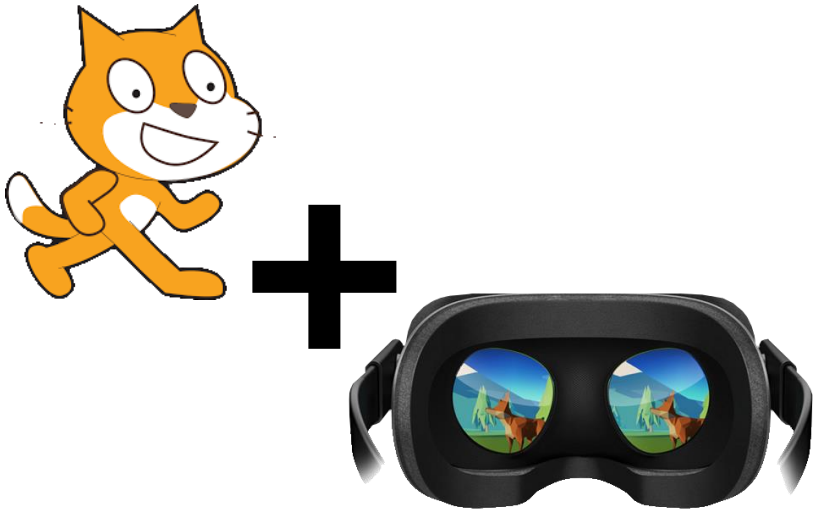
# ScARtch

## AMBIENTE DIDATTICO IN REALTÀ AUMENTATA PER L'AVVIAMENTO ALLA PROGRAMMAZIONE

Matteo Boschini

# Finalità



- ***Scratch*** è un ambiente di sviluppo a fini didattici che fa uso di un **linguaggio di programmazione grafico** a blocchi.
- La tecnologia di **realtà virtuale** consente un'**interazione più intuitiva e naturale** con il calcolatore

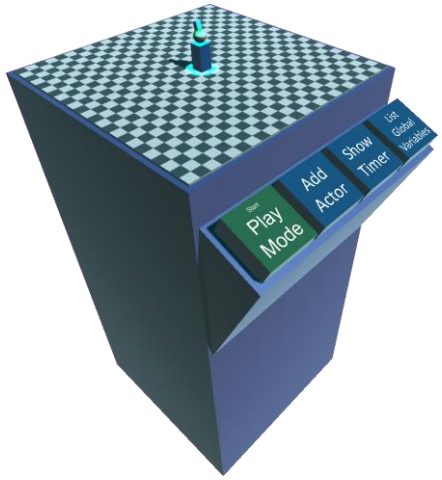
- La applicazione della seconda al primo **facilita l'interazione** dell'utente (specie se non abituato alle interfacce classiche).

# Obiettivi

- Definizione di un linguaggio **grafico** a **blocchi** basato sul paradigma di **programmazione strutturata** con le seguenti caratteristiche:
  - Le istruzioni sono rappresentate da **blocchi componibili** in script.
  - Blocchi speciali con **forme intuitive** rappresentano le diverse **strutture di controllo**.
  - Introduzione di **variabili** ed **espressioni** di diversi tipi.
  - Implementazione un **sistema di trasmissione di messaggi** per consentire ad una istruzione l'innesco di altri script.
  - Possibilità (limitata) di fornire **input** attraverso i controller VR.
- Realizzazione di un **ambiente di sviluppo**, detto *Playground*, in cui l'utente può:
  - Costruire **script**.
  - Metterli in esecuzione ed osservare i loro effetti su **elementi grafici**.

# Overview

- **Playground**
  - **Scena**: uno sfondo statico
  - **Attori**: entità che si muovono sulla scena.
  - **Archivio di suoni** e **modelli**: rispettivamente effetti sonori e modelli tridimensionali che possiamo associare agli attori.
  - **Controlli:** in particolare, per passare dalla modalità di realizzazione degli script (**Edit mode**) a quella di esecuzione (**Play mode**) e viceversa.

