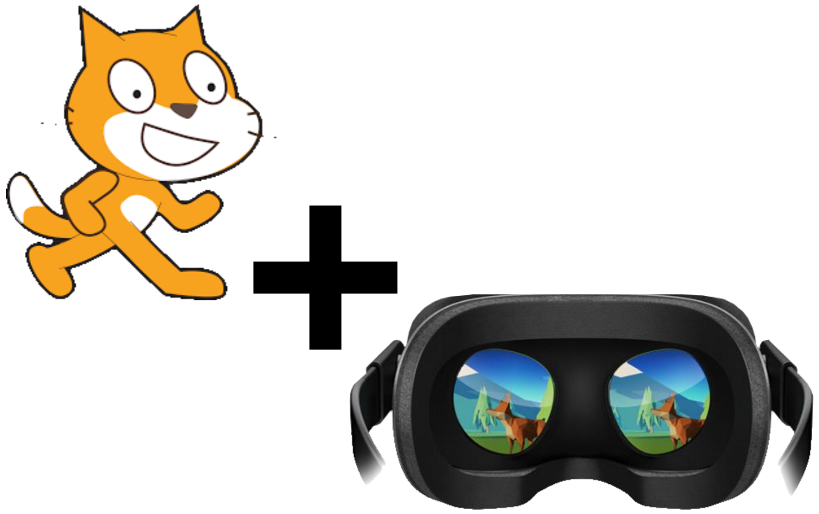
ScARtch

AUGMENTED-REALITY VISUAL ENVIRONMENT FOR PROGRAMMING BEGINNERS

Matteo Boschini

Purpose



- ***Scratch*** is a coding learning enviroment that employs a block-based **visual programming language**.

- **Virtual Reality** technology allows for **intuitive and natural interaction** with computers.

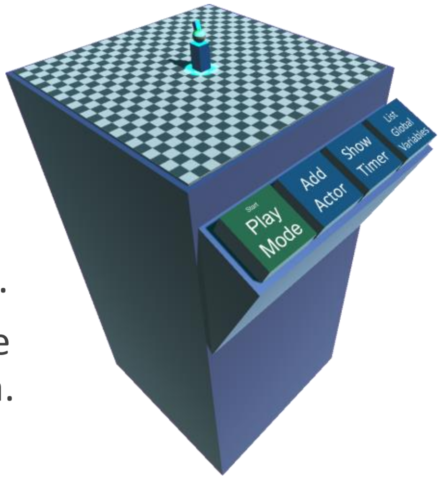
- Applying the latter to the former **simplifies user interaction** (especially relevant if they are not used to traditional UI).

Objectives

- Definition of a **graphic block-based** language built upon the **structured programming** paradigm with the following characteristics:
  - Instructions are represented by **blocks** that can be **composed** to obtain **scripts**.
  - Special blocks with **intuitive shapes** represent specific **control structures**.
  - Introduction of different types of **variables** and **expressions**.
  - Implementation of a **message transmission system** to allow an instruction to trigger other scripts.
  - (Limited) ability to provide **input** through VR controllers.
- Realization of a development environment, called **Playground**, where the user can:
  - **Build** Scripts.
  - Run them and observe their effects on **graphic elements**.

Overview

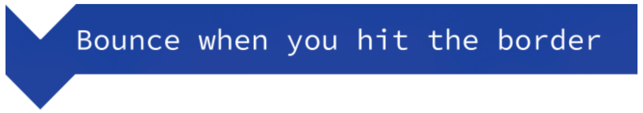
- **Playground**
  - **Scene:** a static background
  - **Actors**: entities moving on the scene.
  - **Sound and Models Archive**: respectively sound effects and three-dimensional models that we can associate with actors.
  - **Controls:** In particular, to switch between the scripting mode (**Edit mode**) and execution mode (**Play mode**) and vice versa.



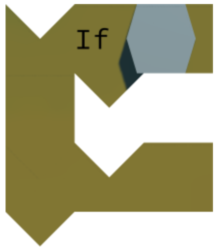
- We associate every **Actor** with
  - A **position**, a **rotation**, a **scale coefficient**, and a **sound volume** value.
  - A three-dimensional **model** that represents it.
  - **Scripts**: programs that are assembled through the appropriate interface.
  - A **message** that can be used to provide output.

Scripting Elements (I)

- **Scripts** are composed of the following elements:
  - **Simple Blocks**, containing a single statement.

