Luigi Ferrando
Leonardo Sasso

AL QUADRATO

LABORATORIO DI CODING E DI EXCEL

a cura di A. Ferraresso, E. Colombini, G. Pettarin

COMPETENZE

DIDATTICA INCLUSIVA

AeD AMBIENTE EDUCATIVO DIGITALE

DeA SCUOLA | Petrini

Luigi Ferrando
Leonardo Sasso

AL QUADRATO

LABORATORIO

DI CODING E DI EXCEL

DeA SCUOLA
Petrini

Petrini

internet: deascuola.it
e-mail: info@deascuola.it

Redattore responsabile: Alessio Delfrati
Redazione: Claudia Borgioli, Giovanni Malafarina
Redazione multimediale: Vincenzo Belluomo
Tecnico responsabile: Alessandro Cafagna
Prestampa: Grande Foredit - Monza
Progetto grafico: Michele Riffaldi
Impaginazione: Luca Savorani, Finotello
Ricerca iconografica: Claudia Borgioli, Luca Savorani
Ricerca iconografica per la copertina: Alice Graziotin
Copertina: Silvia Bassi, Matteo Rossi

Art Director: Nadia Maestri

Microsoft Excel è un marchio depositato di *Microsoft Corporation*.

Si ringraziano per i materiali forniti e i preziosi suggerimenti Gianluca Angi, Giulio Bonanome, Emiliano Boscaro, Casal, Luigi Ferraresco, Nicola Miotello, Onward Luxury Group, Chiara Tovena, Simone Zulian.

Il sito web di mBot è www.makeblock.com

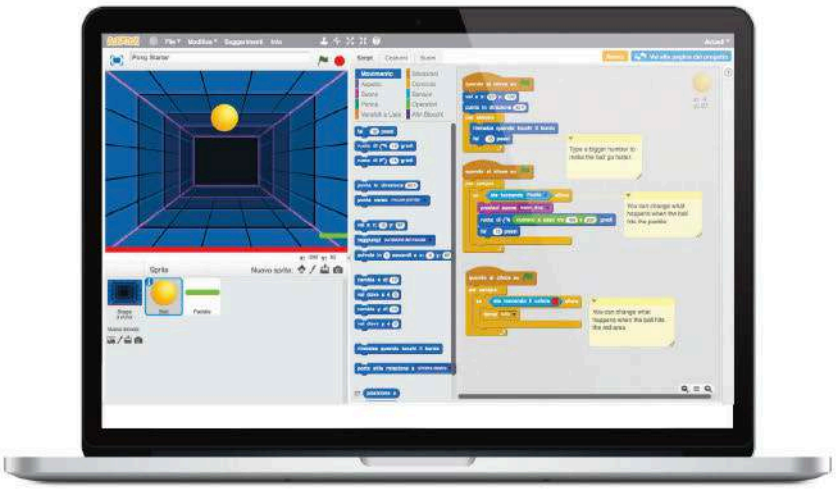
1ª edizione: gennaio 2017
Printed in Italy

Le fotografie di questo volume sono state fornite da: Autori; Shutterstock; Carlo Guaita.
Immagine di copertina: Shutterstock

Presentazione

Il coding per la Matematica

I computer sono sempre più presenti nella nostra vita, soprattutto se pensiamo che sono computer anche smartphone, tablet, console di giochi e tanti altri dispositivi elettronici. Spesso, però, ci limitiamo a utilizzarli in maniera passiva. Giochiamo con un videogame, navighiamo in un sito web e poco più. Usufruiamo, insomma, di contenuti che altri hanno già preparato per noi. È come se potessimo percorrere solamente le strade già tracciate dagli altri.



E se cominciassimo a **creare** noi strumenti e contenuti interattivi? L'informatica, se messa al servizio dell'immaginazione e della creatività, può ottenere risultati impensabili. Che cosa hanno in comune un videogame, una app per lo smartphone e un sito web? Sono tutti programmi per computer.

Imparare a **programmare** significa essere in grado, ad esempio, di creare in autonomia una app e di poterla condividere con i nostri amici.

Ma come si programmano i computer? Per prima cosa, bisogna imparare a comunicare nel loro **linguaggio**. I computer, per funzionare, si attendono da noi una lista di istruzioni ben precise, il **programma**, appunto, e queste istruzioni vengono espresse in un **linguaggio di programmazione**.

Così come avviene per gli esseri umani, anche nel mondo dei computer esistono diversi linguaggi, alcuni più semplici e facili da imparare, altri molto complessi e che richiedono molto sforzo per essere assimilati.
In questo libro abbiamo deciso di partire con **Scratch**, uno dei migliori strumenti didattici esistenti in questo campo. Sviluppato all'interno della prestigiosa università statunitense Massachusetts Institute of Technology (MIT), Scratch presenta un approccio visuale, intuitivo e semplificato alla programmazione, indicatissimo per ottenere in poco tempo dei buoni risultati e aumentare nel tempo le proprie conoscenze in questo campo.

I concetti di programmazione presentati da Scratch sono comunque trasferibili ai linguaggi di programmazione più "seri", quali Java, Python e molti altri ancora.

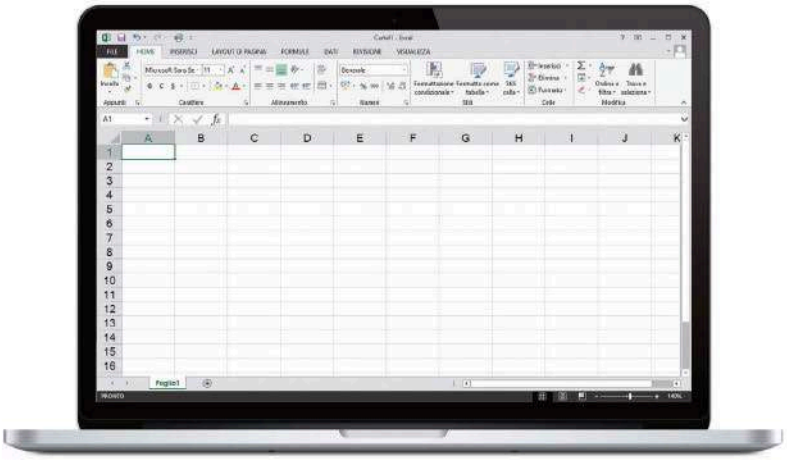
La **programmazione a blocchi** di Scratch ha ispirato anche l'ambiente didattico **Code.org**, utilizzato dal progetto ministeriale «Programma il futuro», che comprende l'iniziativa «L'ora del codice».

