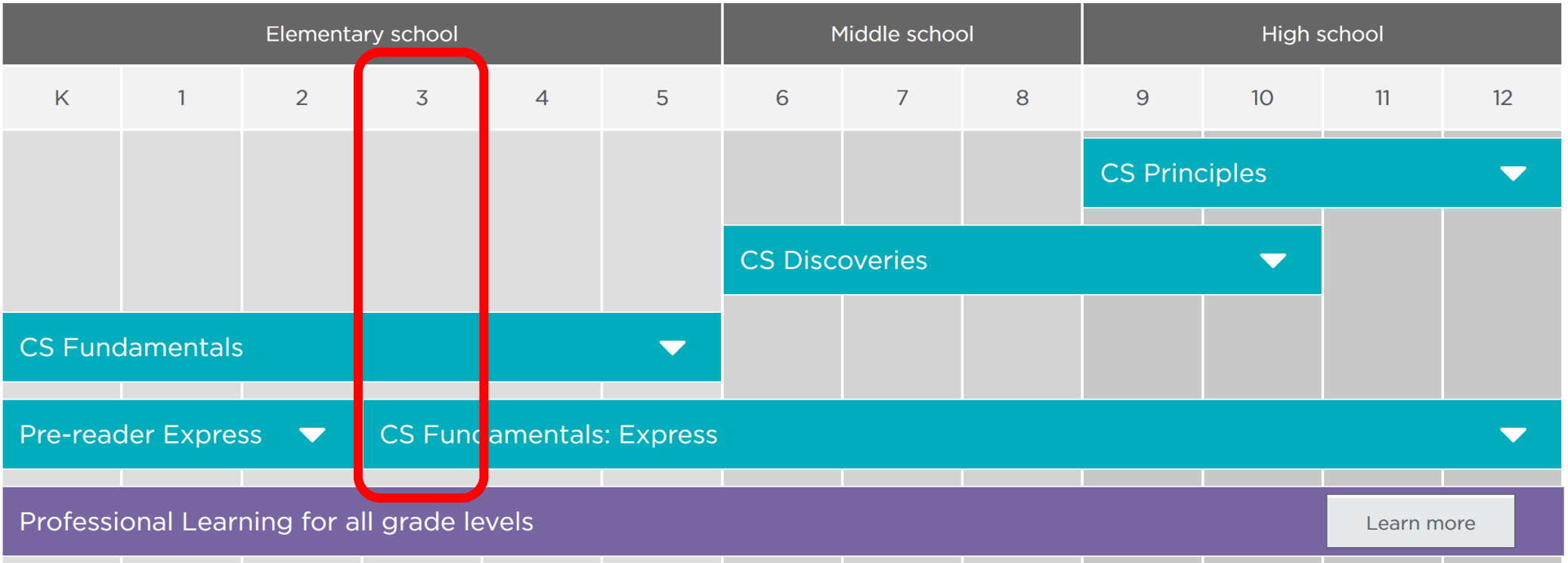
NO personal agent attributes

Procedures (NO return value)

Possibility of data type enforcement

Puzzle-like connectors with different shapes: Actors, numbers, text, booleans

Complete curriculum from Elementary to High school



A course tailored to students of each year:

E.g. Course D for 3rd grade (K3): **algorithms, nested loops, while loops, conditionals, and events**. Beyond coding, students learn about **digital citizenship**.

Both “unplugged” and programming activities

Course D for 3rd grade (K3 = 8 y old)

Lesson 1: Graph Paper Programming ***(Unplugged)***

In this lesson, you will program your friend to draw pictures!

Lesson 2: Introduction to Online Puzzles **(Sequencing | Debugging | Loops | Angry Bird | Collector | Artist | Harvester)**

This lesson will give you practice in the skills you will need for this course.

Lesson 3: Relay Programming ***(Unplugged | Relay Programming | Algorithms)***

Remember at the beginning of the course when you made drawings with code? In this lesson, you will be working with a team to do something very similar!

Lesson 4: Debugging with Laurel **(Debugging | Bug | Collector | Laurel)**

Have you ever run into problems while coding? In this lesson, you will learn about the secrets of debugging. Debugging is the process of finding and fixing problems in your code.