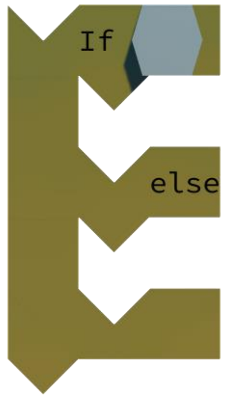


  - **Control blocks**, Used for control structures (if, while,...). They have a *port* in which a sequence of additional blocks can be added.

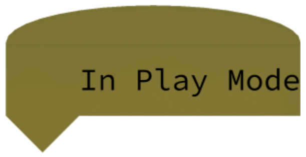


Scripting Elements (II)

- **Scripts** are composed of the following elements:
  - **Double control Blocks**, used for the If/else control structure. They have two *ports* for the insertion of additional block sequences.

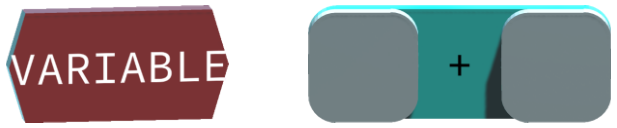


  - **Hats**, elements that begin scripts and contain their execution condition.

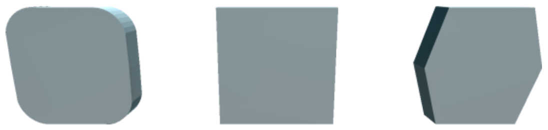


Scripting Elements (III)

- Some blocks have **boxes** where **operands** can be inserted.
  - An operand is a **variable** or an **expression** of other operands. Both of these elements are represented with appropriate scripting elements.

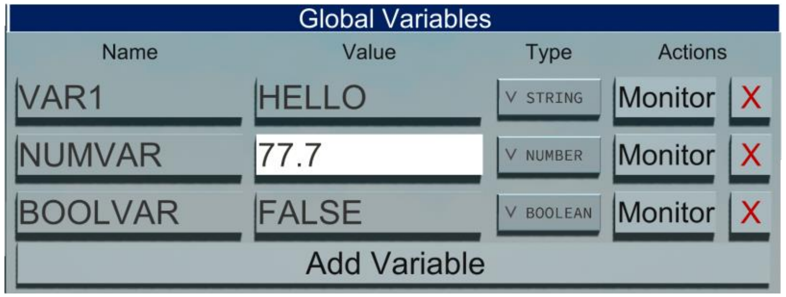


  - An operand is always associated with a type (can be **string**, **number**, or **Boolean**). Numbers and Booleans can also be inserted in boxes that require a string. Different types of boxes and scripting elements are recognizable by their **shape**.



Scripting Elements (IV)

- **Variables** are defined through the controls of the programming environment (separately from scripts), there are, however, instructions to **assign them new values**.

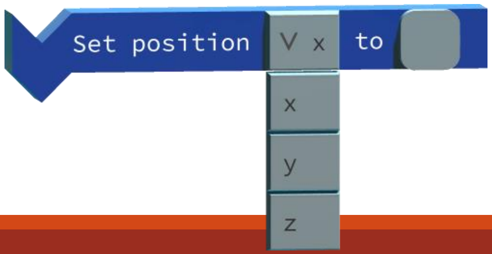


Global Variables

Name	Value	Type	Actions	
VAR1	HELLO	V STRING	Monitor	X
NUMVAR	77.7	V NUMBER	Monitor	X
BOOLVAR	FALSE	V BOOLEAN	Monitor	X

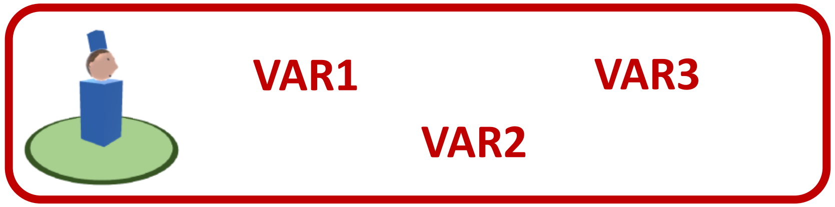
Add Variable

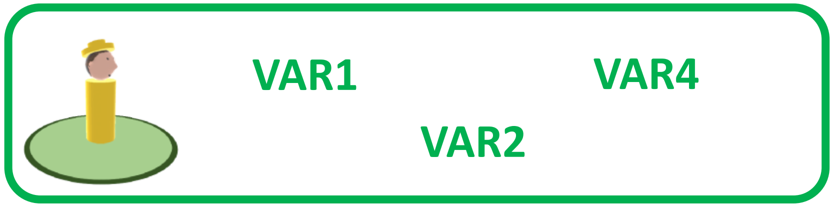
- Some blocks have **options**: boxes with drop-down menus for selecting a value in a predetermined list.