Partendo dai concetti di Scratch, analizzeremo poi i migliori strumenti di Code.org, tra i quali **App Lab**, che affianca alla programmazione a blocchi una buona introduzione alla programmazione testuale tramite l'usatissimo linguaggio di programmazione Javascript.

La programmazione ha uno stretto legame con la **matematica**: si basa su **algoritmi**, precise sequenze di istruzioni con una logica analoga a quella delle operazioni di calcolo, la logica del **pensiero computazionale**. Quando si programma, perciò, si mettono sempre in gioco anche **competenze matematiche**. In questo manuale vedremo però anche applicazioni specifiche per creare veri e propri strumenti matematici.

Il foglio di calcolo

Come scopriremo in queste pagine esistono software che permettono a milioni di utenti in tutto il mondo di usare la matematica per costruire grafici, tabelle, fare dei calcoli, analizzare risultati di esperimenti scientifici e così via: sono i **fogli di calcolo**. Il foglio di calcolo più diffuso è **Excel**, di cui esiste anche una versione *open source* che si chiama **Calc**.



In questo manuale proponiamo una serie di attività che ci permettono di imparare a utilizzare il foglio di calcolo, dai primi passi fino a funzioni avanzate.

Le attività ripercorrono alcuni argomenti di matematica che si affrontano in classe, con le quali realizzeremo degli strumenti utili per la risoluzione degli esercizi.

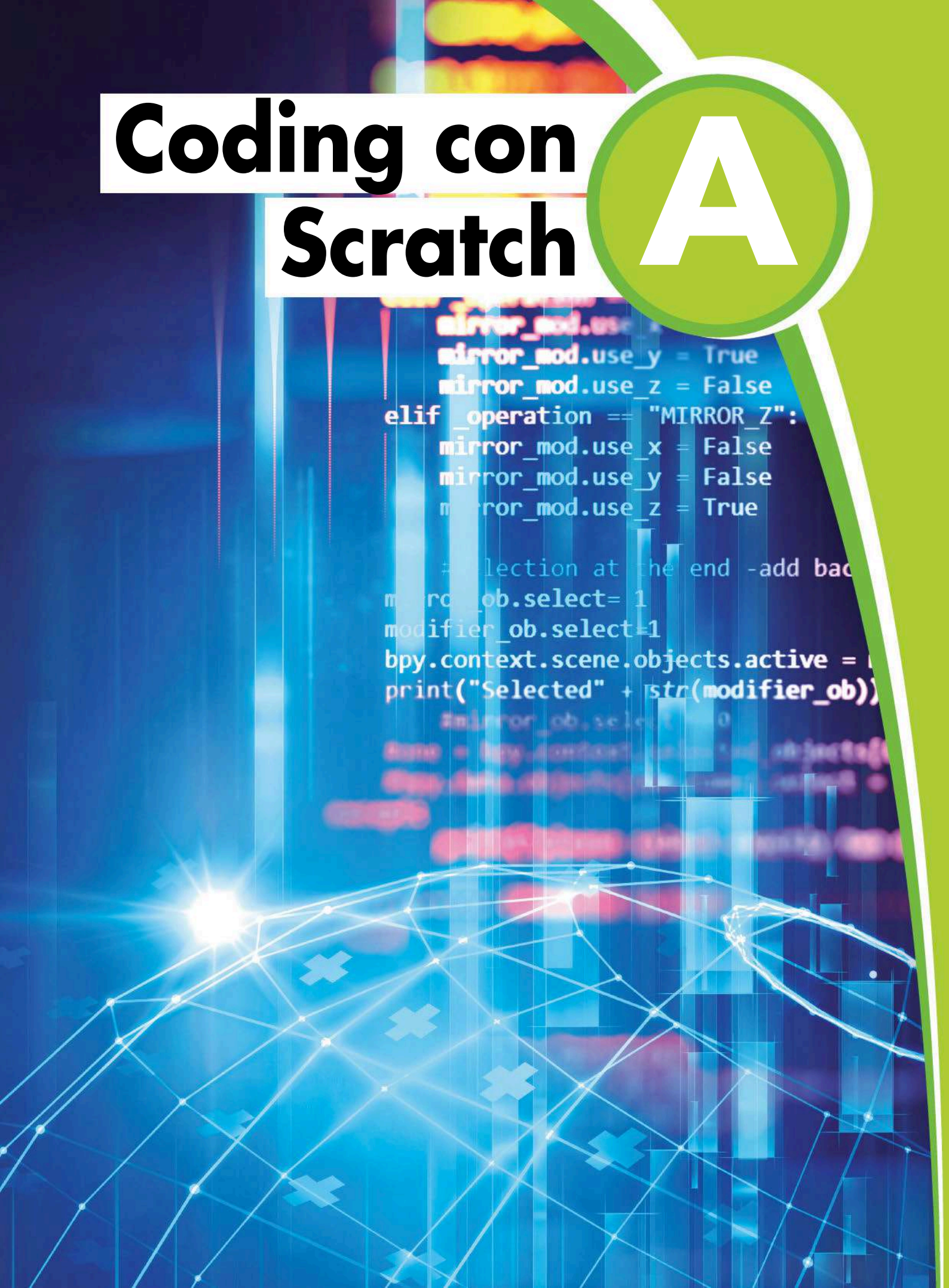
Nel loro svolgimento occorre prendere dimestichezza con gli **algoritmi** e la **logica**, proprio come nella programmazione con Code.org: con un po' di fantasia e concentrazione saremo in grado di realizzare fogli di lavoro utili e accattivanti.

