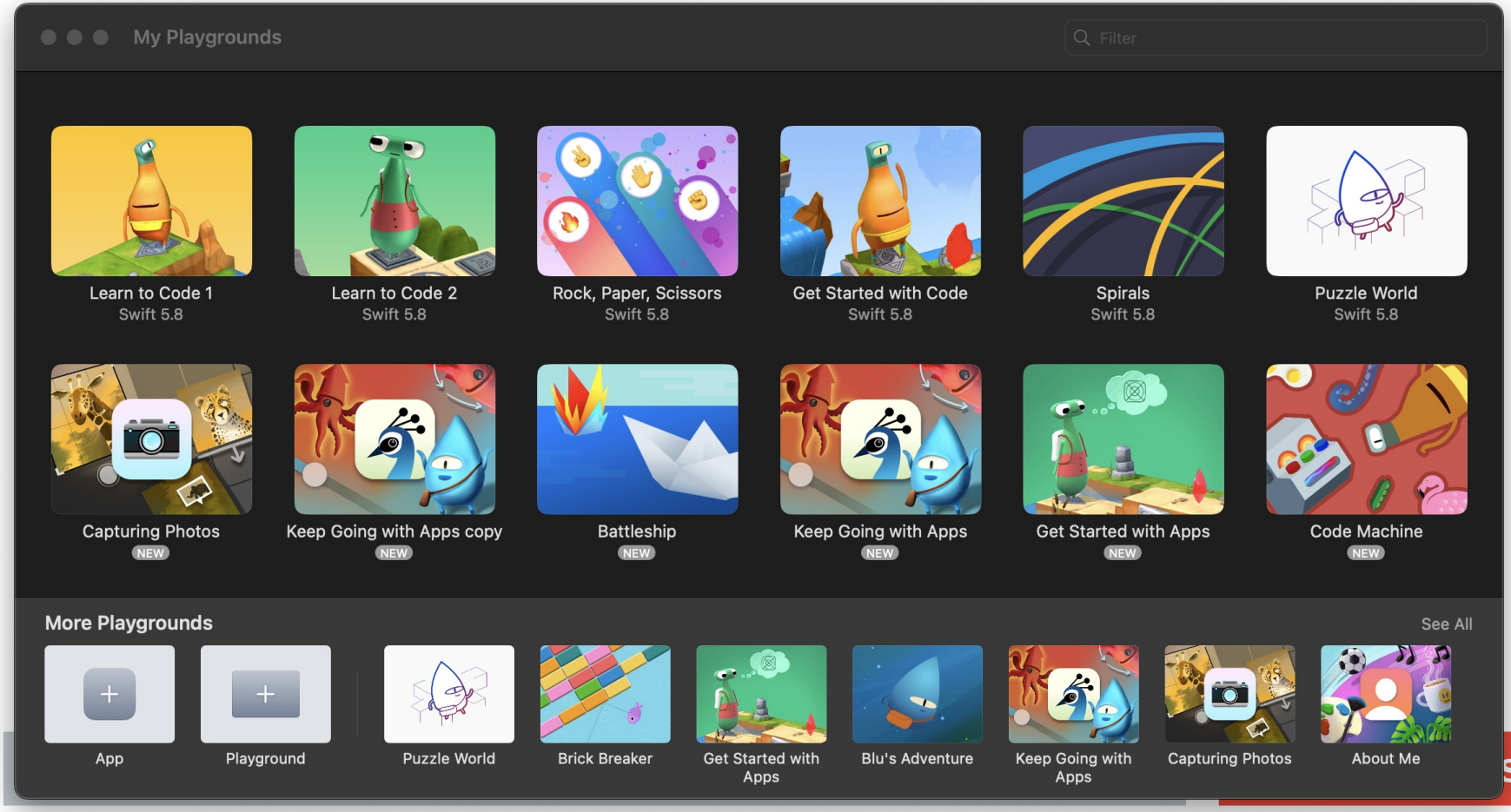
Other interesting systems

Andrea Sterbini – sterbini@di.uniroma1.it

Swift Playgrounds (for MacOS)



systems

Swift programming language

Born as a replacement of Objective-C (originally for NeXT OS, then in Mac OSX)

Runs well together with C/C++ and Objective-C compiled libraries

Statically typed, with type inference

With a simpler syntax (e.g. no ‘;’, simpler range specs: 1 … N)

With closures (aka anonymous functions)

Protocol-based class extensions (called “mixins” in other languages)

Functions can be passed as values

Generic functions/methods, operator overloading

Array bounds always checked

Variables always initialized …….

No pointer arithmetic or other unsafe operations

Playgrounds: small programming tasks

E.g. Loops, conditionals, boolean ops, functions, variables, types, methods ….