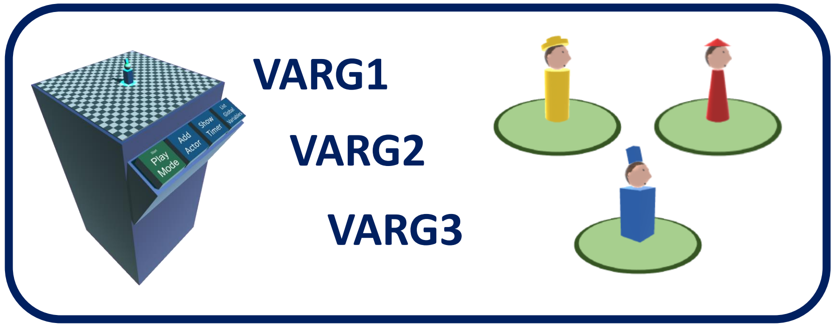
Scoping

- Each **actor** defines **local variables** on which he has exclusive visibility.



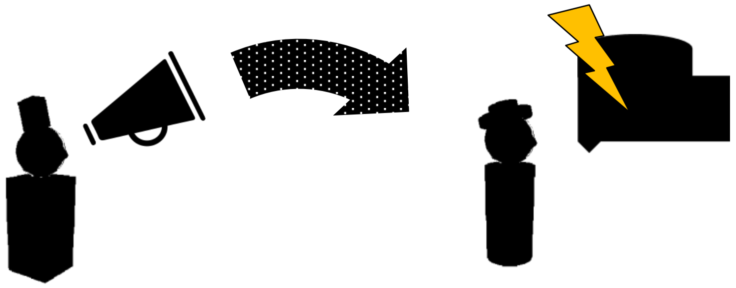


- **Global variables** that are visible to any actor can also be defined.

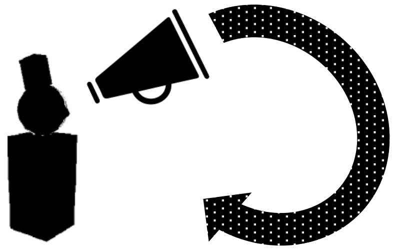


Messages

- An actor can **broadcast a message** that contains a string, triggering the execution of scripts that begin with the **appropriate receiving hat**.

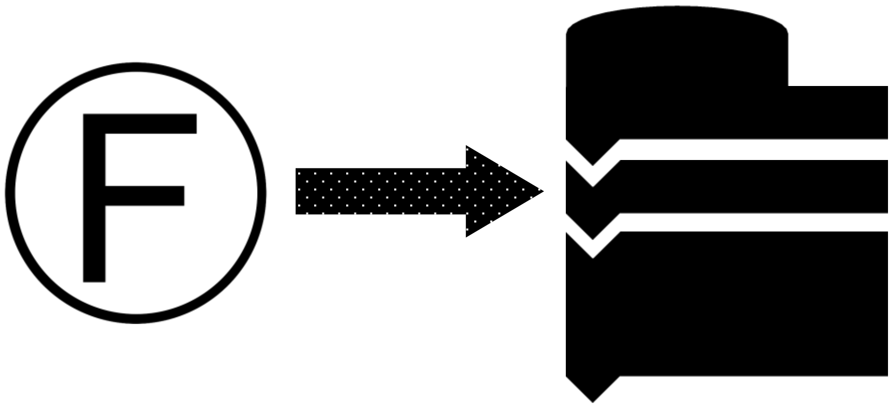


- You can take advantage of this mechanism for simulating function calls (without explicit arguments).