Icona Download

I programmi vanno costruiti seguendo le istruzioni del manuale. Tuttavia i file dei programmi più avanzati, contrassegnati da questa icona, si possono scaricare dal sito del libro di *Al quadrato* di L. Ferrando, L. Sasso: www.deascuola.it/alquadrato.

Indice

Coding con Scratch



1 Coding e computer

Alcuni dispositivi di uso comune

Farsi capire da un computer

Con il termine *coding*, che deriva da *code*, cioè "codice", si può intendere una forma di programmazione semplificata e intuitiva degli elaboratori elettronici, che di seguito chiameremo semplicemente **computer**. Va tenuto presente che i computer non sono solo quegli apparecchi fissi o portatili dotati di tastiera e di monitor che tutti noi conosciamo ormai da anni.
Anche gli **smartphone**, le **console** per **videogiochi** e i **navigatori satellitari** delle automobili, per esempio, sono dei veri e propri computer, anche se più piccoli e maggiormente specializzati.

Dal punto di vista dell'**hardware** (cioè la parte fisica), la maggior parte dei computer che conosciamo oggi si basa sulla cosiddetta **architettura di von Neumann**.
John von Neumann (1903 – 1957) era un geniale matematico e fisico statunitense, di origine ungherese, che nel 1945 definì lo schema dei componenti che avrebbe dovuto avere un computer.

Questi componenti erano:

- un processore centrale (detto CPU, abbreviazione di *central processing unit*);
- la memoria principale (RAM, *random access memory*);
- l'unità di *input* per l'inserimento di dati;
- l'unità di *output* per restituire i dati elaborati all'utente;
- il *bus*, ovvero un canale per collegare le varie parti tra loro.

John von Neumann

Questa intuizione fu talmente potente che viene tutt'ora utilizzata, anche se con qualche modifica. Per esempio con l'aggiunta della memoria di massa (come gli hard disk e i supporti rimovibili), che all'epoca non esistevano.

Il cuore di computer è la CPU, un processore che può eseguire le sequenze di istruzioni che gli vengono impartite. Ogni CPU, a seconda della marca e del modello, è in grado di riconoscere solo un ben determinato insieme di comandi. Questi comandi sono costituiti da numeri (in inglese ***opcode***, che significa *operation code*, codice operativo) ed eseguono operazioni piuttosto semplici, coma la somma o il confronto di due valori. Per questo vengono dette "primitive".

Un **programma** non è altro che una sequenza di istruzioni.
Teoricamente, è possibile scrivere dei programmi utilizzando diret-

tamente le istruzioni comprese dalla CPU. In questo caso si parla di programma scritto in **linguaggio macchina**.
In questa maniera, però, è veramente difficile realizzare, in tempi ragionevoli, un'applicazione funzionante e priva di errori.
Per ovviare a questo problema, nel corso dei decenni sono stati ideati numerosi linguaggi di programmazione più semplici da utilizzare, detti "di alto livello". Questi linguaggi "artificiali" si avvicinano al linguaggio umano, anche se possiedono meno "vocaboli" (istruzioni) e seguono delle regole grammaticali rigide. Tra i linguaggi di programmazione di alto livello oggi più diffusi ricordiamo **C**, **C++**, **Java**, **Python**, **PHP** e **Javascript**.

I linguaggi di alto livello non vengono compresi direttamente dal processore centrale, ma esistono dei programmi che effettuano una "traduzione" nel linguaggio macchina. Spesso a una sola riga di istruzioni di un linguaggio di alto livello, per esempio:

```
print("Ciao a tutti!")
```

che ordina al computer di stampare la frase "Ciao a tutti", corrispondono numerose istruzioni in linguaggio macchina.

I programmi che si installano da DVD, da *pen drive* ("chiavette") o si scaricano tramite Internet sono spesso già pronti all'uso, ovvero già tradotti in istruzioni direttamente comprensibili dal processore centrale.
In questo caso si parla di **programmi eseguibili**, o semplicemente di **eseguibili**. Una app per smartphone è un esempio di eseguibile.

Macchine per calcolare

In Italia i computer si diffusero negli uffici, nelle scuole e nelle case all'inizio degli anni '80 del secolo scorso. Negli Stati Uniti questo fenomeno, invece, era iniziato verso la metà degli anni '70. In precedenza i computer esistevano già, ma erano molto più ingombranti e anche molto più costosi; pertanto venivano installati soprattutto in grandi aziende, in enti pubblici o privati e presso le forze armate.

Il computer **Mark I**, presentato dall'Università di Harvard (Boston) e dalla IBM nel 1943, era lungo circa diciassette metri, pesava quasi cinque tonnellate e conteneva oltre 765.000 componenti elettromeccanici. Era molto meno potente del piccolo smartphone che abbiamo in tasca: poteva memorizzare solo 72 numeri di 23 cifre ciascuno al massimo.

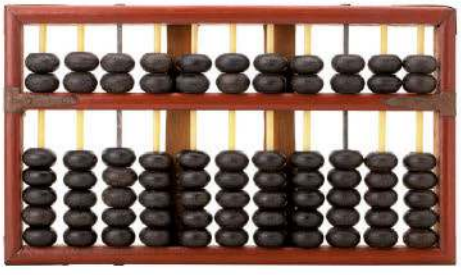
L'uomo ha costruito, o almeno tentato di costruire, computer nel corso di diverse epoche. Inizialmente si parlava di "macchine per il calcolo automatico", e potevano essere anche a vapore. Anche l'**abaco** può essere considerato uno strumento per aiutare l'uomo nel calcolo.

Uno degli aspetti più importanti dei computer è che possono essere programmati: sono così in grado di svolgere compiti per i quali, magari, non erano neppure stati concepiti.

Lo sai che?