Start with a short set of slides explaining new concepts

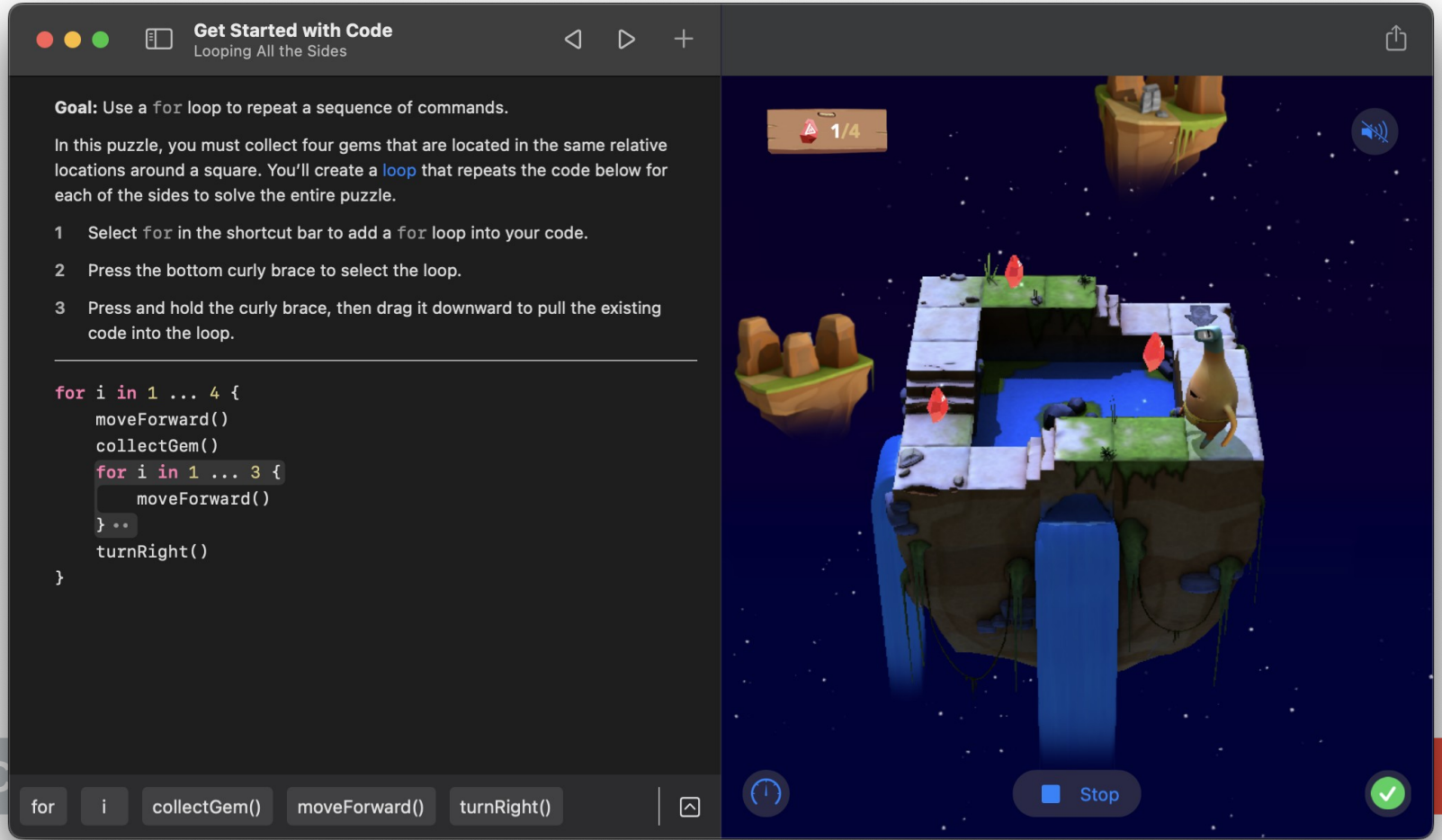
Describe the task (normally involving an animated character)