Lesson 5: Events in Bounce **(Event | Bounce)**

Ever wish you could play video games in school? In this lesson, you will get to make your own!

...continue Course D for 3rd grade (K3 = 8 y old)

Lesson 6: Loops in Ice Age **(Loop | Ice Age | Scrat)**

Lesson 7: Drawing Shapes with Loops **(Loop | Artist)**

In this lesson, loops make it easy to make even cooler images with Artist!

Lesson 8: Nested Loops in Maze **(Nested Loops | Loops | Bee | Maze)**

Loops inside loops inside loops. What does this mean? This lesson will teach you what happens when you place a loop inside another loop

Lesson 9: Fancy Shapes using Nested Loops **(Nested Loops | Loops | Artist)**

More nested loops! This time, you get to make some AMAZING drawing with nested loops.

Lesson 10: Snowflakes with Anna and Elsa **(Loop | Nested Loop | Frozen)**

Anna and Elsa have excellent ice-skating skills, but need your help to create patterns in the ice. Use nested loops to create something super COOL.

...continue Course D for 3rd grade (K3 = 8 y old)

Lesson 11: While Loops in Farmer (While Loops | Loops | Farmer)

Loops are so useful in coding. This lesson will teach you about a new kind of loop: while loops!

Lesson 12: Until Loops in Maze (Conditional | Loop | Maze | Angry Bird | Zombie)

You can do some amazing things when you use `until` loops!

Lesson 13: Conditionals with Cards (Conditionals | Unplugged)

It's time to play a game where you earn points only under certain conditions!

Lesson 14: If/Else with Bee (Conditional | Bee | Maze)

Now that you understand conditionals, it's time to program Bee to use them when collecting honey and nectar.

Lesson 15: Harvesting with Conditionals (Conditional | Loop | Harvester)

It's not always clear when to use each conditional. This lesson will help you get practice deciding what to do.

...continue Course D for 3rd grade (K3 = 8 y old)

Lesson 16: Digital Citizenship (Common Sense Edu. | Unplugged)

Some information is not safe to share online. This lesson will help you learn the difference betweensafe and private information.

Lesson 17: Ninjas vs. Pirates Game (Play Lab | Event)

This lesson will guide you through making your very own video game.

Lesson 18: Binary Images (Binary | Unplugged)

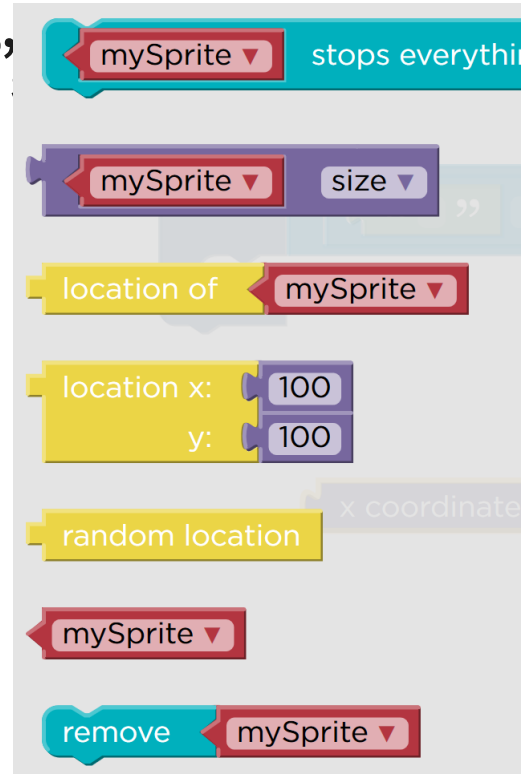
Learn how computers store pictures using simple ideas like on and off.