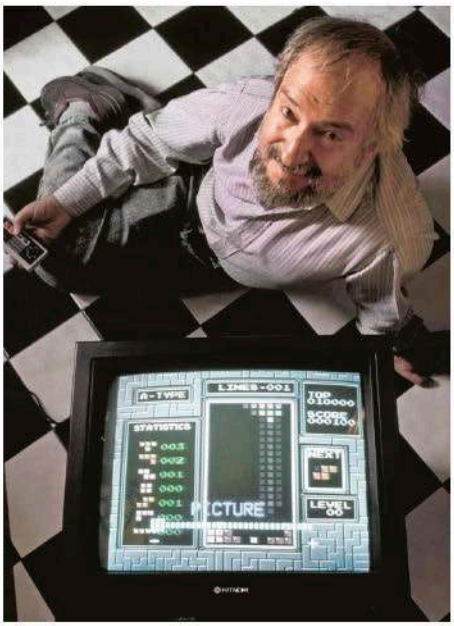
Software

Quando si parla di programmi, si usa spesso il termine **software**.
Questa parola è stata ideata dal matematico inglese Alan Turing, considerato uno dei padri dell'informatica. Un programma di videoscrittura o un videogioco sono esempi di software. A seconda del contesto in cui lo si usa, con software si possono intendere anche i dati memorizzati.



Un antico abaco cinese

2 Strumenti per imparare il coding



Seymour Papert

Imparare a programmare

I due principali strumenti oggi esistenti per imparare a programmare durante i primi anni del percorso scolastico sono **Scratch** e **Code.org**. Entrambi sono gratuiti.
Non si tratta di novità assolute, perché, già negli anni '60 del secolo scorso, il matematico pedagogista sudafricano **Seymour Papert** (1928 – 2016) aveva presentato il linguaggio didattico Logo, poi utilizzato da molti studenti, anche italiani.

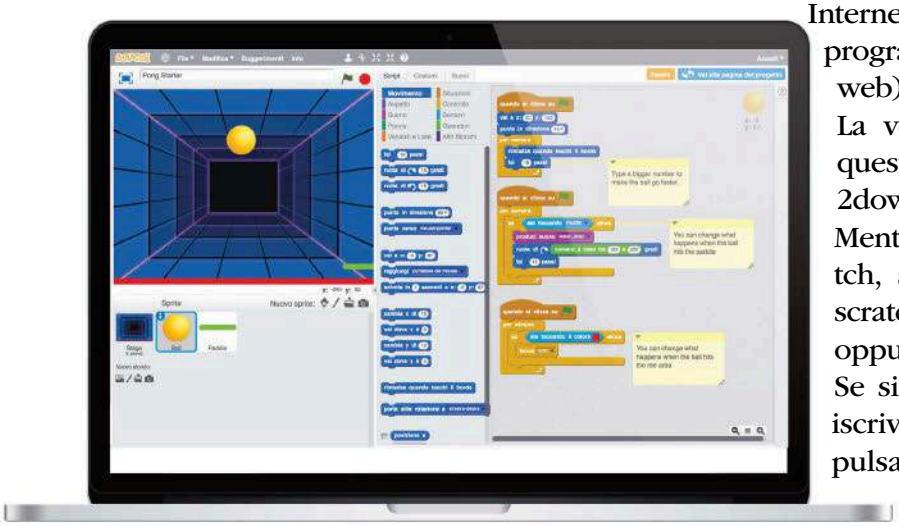
Scratch è l'erede di Logo, sia perché ne riprende alcuni concetti base, sia perché, come Logo, è nato nei laboratori del Massachusetts Institute of Technology (MIT), una delle più prestigiose università al mondo.
Code.org è un sito web, ispirato a sua volta ai blocchi di programmazione di Scratch.
Questi blocchi colorati ricordano i mattoncini assemblabili dei giochi da costruzione e rendono la programmazione di un computer simile alla costruzione di un puzzle.

Scratch

Scratch è un software che, grazie alla presenza di numerosi elementi grafici dall'uso molto intuitivo, consente di scrivere programmi anche senza possedere un'approfondita conoscenza dell'architettura dei computer.

Scratch 2 funziona sia in modalità *offline* (il programma viene scaricato e installato nel computer e poi non è più necessario avere un accesso a Internet per utilizzarlo), sia in modalità *online* (il programma viene utilizzato direttamente dal sito web).

Un programma in Scratch 2



La versione offline di Scratch 2 si scarica da questa pagina: https://scratch.mit.edu/scratch-2download.
Mentre, per utilizzare la versione online di Scratch, si deve andare nella home page: https://scratch.mit.edu e cliccare sul pulsante "Provalo" oppure su "Crea".
Se si utilizza la versione online è consigliabile iscriversi alla comunità di Scratch, cliccando sul pulsante "Unisciti alla comunità di Scratch" che si trova in alto nella home page. Una vol-

ta iscritti sarà possibile accedere cliccando sul pulsante “Entra”.
L’iscrizione al sito fornisce alcuni vantaggi, per esempio il salvataggio automatico dei nostri progetti e la possibilità di condividere i nostri progetti con gli altri utenti di Scratch.

Code.org

Code.org è il sito web dell’omonima organizzazione non-profit che propone lezioni online di programmazione, percorsi e sfide rivolte a studenti e insegnanti.
Le lezioni consistono in brevi video da guardare, istruzioni da leggere e piccole sfide di programmazione da superare.
Per seguire tali lezioni è necessario avere accesso a Internet, poiché l’uso di questo strumento di apprendimento avviene tramite il sito web, a eccezione di alcune parti riguardanti l’informatica *unplugged* (senza computer) che sono dedicate ai più piccoli.
Code.org può essere utilizzato sia da computer sia da tablet (anche se è consigliabile avere un tablet con il display abbastanza grande e dotato di un processore veloce).
Esistono diversi percorsi progressivi, che vengono consigliati in base al grado di conoscenza della programmazioni e all’età.
Alcune attività si basano su videogiochi o film famosi.
Dalla home page: https://code.org/ è possibile accedere a Code.org cliccando sul pulsante “Accedi”, posizionato in alto a destra.

Una pagina del sito web Code.org.