With automatic checks for the program syntax

With programming templates to write control structures and function calls

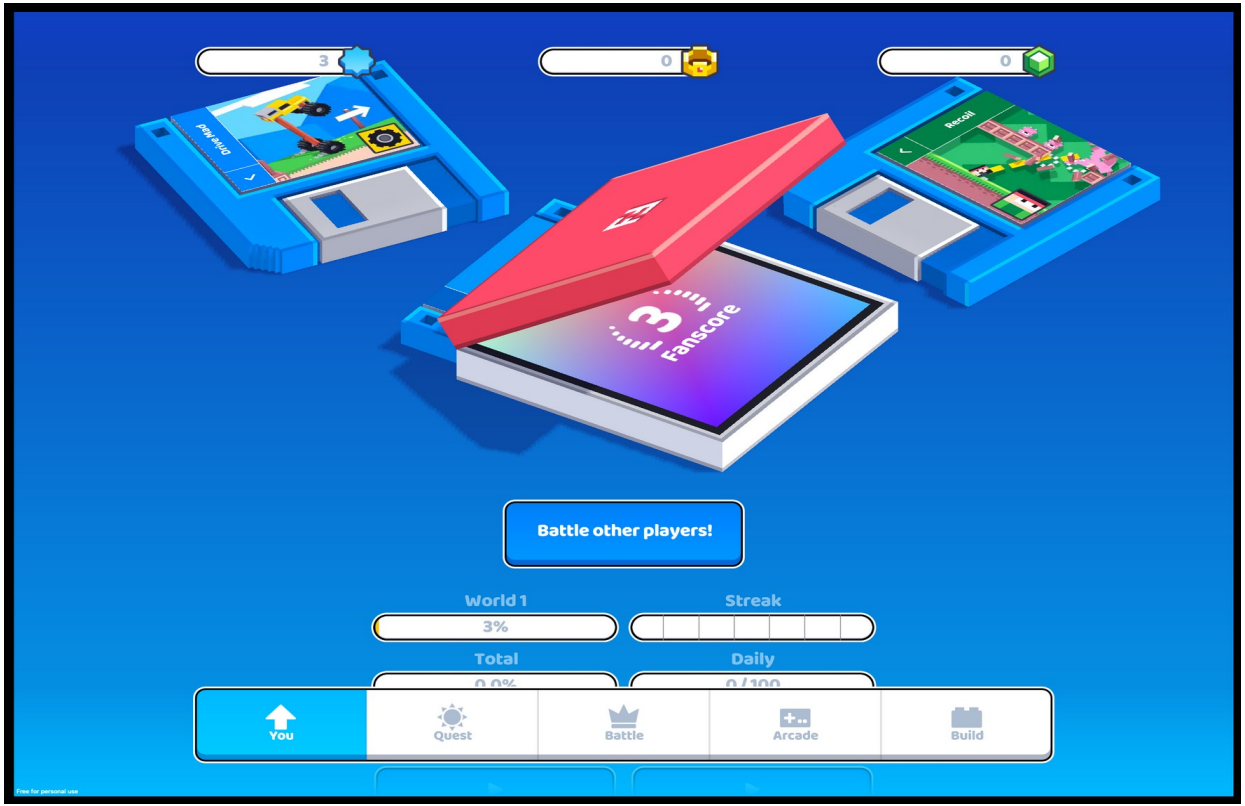
Small programming tasks moving a creature in a labyrinth

- Move around
- Collect gems
- Activate buttons
- Activate portals
- Use conditions
- Use loops
- ...



Methods in C

FANCADE



Methods in Computer Science education: Analysis

22-23 Other systems

Fancade: a 3D data-flow game building language

Available in Android, Windows, MacOS, or in the Browser

Data-flow computation WITH VARIABLES

Typed links (Number, Boolean, Position, Rotation, Object, Constraint, …)

NO text! (except for code comments and menus)

Executed 60 times for second (implicit forever loop)

Execution order: left-right and top-down (with respect to the isometric positions)

Explicit Variables (used also to reduce wires and increase fan-out)

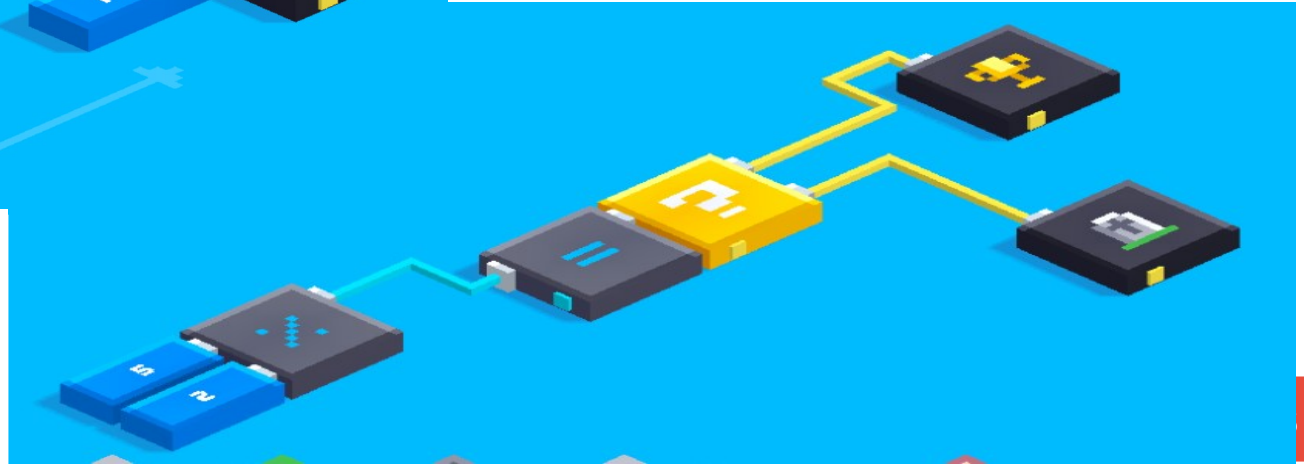
Code can read/write game blocks properties (position/speed/bounce …)

Can add or change movement/rotation constraints for blocks/agents

Explicit Control = before/after links (jellow wires)

