Scratch.mit.edu

Andrea Sterbini – sterbini@di.uniroma1.it

Scratch.mit.edu

structured visual programming (no GOTO)

Visual code editor with blocks (NO syntax, almost no typing)

Web-based or local visual editor (https://scratch.mit.edu/download)

Blocks contain text/commands (not OK for pre-scholar students)

Available data types

Numbers, strings, booleans, simple lists (heterogeneous)

Main features

GLOBAL variables + Agent variables + Agent cloning

Procedures: "My blocks" (NO return value! BUT: you can simulate it with a variable)

PARALLEL execution of multiple scripts for the same event!

Message based coordination and synchronization

Event-based programming: (touched, hit, key pressed, message received)

Blocks Categories

Motion: move the agent (e.g. the Cat)

Looks: change agent's appearance

Sound: produces/plays sound

Events: definition of callbacks to be executed on events

Control: if-then-else, conditional/counted loops, ...

Sensing: reading attributes or ask for input

Operators: mathematical/logical operations

Variables: variable definition/getter/setter/increment

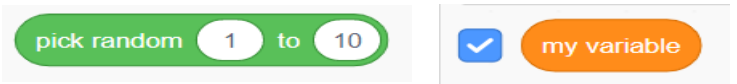
Lists: list definition/getter/setter and manipulation

Blocks Shapes
(see https://en.scratch-wiki.info/wiki/Blocks)

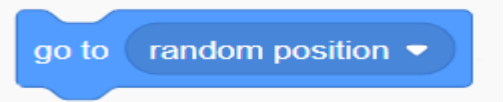
HAT: event handlers (definition/start)

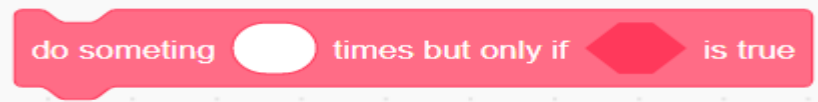
REPORTER: function results/variable values/agent attributes (numbers, lists or strings)

ANGULAR: boolean values/operators/parameters

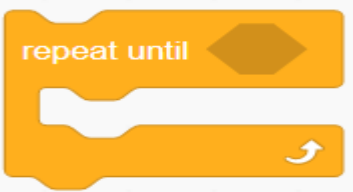


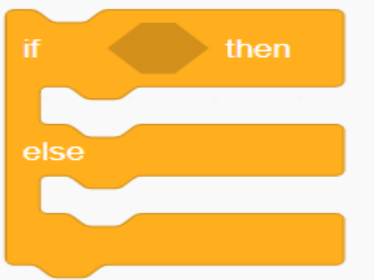
STACK: instructions, to be connected to each other in a sequence





C/BRACKET: grouping instructions/control structures





CAP: end of program/agent death

Scratch blocks

Programming Environment:

Stage + Multiple Agents + Costumes + Sounds

Stage with multiple pictures (containing "global" code and vars)

- Switchable background (with when-switched event)
- Background vectorial editor (with text)
- Reacts to events and messages (you cannot call its code)

Multiple agents: contain their personal code/variables and can:

- Move and draw (Turtle-inspired), change appearance, ask or show text, play music or sounds, interact with each other through events and messages …
- The agent's "costume" is vectorial and could contain text (but you cannot separately move the costume's parts, unless they are separate agents)

The agent CAN READ/SET her personal vars and the global variables

Agents can be cloned!!! (each clone has a copy of her mum's variables)

It CANNOT SET other agent's variables (can READ from mum's only)

Programming styles

Event-based

Multiple threads for the same event!