Se non siamo già iscritti, nella nuova pagina che ci viene mostrata è possibile registrarsi cliccando sul pulsante “Iscriviti”.
Nel caso in cui le attività di programmazione devono essere svolte insieme alla classe, è bene, prima di registrarsi, chiedere al professore di riferimento.
Esistono comunque delle attività utilizzabili anche senza essere iscritti, che vengono mostrate sempre nella pagine a cui si arriva premendo il pulsante “Accedi”. È utile provarle per capire come funziona Code.org.

3 Esploriamo Scratch

Figura 3.1: Iniziare un nuovo progetto

Come seguire le esercitazioni

Per seguire questo capitolo, e anche i successivi, è necessario accedere al sito di Scratch o utilizzare la versione offline. Se si sta già utilizzando Scratch, è bene iniziare un nuovo progetto (opzione "Nuovo") dal menu **File** posizionato in alto, al fine di tornare alle impostazioni iniziali (▶ Figura 3.1).

Figura 3.2: Cambiare l'impostazione della lingua

A seconda del tipo di installazione, Scratch potrebbe apparire in lingua inglese. L'icona a forma di mappamondo, posizionata a destra del logo, consente di cambiare lingua (▶ Figura 3.2).

L'interfaccia utente

Esaminiamo ora l'interfaccia grafica di Scratch. Nel mondo del software l'interfaccia grafica di un programma consente a un utente di "interagire" con il software, cioè effettuare delle scelte, compiere delle operazioni, immettere dei dati e così via. Il tutto in maniera "amichevole".
L'interfaccia utente di Scratch è articolata in tre modalità:

- programmazione;
- editor grafico;
- editor sonoro.

Vediamo quali sono le sezioni principali dell'interfaccia grafica di Scratch in modalità programmazione (▶ Figura 3.3).
Come moltissimi altri programmi, anche Scratch ha la classica barra dei menu A, che consente di aprire un file, salvarlo e così via. Ulteriori opzioni verranno esaminate in seguito nel libro.
Il gatto B è uno **sprite**, ovvero un'immagine grafica in grado di muoversi sopra lo sfondo.
L'area bianca C si chiama **stage** e rappresenta il "palcoscenico" su cui verrà visualizzato il nostro programma. Può essere visualizzata a pieno schermo premendo sull'icona D.
Gli sprite possono essere caricati, modificati e così via. Queste azioni vengono attivate tramite il **menu degli sprite** E.
Al punto F vediamo nuovamente il gatto, o meglio, la sua miniatura, che serve a selezionarlo.
Questa sezione si chiama **elenco degli sprite** e tiene nota di tutti gli sprite che vengono utilizzati in un programma. Anche lo stage può essere selezionato e ha la sua miniatura G e il **menu degli sfondi** H, che riguarda i fondali che possono decorare lo stage.
Il **menu degli strumenti** I consente di eseguire con celerità alcune

operazioni sugli sprite o chiedere aiuto.
Per cambiare modalità, passando per esempio dalla modalità programmazione alla modalità editor grafico o editor sonoro, basta fare click sulle **tab** ("linguette") J che si trovano subito sotto il menu degli strumenti. Quando si è in modalità programmazione, la tab evidenziata è "Script".
Scratch comprende circa un centinaio di istruzioni diverse, rappresentate da blocchi colorati. Se fossero visualizzate tutte contemporaneamente sullo schermo, si avrebbe una gran confusione.
Per questo motivo i blocchi sono stati suddivisi in categorie, o "famiglie". Esistono dieci categorie di blocchi, ognuna con il proprio colore che la contraddistingue. Cliccando sull'etichetta di ciascuna **categoria** K, è possibile far apparire i blocchi che le appartengono.
I blocchi disponibili appaiono nella **tavolozza dei blocchi** L.
Se due blocchi appartengono alla stessa categoria, significa che si occupano di uno stesso aspetto delle possibilità di Scratch.
Per costruire un programma, bisogna trascinare le istruzioni dalla tavolozza dei blocchi all'**area degli script** M.
Le icone al punto N ci consentono di ingrandire o ridurre le dimensioni dei blocchi che si trovano nell'area degli script.

Figura 3.3: L'interfaccia grafica di Scratch

4 Primi passi con Scratch

Figura 4.1: Un blocco per spostare il gatto

Figura 4.2: Le dieci categorie di blocchi

Eseguire un blocco

Iniziamo un nuovo progetto, quindi clicchiamo, con il pulsante del mouse, sul blocco *fai 10 passi* della categoria "Movimento" (▶ Figura 4.1). Osserviamo che il gatto si sposta un po' a destra.
In questa maniera non stiamo ancora programmando (non abbiamo costruito un programma), ma stiamo attivando l'esecuzione di una singola istruzione.

Possiamo provare a "navigare" tra le varie categorie di blocchi (▶ Figura 4.2), testandone l'esecuzione di alcuni.
Dopo queste prove è consigliabile iniziare, un'altra volta, un nuovo progetto, senza salvare quanto fatto.

Scratch è visuale

Possiamo immaginare un programma come una serie di istruzioni passo-passo scritte una di seguito all'altra, dall'alto verso il basso.
Solitamente le istruzioni di un programma vengono scritte una a una, utilizzando appositi editor di testo che possono essere più o meno evoluti.
A seconda del linguaggio di programmazione che abbiamo scelto, un software (detto "compilatore" o "interprete") si prenderà cura di tradurre il nostro lavoro in codici eseguibili dalla CPU del computer.
Utilizzando Scratch, gran parte di queste operazioni vengono già svolte per noi. Possiamo concentrarci sul progetto da realizzare.
Inoltre, non è necessario digitare le istruzioni, in quanto sono già disponibili sotto forma di blocchi e possiamo verificarne immediatamente il funzionamento. Questo accade perché Scratch è sia un linguaggio di programmazione sia un ambiente di sviluppo visuale.