Loops

do:
Num += 1
while num<100



If-then-else

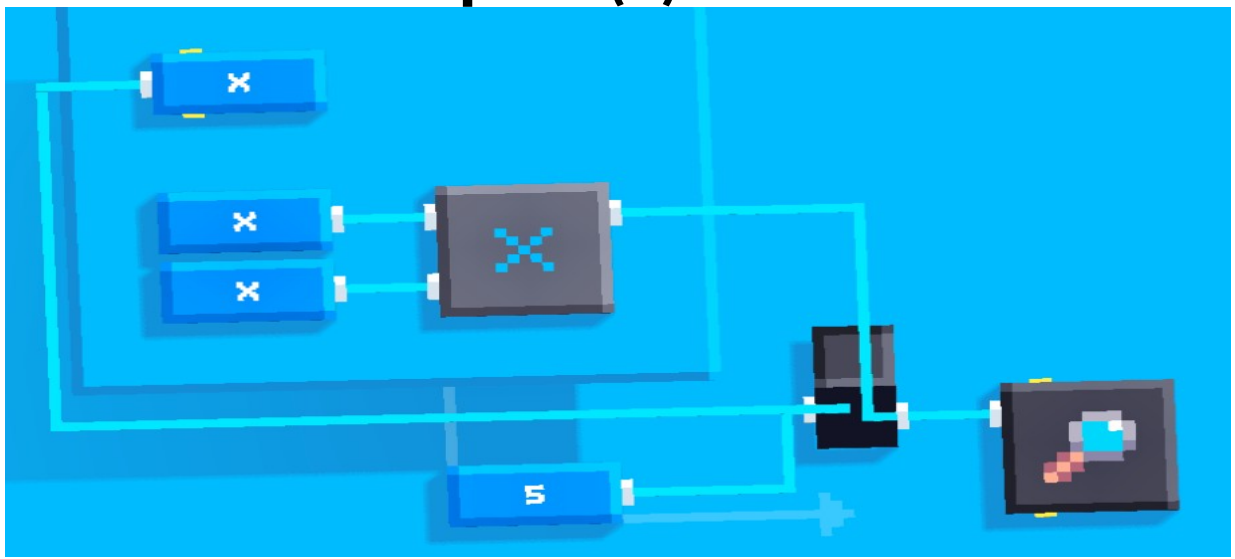
if 5%2==0:
then: win else: lose

Functions: custom blocks

Functions? **YES**

Just put “code” inside a block, with external connectors

def square(x): return x*x

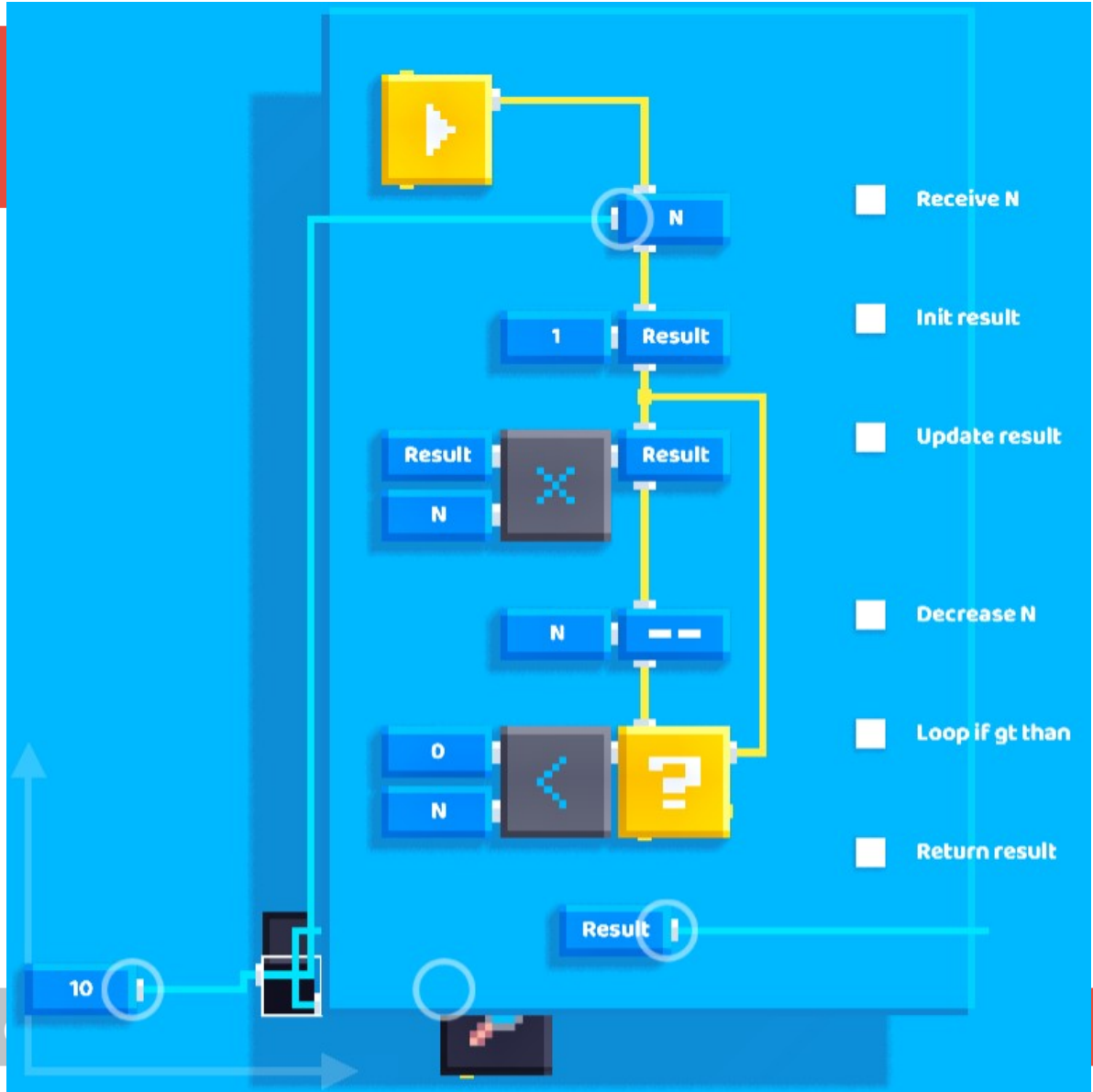


Modularization:

- **custom blocks can be placed inside other blocks**
- **BUT: <u>NO recursion is allowed</u> (direct or indirect)**

No recursion (even implicit)

Iterative factorial(N)



Many game-oriented

VALUES (different types)

VARIABLES (getter/setters

MATH (operators)

CONTROL (if-then, loops ..

PHYSICS (gravity, vectors,

GAME (win/lose/score)

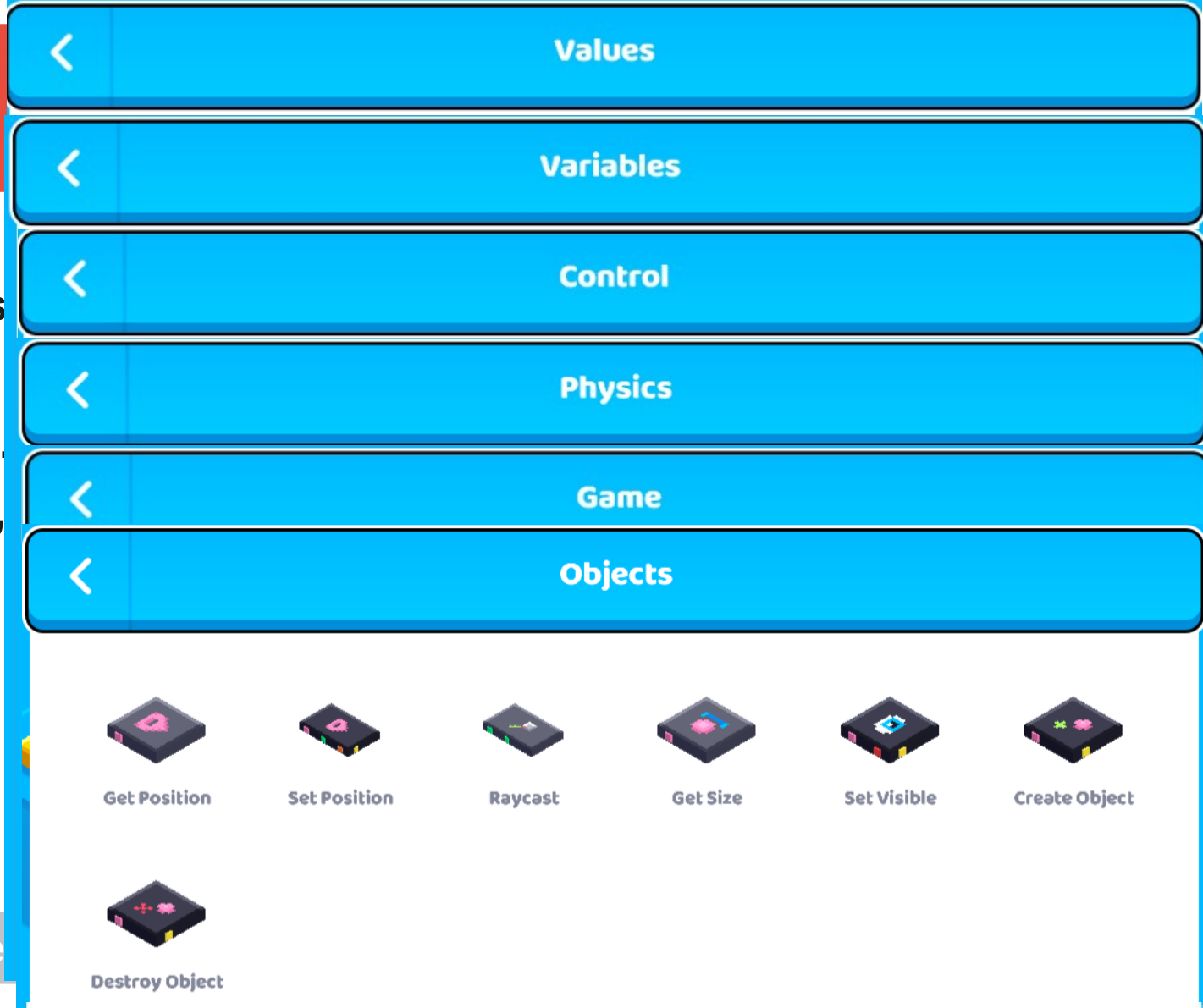
SCENERY

SCRIPTS

INSPECTORS

CREATURES

Methods in Compute



DEMO

PlayOsmo

TANGIBLE interaction for prescholar kids (with iPad or Fire)