Agents updates itself by reading the other's state (and globals)

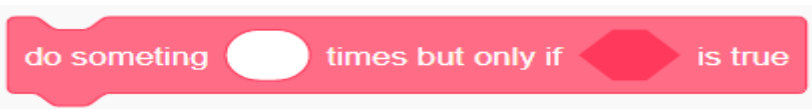
Code modularization: through PROCEDURES (new blocks)

NOTICE: there is no "return" instruction

(but can be faked with a global variable)

Procedures are LOCAL to the Agent or Stage

They accept simple arguments (numbers/lists/booleans)



RECURSION? YES (but no "return" and no local variables)

Extensions

Lego Mindstorms EV3/WeDo/Boost

Music

Video sensing

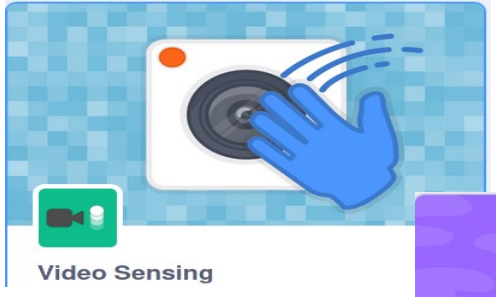
Pen (turtle graphics)

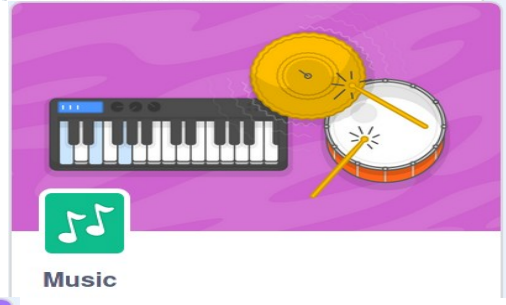
Text-to-speech

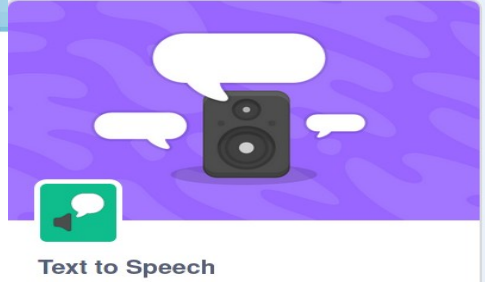
Translate

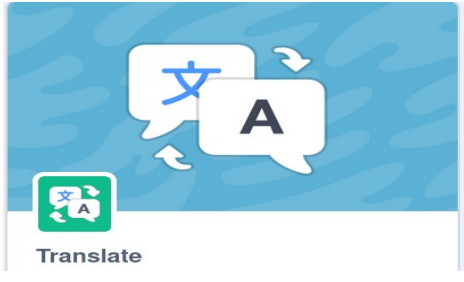
Micro:Bit microcontroller

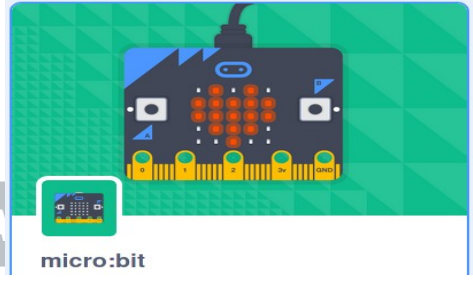
Force/acceleration sensor











DEBUGGING

You can show Global and Agent's variables

You can change them at runtime (with a slider if numbers)

You can slow-down execution and highlight the running block

You can try what a block does by clicking on it

You can build an "observer" agent that tracks message "probes"

This way you keep the code separate from the debugger agent

Code quality/complexity tool: www.DrScratch.org

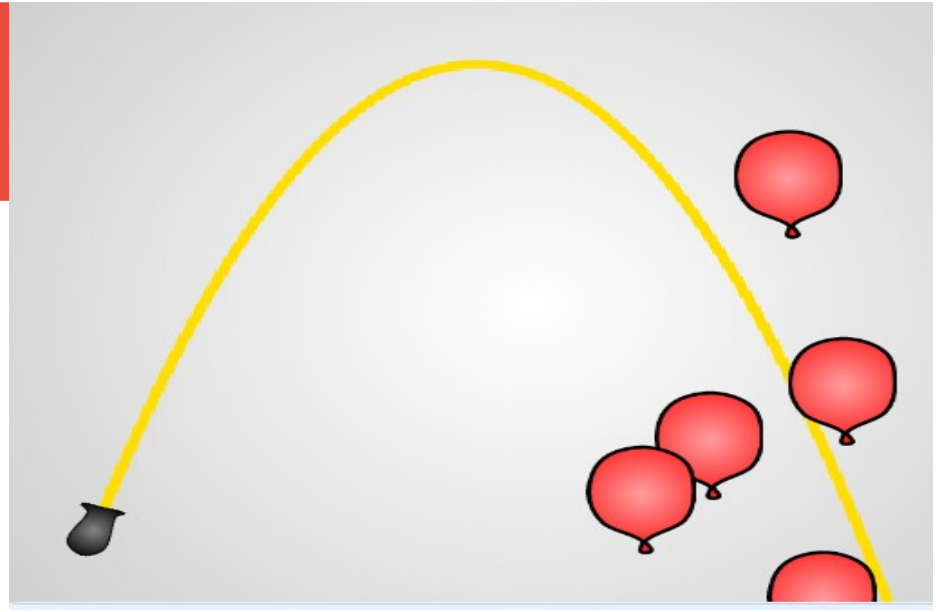
Extracts many nice indicators from the submitted project:

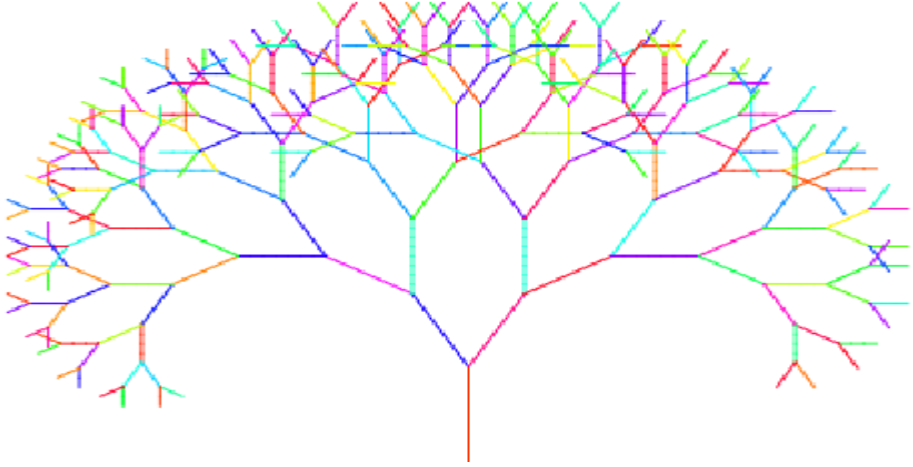
- Flow control: 0=sequence, 1=repeats, 2=if-then-else
- Data representation: 0=no variables, 1=variables, 2=lists
- Abstraction: 0=single program, 1=modularity, 2=clones
- Interactivity: 0=single event, 1=ask/say+mouse, 2=video/audio
- Synchronization: 0=wait time, 1=messages, 2=wait until/when X
- Parallelism: 0=single thread, 1=multi-thread, 2=when X events
- Logic: 0=if-then, 1=if-then-else, 2=if with multiple conditions

Then you get a nice certificate and best-practices suggestions

Examples

Shoot the balloons ==>





<== Procedural Recursive tree

<== Rec. tree with parallel clones

Scratch language ability to show C.T. topics

Algorithm	YES	(stacked blocks)	
Structured control	YES	(bracket blocks, loops, if-then-else)	
Code reuse	YES	(My Blocks, but only procedures)	
Procedures	YES	Functions	NO
Scope	LIMITED	(global or agent scope, no local)	
Memory usage	LIMITED		
global variables YES, agent variables YES, local variables NO			
Data Types	LIMITED (numbers, booleans, strings, flat lists)		
Static types	LIMITED (booleans vs. other values)		
Agents attributes	YES	with methods	(MESSAGES)
Events	YES	Messages	YES (named, empty)
Concurrency	YES	Clones	YES

Scratch + extensions vs. interdisciplinary subjects

Agent-based simulation	(Physics, Math, ...)
Agent-based animation	(Literature, History, Art, …)
Music extension	(Music, Rhythm, Harmony)
Text-based interaction	(Interactive Fiction, Text generation)
Lego Robot extension	(Robotics, Physics)
Voice recognition / Text to Speech	(non-textual interaction)