Figura 4.3: Le tab delle tre modalità

Figura 4.4: Lo sprite selezionato

Blocchi di istruzioni

Un programma, in Scratch, può contenere numerosi sprite (o anche nessuno) e ciascuno sprite viene programmato singolarmente.
Prima di cominciare a programmare uno sprite, dobbiamo:

- verificare di essere in modalità programmazione, cioè la tab script deve essere attiva (▶ Figura 4.3);
- controllare di aver selezionato lo sprite desiderato, cioè cliccare sulla miniatura che si trova nell'elenco degli sprite e accertarsi che sia bordata di azzurro (▶ Figura 4.4).

Ora possiamo costruire il nostro primo **script**.

Che cos'è uno script

Scrivere un programma, in un certo senso, è come risolvere un problema matematico.
In matematica, uno dei metodi più utilizzati per affrontare con successo un problema è quello di scomporlo in problemi più piccoli e più semplici da risolvere. Anche Scratch adotta questa tecnica.
Un programma scritto nell'ambiente Scratch non è rappresentato da un unico elenco di istruzioni, ma può essere diviso in sequenze di istruzioni più brevi chiamate script.
A ogni sprite possiamo associare più **script**, e anche lo stage può essere programmato. Lo script che prepariamo per **Sprite1** (il gatto, che compare sempre all'avvio del programma) è mostrato in Figura 4.5. Gli script devono essere costruiti nell'area degli script. I blocchi si prendono dalla tavolozza dei blocchi e si agganciano tra di loro avvicinandoli uno alla volta, dal basso verso l'alto (▶ Figura 4.6).

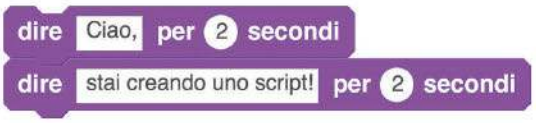


Figura 4.5: Script n. 1 per Sprite1

Figura 4.6: Come agganciare due blocchi

Quando appare il bordo bianco, significa che è possibile rilasciare il tasto del mouse. Per sganciare un blocco da un altro basta cliccarci sopra e trascinarlo verso il basso. Per cancellare un blocco basta trascinarlo nuovamente nella tavolozza dei blocchi.
Il nostro script utilizza per due volte lo stesso blocco, *dire «Hello!» per 2 secondi*, che si trova al primo posto della categoria "Aspetto". In questa categoria si trovano quei blocchi che consentono, per esempio, di manipolare l'aspetto di uno sprite o di fargli "dire" o "pensare" qualcosa attraverso dei fumetti che vengono visualizzati sullo stage. Rispetto al blocco originale, abbiamo modificato le scritte. Per farlo, abbiamo cliccato nella casellina bianca dove c'è scritto «Hello!» e abbiamo digitato i nuovi testi.

Eseguire uno script

A questo punto non succede ancora nulla, poiché Scratch si aspetta un "segnale" che gli comunichi di eseguire lo script.
Se clicchiamo sullo script, lo possiamo attivare. Attorno allo script apparirà un alone giallo (che indica che lo script è in esecuzione), mentre potremo vedere il gatto "dire" quello che abbiamo scritto (▶ Figura 4.7). Ovviamente risulta un po' scomodo eseguire gli script in questa maniera, specie se ne dobbiamo mandare in esecuzione più di uno. Nel prossimo capitolo vedremo come attivare l'esecuzione degli script tramite gli "eventi".

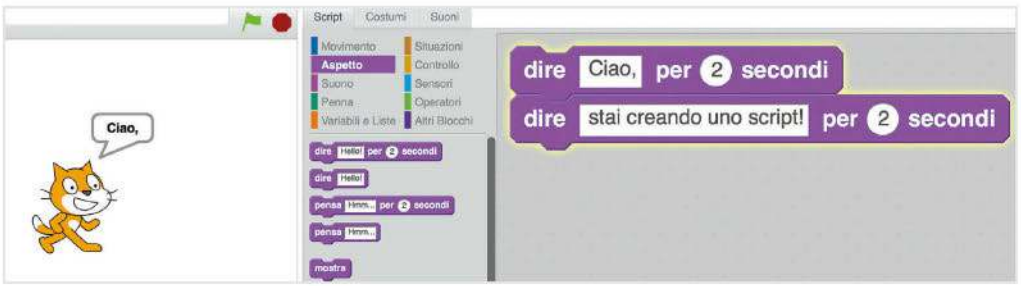


Figura 4.7: Lo script è in esecuzione

5 Eventi e movimenti

Figura 5.1: La categoria "Situazioni"

Figura 5.3: Le icone per far partire e arrestare gli script

Figura 5.4: Script n. 2 per Sprite1

Figura 5.5: Le opzioni di un menu a tendina

Gestire un evento

Ripartiamo dallo script creato nel capitolo precedente e spieghiamo come collegare l'esecuzione di uno script al verificarsi di un evento. Clicchiamo sulla categoria "Situazioni" e notiamo che ci sono dei blocchi a forma di "cappello" (▶ Figura 5.1).

Trasciniamo il primo blocco, *quando si clicca su (bandierina verde)*, sopra lo script creato in precedenza (▶ Figura 5.2 a | b).

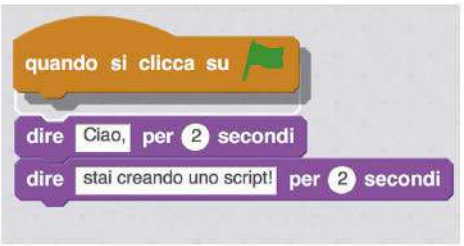


Figura 5.2.a: Script n.1 per Sprite1

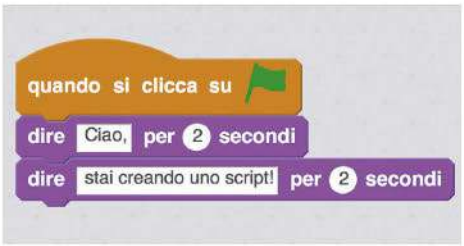


Figura 5.2.b: Lo script n. 1 è completo

Il nostro script ora verrà eseguito ogni volta che l'utente farà un click sull'icona a forma di bandierina verde collocata sopra lo stage, a destra (▶ Figura 5.3).
Il pulsante rosso a forma di stop serve, invece, a fermare l'esecuzione di tutti gli script del programma.