Your tablet scans the table in front of it (on a stand, with a mirror on camera)

You play with tiles (instructions or tangram pieces or letters or drawings)

It recognizes your “code tiles” or words or drawing depending on the game/app

Loops and simple repetitions

Blocks represent action and direction

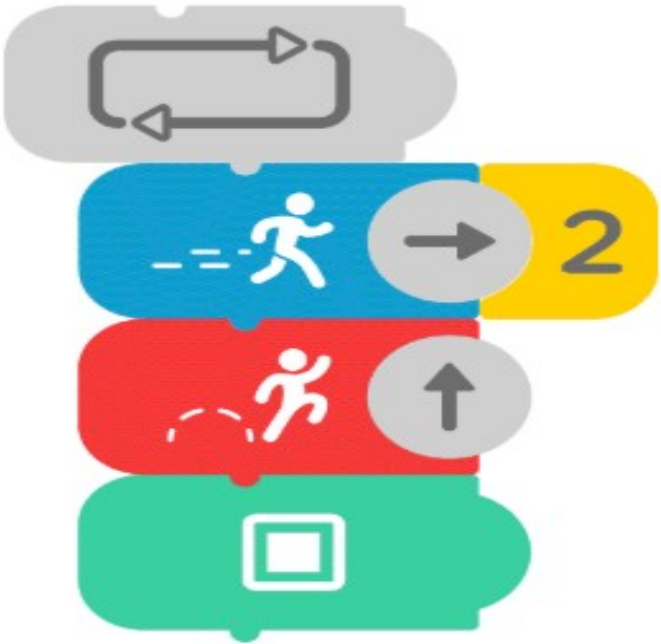
- modifier: number of repetitions

Loops

The instructions' meaning depends on the game

- Music playing game

- Monster in a labyrinth game



But also many game apps

Drawing (virtual reality helped drawing)

Spell/word games (character recognition)

Tangram (shape recognition)

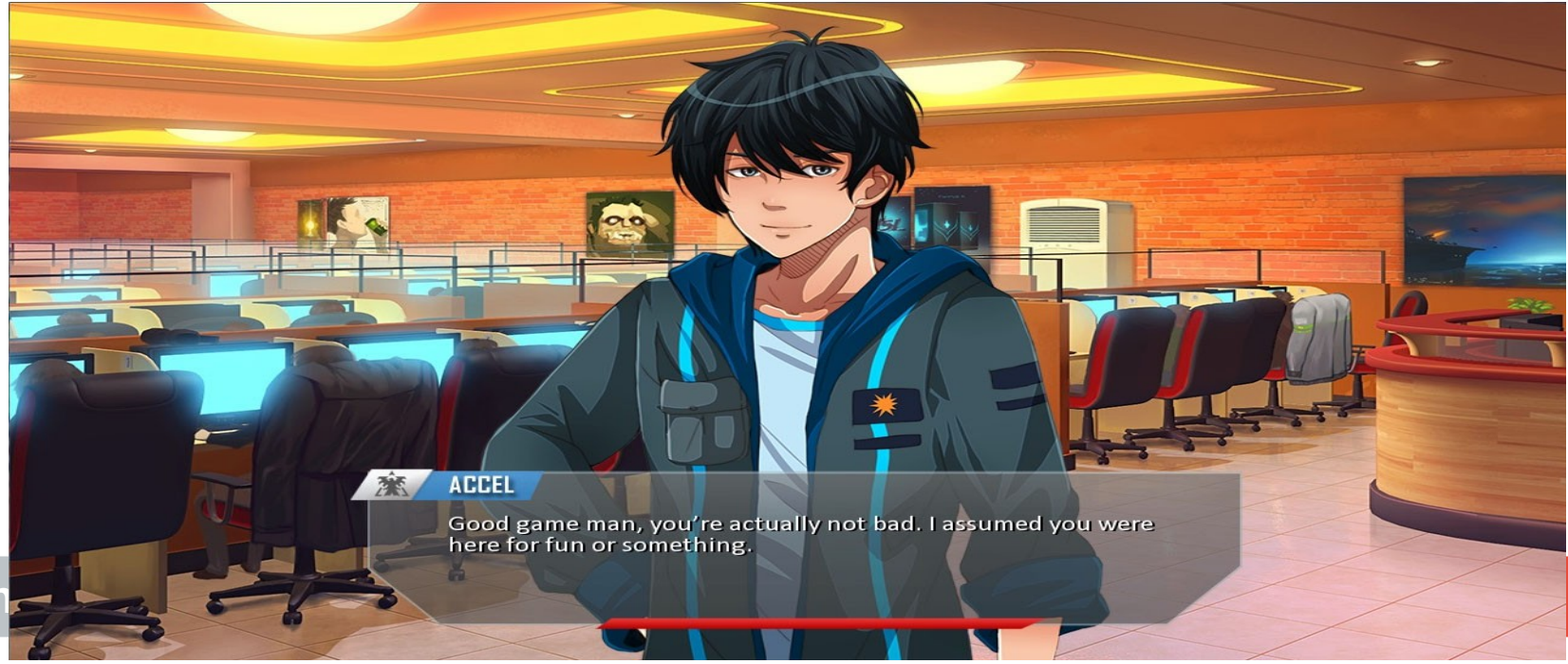
Pizza shop (shapes + money/math)
Wizards (math or words)
...