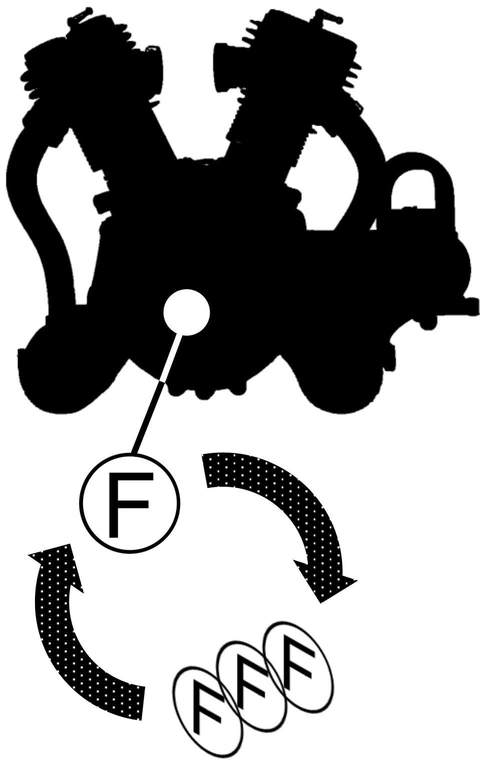
Evaluation (I)

- In Play Mode, under certain conditions (e.g. hat), an **execution flow** is generated, which contains a **pointer** to the current block.
- The block contains in its class the logic for **evaluating** and **updating** the flow with the next block.



Evaluation (II)

- The script evaluation engine executes the instructions **sequentially and with Time-Division**. Instructions are kept in a **queue of execution flows**.
  - The **first statement of the flow** stream is executed.
  - Upon completion, if the flow **still contains instructions**, it is inserted in the **back of the queue**.
  - After a "didactic" waiting time, the first instruction of the following flow is executed.



Architettura dell’ambiente

Model

Scripting

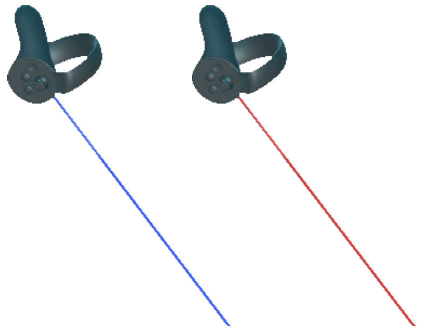
Controller

View

Interface (I)

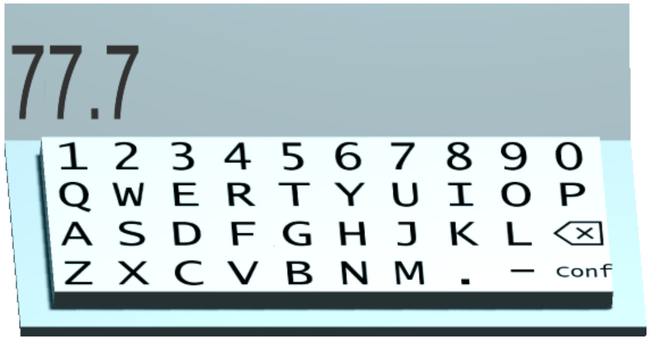
- The user displays the environment through a **NewtonVR-compatible headset** (i.e. *Oculus Rift* o *HTC Vive*).
  - You can move by **teleporting** (using buttons *B/Y*).

- Interaction is possible through the **controllers**.
  - Windows and scripting elements can be **grabbed** (using the *grip* button).
  - Buttons, actors, textboxes, etc. allow for interaction with laser pointers (activated with buttons A/X).
    - The **blue pointer** is used for selection.
    - The **red pointer** is used for deleting items/closing windows.
  - During Play mode, only the blue pointer is available and you scripting elements cannot be moved.



Interface (II)

- For interactions that require textual input, a **virtual keyboard** is employed.
  - It is activated when by **clicking the analog stick** and appears near the controller.
  - Select a text area while the keyboard is open and assigns it **focus**.
  - Any virtual keyboard input is **subjected to a compatibility check** before being accepted. Syntax errors are filtered at this level.



Demo

- Creation of a program from scratch (actor that moves and says «Hello World»).
- Factorial computation (iterative: new allocation record creation is not supported).
- Actor following the controller and example the usage of the messaging system.

Conclusions

- Possibilities of further development:
  - A proper **saving and loading system**, possibly emphasizing sharing (see Scratch Community).
  - Introduction of a function definition sub-system and correct handling of **activation records** (allowing, in particular, to define recursive functions).
  - Expansion of the repertoire of instructions under the **sensors** category by introducing blocks to detect **collisions** between actors.
  - VR/AR platforms are currently evolving:
    - Porting on smartphone-based platforms.
    - Consider porting on future platforms that are being developed (Google Daydream, Windows Holographic, Apple ARKit, new standalone headsets that will appear in the next few years).