Aggiungere uno script

Possiamo aggiungere un secondo script allo sprite del gatto, per esempio uno script che controlla se viene premuto il tasto **A** della tastiera e, nel caso, ruota il gatto di 3 gradi (▶ Figura 5.4).

Per creare questo secondo script abbiamo utilizzato il blocco *ruota di (senso orario) 15 gradi* dalla categoria "Movimento" e abbiamo sostituito il valore 15 con il valore 3.

Il blocco della categoria "Situazioni" che abbiamo usato per gestire la pressione del tasto A è il blocco *quando si preme il tasto (spazio)*.

Notiamo che questo blocco presenta un piccolo triangolo nero. Cliccandoci sopra appare un menu a tendina che ci consente di scegliere tra le varie opzioni (▶ Figura 5.5). Entrambi gli script devono apparire nell'area degli script di **Sprite1** (▶ Figura 5.6). Se proviamo a cliccare sulla bandierina verde con il

Figura 5.6: I due script sullo stage

Figura 5.7: Il gatto parla mentre ruota

mouse, tenendo premuto il tasto **A** della tastiera, vedremo il gatto ruotare mentre ci saluta (▶ Figura 5.7).

La forma dei blocchi

Abbiamo già visto che i blocchi di Scratch sono caratterizzati dal colore: il colore di un blocco determina la sua appartenenza a una delle dieci categorie.

Abbiamo certamente già notato che i blocchi non hanno tutti la stessa forma. Ne esistono, infatti, di quattro forme differenti.

- **Blocchi di comando**
  Sono i blocchi più comuni, di forma rettangolare, e servono a impartire comandi diretti. Un blocco di questo tipo è *fai 10 passi*, della categoria "Movimento" (▶ Figura 5.8).
  Se ci clicchiamo sopra, infatti, il gatto si sposta in avanti (chiariremo in seguito il concetto di "passi").

Figura 5.8: Blocco che muove uno sprite

- **Blocchi funzione**
  Sono blocchi di forma rettangolare con gli angoli arrotondati oppure con i lati appuntiti e servono a farci sapere quanto vale una certa quantità. Tecnicamente si dice che "ritornano" un valore, nel senso che calcolano un valore (tramite una funzione matematica) e ce lo restituiscono. Un blocco di questo tipo è *posizione x*, sempre della categoria "Movimento". Se ci clicchiamo sopra ci dice qual è la coordinata x del gatto (▶ Figura 5.9).

Figura 5.9: Blocco che indica la coordinata x di uno sprite

- **Blocchi e graffette**
  Alcuni blocchi della categoria "Controllo", come quello di Figura 5.10, sono aperti. Possono così contenere altri blocchi decidendo quando eseguirli e quante volte eseguirli.

Figura 5.10: Blocco che ripete altri blocchi

- **Blocchi attivatori**
  Sono blocchi che abbiamo già incontrato e sono importantissimi, perché servono ad attivare i vari script di cui si compone un programma. Sono caratterizzati da una specie di cappello, come a indicare che devono andare in testa al resto dello script. Si trovano nella categoria "Situazioni". Quando si verifica l'evento che è scritto nel cappello, allora parte l'esecuzione di tutti i blocchi che ci sono attaccati sotto. Il blocco *quando si clicca questo sprite* è un blocco attivatore (▶ Figura 5.11).

Figura 5.11: Blocco che attiva i blocchi attaccati sotto

Muovere uno sprite tramite i tasti

L'importanza dei blocchi attivatori può essere evidenziata con il piccolo programma che andiamo a costruire.
Prima di partire, iniziamo un nuovo progetto dalla barra dei menu.
Costruiamo quindi, per lo sprite del gatto (**Sprite1**), i quattro script mostrati in Figura 5.12.

Figura 5.12: Quattro script per Sprite1

Il blocco *quando si preme il tasto [ ]* lo conosciamo già (categoria "Situazioni"), mentre i blocchi *cambia x di ( )* e *cambia y di ( )* si trovano nella categoria "Movimento".
Una volta terminato questo lavoro di programmazione, è possibile far muovere lo sprite del gatto sullo stage utilizzando i tasti freccia della tastiera.
Possiamo aggiungere ora un quinto script (▶ Figura 5.13). Il blocco *vai a x: ( ) y: ( )* si trova nella categoria "Movimento".
Se testiamo nuovamente il programma, vediamo che, ovunque si trovi il gatto, premendo il tasto spazio lo sprite viene riportato al centro dello stage.

Figura 5.13: Script n. 5 per Sprite1

Stage e coordinate

Si sarà intuito che Scratch utilizza un sistema di riferimento cartesiano. Lo stage, quindi, può essere visto come un vero e proprio piano cartesiano (▶ Figura 5.14).