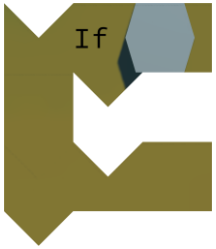



- Ad ogni **Attore** sono associati
  - Una **posizione**, una **rotazione**, un **coefficiente di scala** e un valore di **volume sonoro**.
  - Un **modello** tridimensionale che lo rappresenta.
  - **Script**: programmi realizzabili con l'apposita interfaccia.
  - Un **messaggio** che può essere usato per fare output.

# Elementi di Scripting (I)

- Gli **script** sono composti dagli elementi seguenti:
  - **Blocchi semplici**, che contengono una sola istruzione.


  - **Blocchi di controllo**, usati per le strutture di controllo (*if, while, …*). Presentano una *bocca* in cui è possibile inserire una sequenza di blocchi aggiuntiva.



# Elementi di Scripting (II)

- Gli **script** sono composti dagli elementi seguenti:
  - **Blocchi di controllo doppi**, usati per la struttura di controllo if/else. Presentano due *bocche* per l'inserimento di sequenze di blocchi aggiuntive.


  - **Cappelli**, elementi che aprono gli script e ne contengono la condizione di esecuzione.

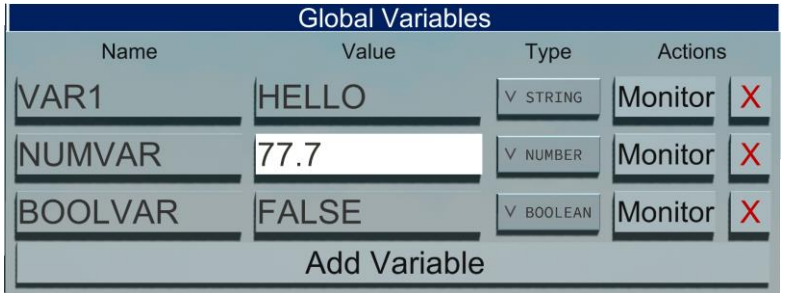
# Elementi di Scripting (III)

- Alcuni blocchi presentano **caselle** in cui possono essere inseriti **operandi**.
  - Un operando è una **variabile** o una **espressione** di altri operandi. Entrambi questi elementi sono rappresentati con opportuni elementi di scripting.

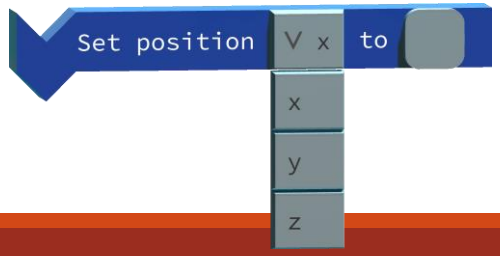

  - Un operando è sempre associato ad un **tipo** tra **stringa**, **numero** e **booleano**. In una casella in cui si richiede un operando di tipo stringa, è possibile usare anche operandi di tipo numero e booleano.
    Caselle ed elementi di scripting di tipo diverso sono riconoscibili dalla loro **forma**.

# Elementi di Scripting (IV)

- Le **variabili** si definiscono con i controlli dell'ambiente di programmazione (separatamente rispetto agli script), ma sono disponibili istruzioni per **assegnare loro valori diversi**.



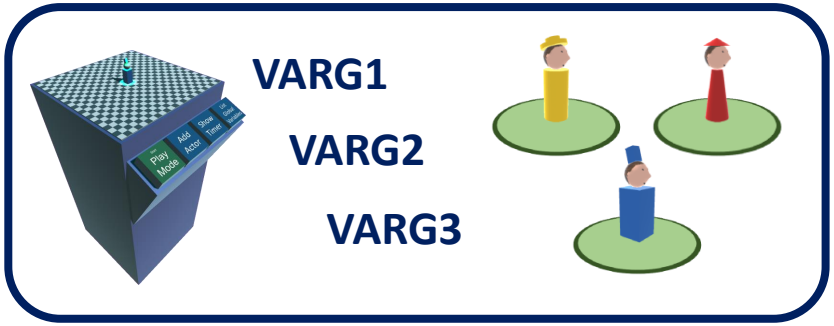

- Alcuni blocchi presentano **opzioni**: caselle con menù a tendina per la selezione di un valore in un elenco prestabilito.