Lesson 19: Binary Images with Artist (Binary | Artist)

In this lesson, you will learn how to make images using on and off

Visual language
User interaction and common features

Visual choosers to simplify input: Sprite's "costumes" colours, angles, positions, sound/music, …

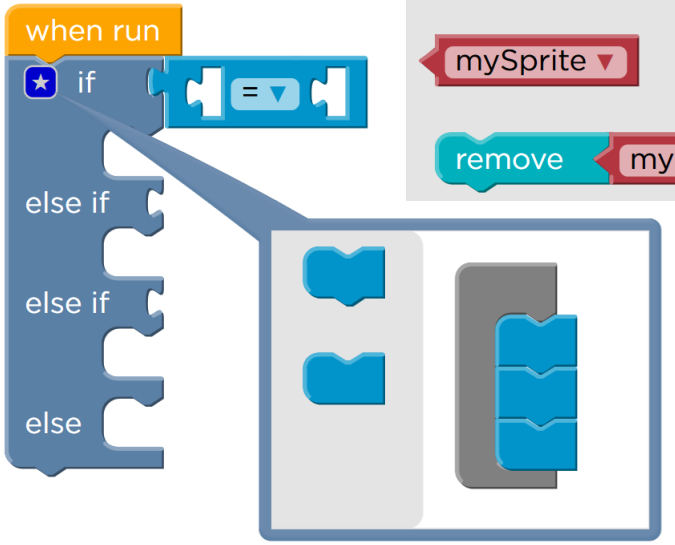


Typed connectors: positions, sprites, numbers, conditions, text

Extensible if (if, elif, elif, …, else)

Counted loops (with counter)

Show corresponding JavaScript code



Made with Blockly

A JavaScript library to build visual languages

Easy way to define new types of blocks with:

- Typed inputs (int, string, object, list, boolean, …)
- Typed outputs
- Conversion of the resulting code to many programming languages (JavaScript by default, but also Lua, Python, Dart, …)

The resulting JavaScript can be evaluated to interact with the page

- Labyrinths, Harvesting robots, Games, Simulations, …

Used in: code.org, appinventor.mit.edu, programmailfuturo.it, …

Simple constrained environments: Initial classes

DEMO

Many environments:
Sprite Lab: multiple interacting Actors

Single initial thread of execution (e.g. to create Sprites)

(Multiple) actors reacting to simple events (but NO messages)

Concurrent execution of multiple different events

Multiple threads for the same event

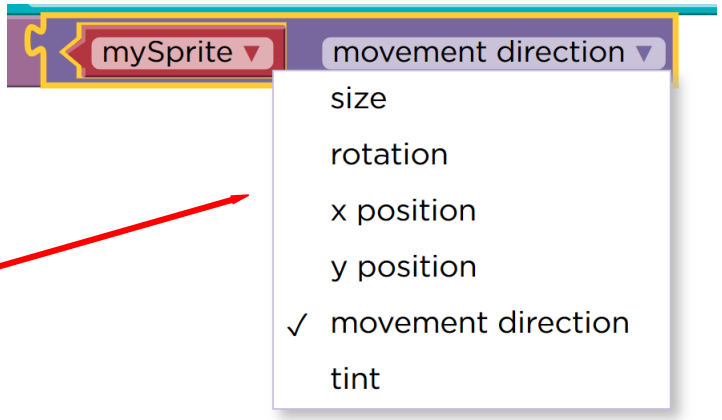
Simple Procedures (without parameters)

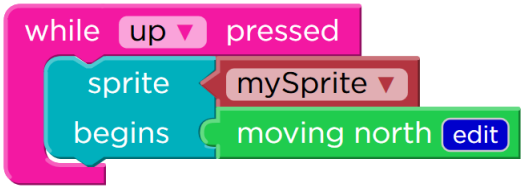
Simple methods (“behaviors”)

Fixed Sprite properties

Global variables

NO lists





Many environments:

Artist: turtle graphics

Single thread of execution

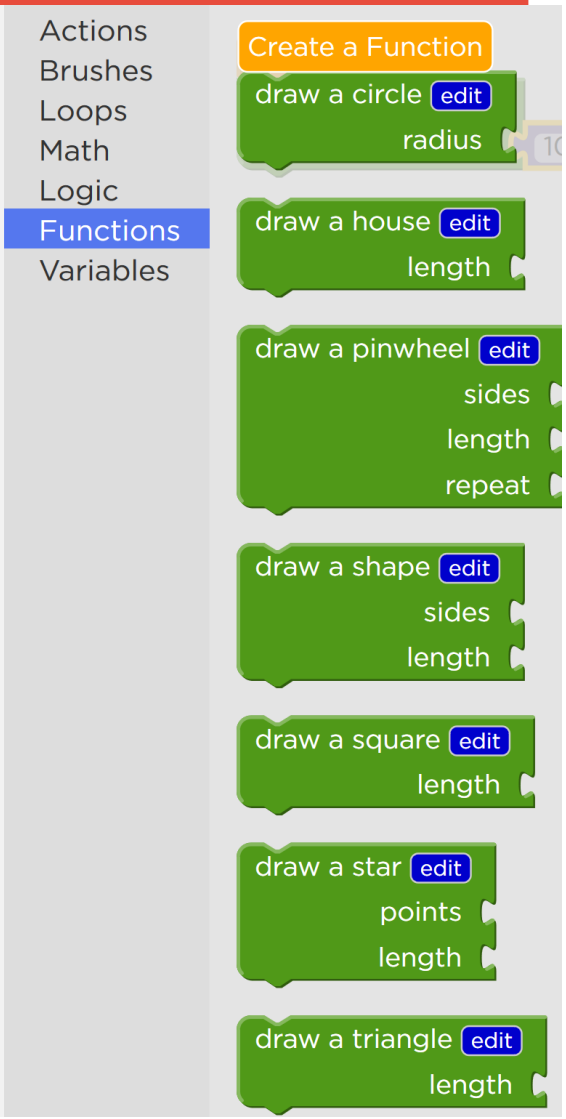
Single agent (Pen)

New: PARAMETRIC procedures

Automatic redraw/run when parameters change

Examples of editable procedures/drawings

NO concurrency



Many environments:

App Lab: build a "phone-like" app

Graphic editing of the App GUI (buttons, fields, labels, …)

Setters/getters of all App widgets properties

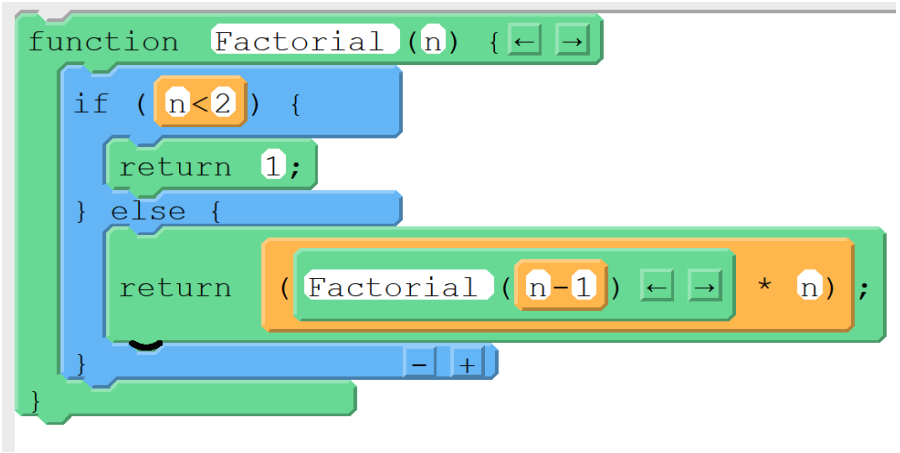
Full JavaScript-like visual syntax

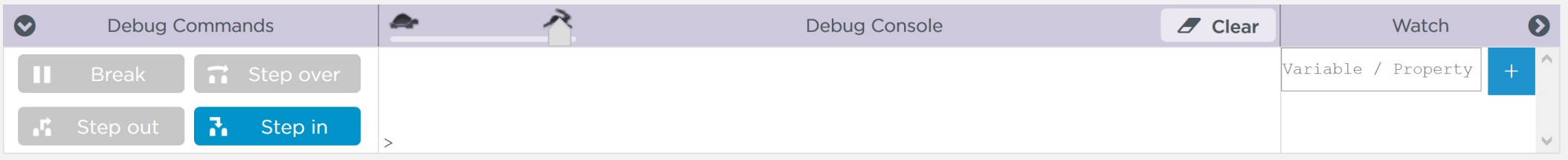
Full functions (args, local vars, return)

DATA store (dictionary OR tables)

Turtle graphics and Canvas

New: DEBUGGER





App Lab Events

Events:

GUI: onEvent(widgetId, event, callback)

Data: onRecordEvent(table, callback(record, event))

Timers: setTimeout(ms, callback)

timedLoop(ms, callback)

Callback functions