Ren’Py

Dialogue-based adventures (Visual Novels)

Python-inspired simplified syntax (with indentation)

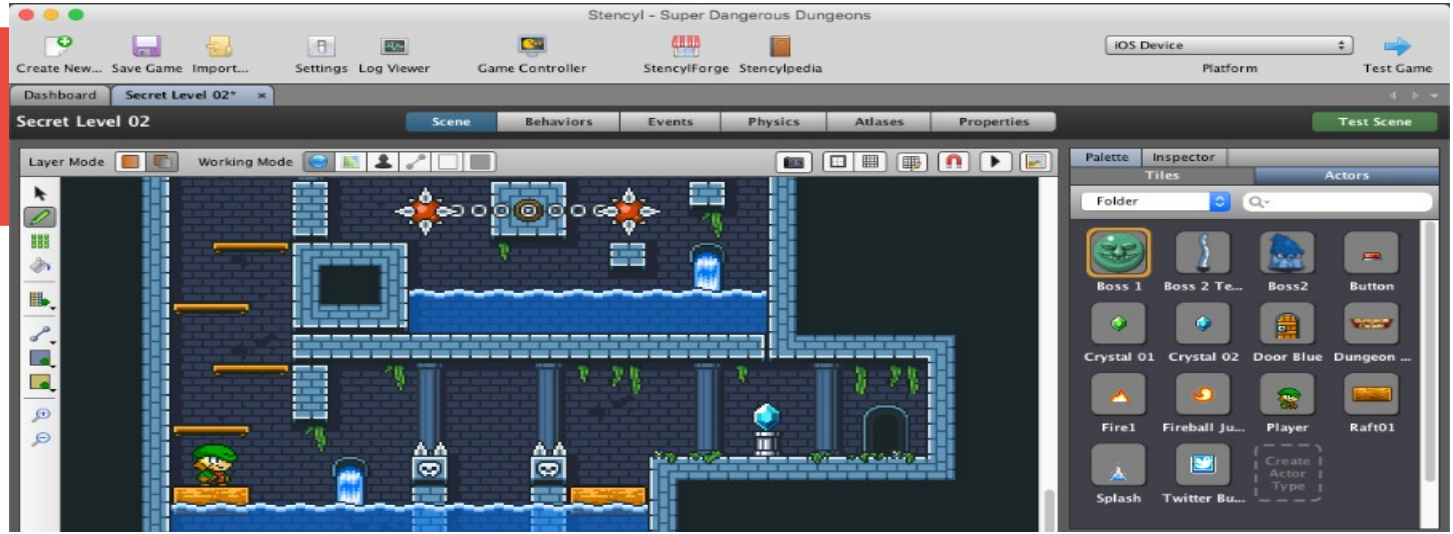
Runs in Python

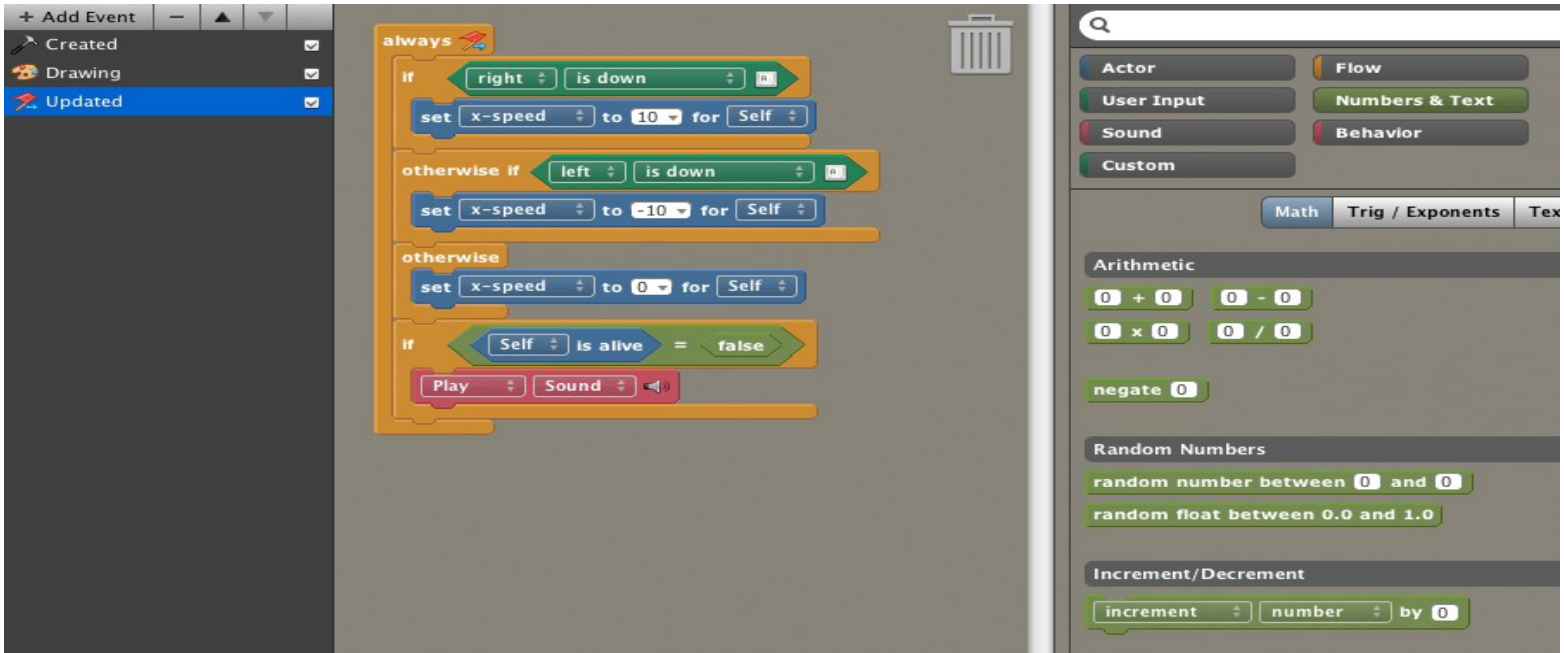


Methods in

Stencyl

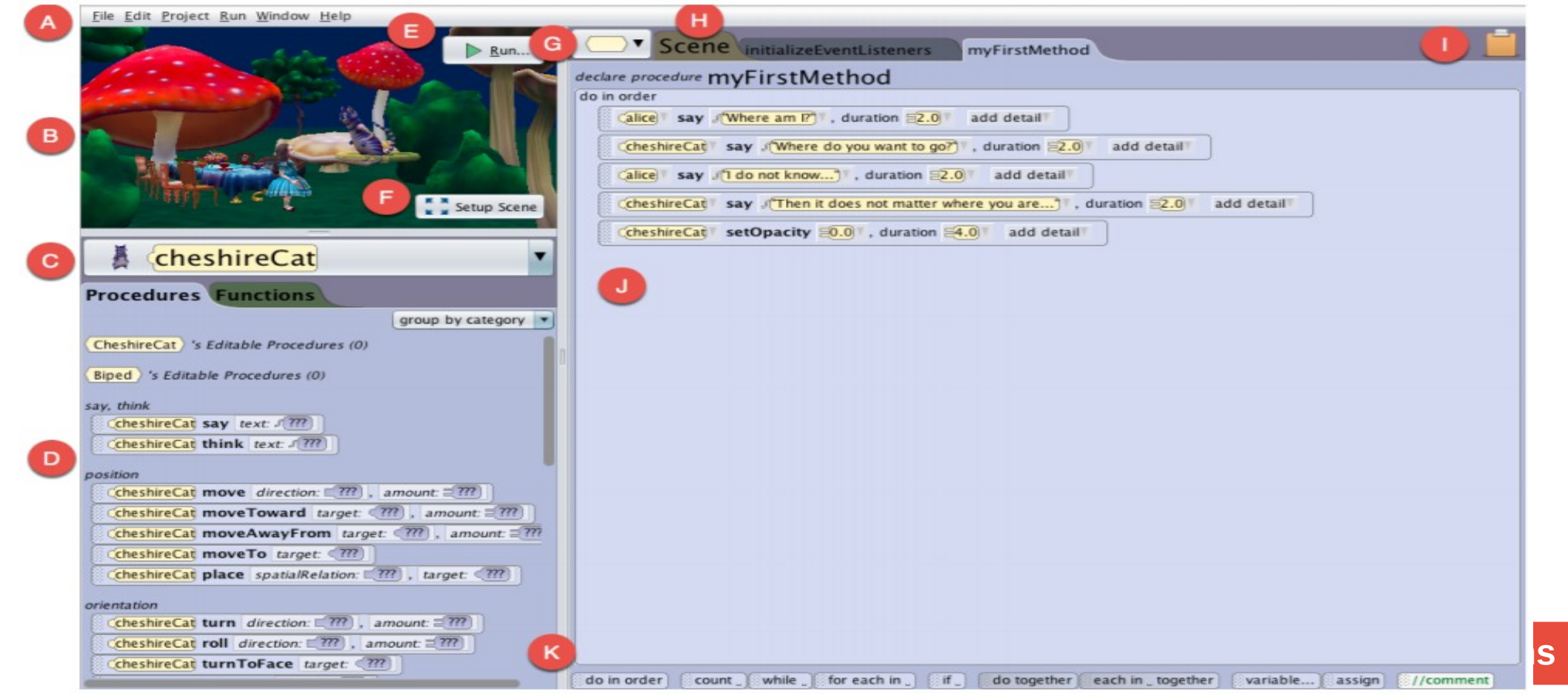
Game designer (platforms)





22-23 Other systems

Alice 3 : 3D world programming



Others

Kodu: Design 3D games on Xbox (and PC)

Kodular: game editor

Construct: HTML5 game editor

MakeBlock: robots/microcontrollers + Scratch

Tynker: Scratch-inspired

Roblox: 3D game programming

… and many other game-editors