# Scoping

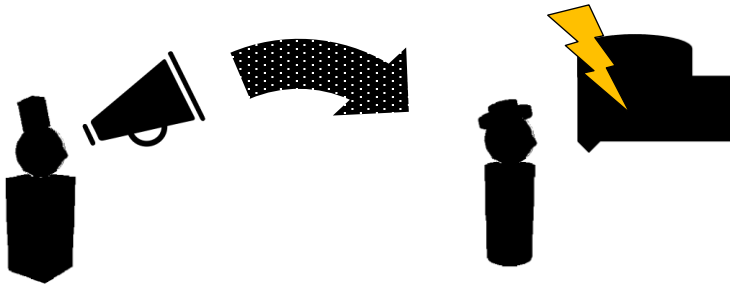
- Ciascun **attore** definisce **variabili locali** su cui ha visibilità esclusiva.


- Sono definibili **variabili globali** che risultano visibili per qualsiasi attore.

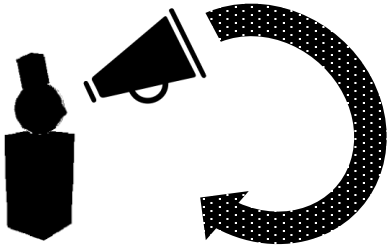


# Messaggi

- Un attore può **trasmettere in broadcast un messaggio** che contiene una stringa, scatenando l'esecuzione di script che cominciano con l'**opportuno cappello di ricezione**.

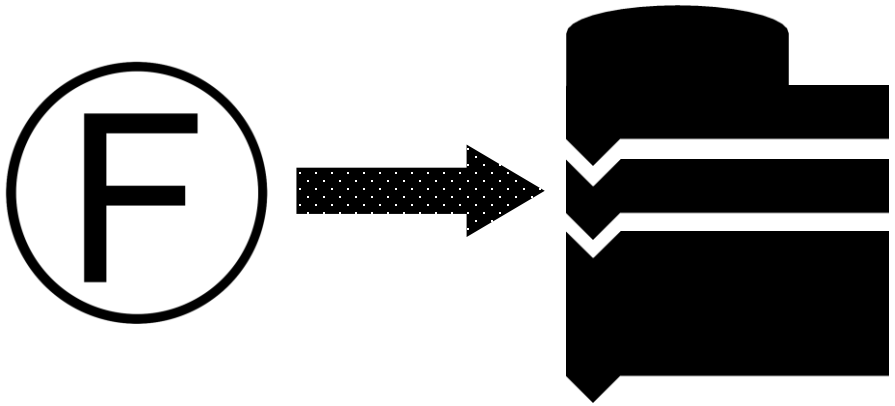


- È possibile sfruttare questo meccanismo per la simulazione di chiamate a funzione (senza argomenti espliciti).



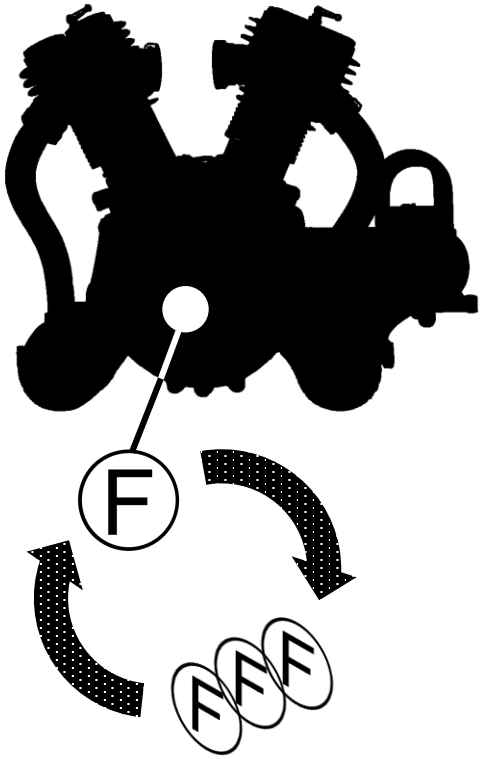
# Valutazione (I)

- In Play Mode, sotto determinate condizioni (es. cappello), viene generato un **flusso di esecuzione**, che contiene un puntatore al blocco corrente.
- Il blocco contiene nella sua classe la logica per la **valutazione** e per **l'aggiornamento** del flusso con il blocco successivo.



# Valutazione (II)

- Il motore di valutazione degli script esegue le istruzioni **in sequenza e a divisione di tempo**. Le istruzioni sono mantenute in una **coda di flussi di esecuzione**.
  - Viene mandata in esecuzione la **prima istruzione del primo flusso**.
  - Al suo completamento, se il flusso non è **esaurito**, viene inserito **in fondo alla coda**.
  - Si ha una attesa ”didattica” e si manda in esecuzione la prima istruzione del flusso seguente.



# Architettura dell’ambiente

**Model**

**Scripting**

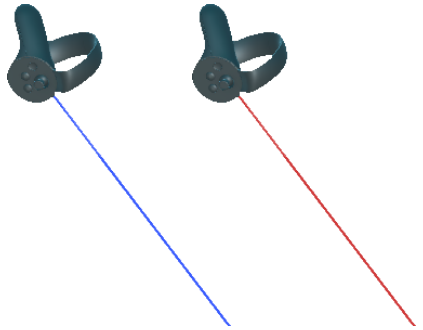
**Controller**

**View**

# Interfaccia (I)

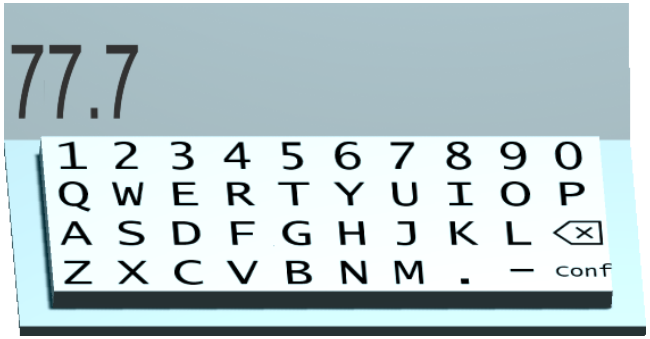
- L'utente visualizza l'ambiente attraverso un **visore** compatibile con **NewtonVR** (i.e. *Oculus Rift* o *HTC Vive*).
  - È possibile spostarsi **teletrasportandosi** (tasti *B/Y* per attivare).

- Si può interagire attraverso i **controller**.
  - Finestre ed elementi di scripting possono essere **afferrati** (usando il tasto *grip*).
  - Bottoni, attori, textbox, ecc. permettono di interagire con i **puntatori laser** (si attivano con i tasti *A/X*).
    - Puntatore **blu** per selezionare.
    - Puntatore **rosso** per eliminare/chiudere.
  - Durante il Play mode, è disponibile solo il raggio blu e non è possibile spostare elementi di scripting.



# Interfaccia (II)

- Per le interazioni che richiedono input testuale, viene introdotta una **tastiera virtuale**.
  - Si attiva alla pressione dello *stick analogico* e compare vicino al controller.
  - Selezionare un'area di testo con la tastiera aperta le assegna il **focus**.
  - Qualsiasi input da tastiera virtuale viene **sottoposto ad un check di compatibilità** prima di essere accettato. Errori di sintassi sono filtrati a questo livello.



# Demo

- Esempio di creazione di un programma da zero (attore che si sposta e dice «Hello World»).
- Esempio di calcolo del fattoriale (iterativo: non si supporta la creazione di nuovi record di allocazione).
- Attore che segue il controller e esempio di impiego del sistema di messaggistica.

# Conclusioni

- Possibilità di sviluppo ulteriore:
  - Un sistema adeguato di **salvataggio e caricamento**, che enfatizzi la **condivisione** (vedi comunità di *Scratch*).
  - Introdurre un sistema di definizione funzioni e gestione dei **record di attivazione**, che consentirebbe, in particolare, la definizione di funzioni ricorsive.
  - Espandere la categoria **sensori** introducendo blocchi per l'individuazione di **collisioni** tra gli attori.
  - Le piattaforme di VR/AR sono correntemente in via di evoluzione:
    - Effettuare porting su piattaforme smartphone-based.
    - Considerare future piattaforme in via di definizione (Google Daydream, Windows Holographic, Apple ARKit, nuovi headset standalone che compariranno nei prossimi anni).