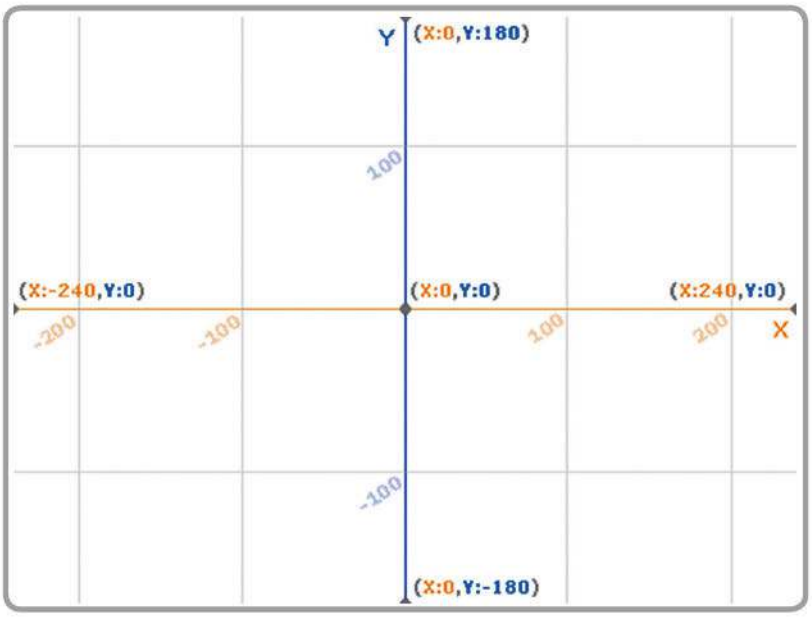


Figura 5.14: Coordinate dello stage

La coordinata *x* di uno sprite serve a indicarne la posizione orizzontale, la coordinata *y* quella verticale.
Scratch adotta un'unità di misura particolare, i passi.
Lo stage è largo 480 passi e alto 360 passi.
Pertanto, uno sprite posizionato tutto a sinistra avrà una coordinata *x* uguale a −240. Se posizionato tutto a destra *x* sarà uguale a 240.
L'asse verticale, invece, presenta i valori da −180 (in basso) a 180 (in alto).
Non esiste un rapporto fisso tra i passi di Scratch e i pixel del monitor. Ci pensa Scratch ad adattare la visualizzazione su qualsiasi display.

Spostamento assoluto e spostamento relativo

In questo capitolo abbiamo usato i blocchi *cambia x di ( )*, *cambia y di ( )* e *vai a x: ( ) y: ( )*. Appartengono tutti e tre alla categoria "Movimento", ma sono leggermente diversi.
I primi due comandano un movimento di tipo relativo.
Il blocco *cambia x di 5*, per esempio, prende il valore attuale della coordinata x dello sprite e lo aumenta di 5 passi.
Il blocco *vai a x: 0 y: 0* comanda un movimento "assoluto". Nel senso che, ovunque si trovi lo sprite in quel momento, lo porta al centro dello stage.
In Figura 5.15 sono riportati i blocchi per il movimento assoluto (A) e quelli per il movimento relativo (B).

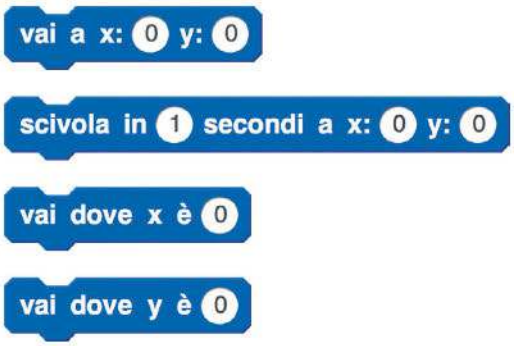


Figura 5.15.a: Blocchi per il movimento assoluto

Figura 5.15.b: Blocchi per il movimento relativo

Possiamo aggiungere un ulteriore script allo sprite del gatto, come quello mostrato in Figura 5.16, che utilizza blocchi che abbiamo già visto.

Con questo script, a seguito della pressione del tasto B, il gatto si posizionerà in alto a sinistra dello stage e poi si sposterà toccando tutti i quattro angoli e ritornando al punto di partenza.
Si può notare come i blocchi vengano eseguiti uno di seguito all'altro partendo da quello più in alto.

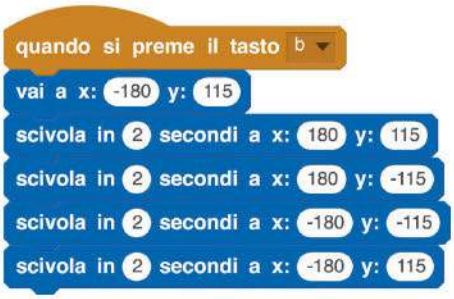


Figura 5.16: Script n. 6 per Sprite1

6 Script più compatti

Figura 6.1.a

Figura 6.1.b

Figura 6.1.c

Figura 6.2.a

Figura 6.2.b

Blocchi e valori

Abbiamo visto che esistono dei blocchi che hanno al loro interno una o più caselline bianche, oppure dei menu a tendina.
Ciò significa che quei blocchi accettano degli input (talvolta detti "**argomenti**").
La possibilità di passare dei valori a un blocco è importante, perché ci consente di utilizzare lo stesso blocco, per esempio, sia per far ruotare uno sprite di 15 gradi, sia per farlo ruotare di 90 gradi.
Nella Figura 6.1 vengono mostrati tre blocchi differenti:

- nel blocco (a) bisogna digitare un valore con la tastiera;
- nel blocco (b) si può sia digitare un valore sia sceglierne uno dal menu a tendina;
- nel blocco (c) si può solo scegliere un valore tra le opzioni del menu a tendina.

Un'ulteriore differenza si ha nella forma delle caselline bianche (▶ Figura 6.2):

- se presentano angoli arrotondati (a), significa che accettano solo valori numerici;
- se presentano angoli squadrati (b), significa che accettano valori testuali.

Facciamo una prova e ci accorgiamo che non è, ovviamente, possibile, scrivere delle lettere nella casellina bianca di un blocco *fai ( ) passi*.
Nelle caselline bianche è anche possibile incastrare dentro un blocco. Possiamo, cioè, utilizzare i blocchi funzione visti nel capitolo 5 (quelli che ci "ritornano" un valore) per calcolare qualcosa da passare a un altro blocco.
Proviamo con uno script d'esempio. Iniziamo un nuovo progetto e costruiamo, per lo sprite del gatto, i due script mostrati in Figura 6.3.
Il primo script (a) calcola la somma di 11 e 12 e poi la fa "dire" al gatto. Il secondo script (b) moltiplica 4 per 5 e poi ruota il gatto esattamente di 20 gradi. In entrambi gli script è stato usato un blocco verde preso dalla categoria "Operatori". In questo caso si tratta di operatori matematici.
Cancelliamo gli script precedenti (basta trascinarli sopra la tavolozza

Figura 6.3.a: La scritta che compare è la somma dei due addendi

Figura 6.3.b: Il numero di gradi è determinato dal risultato della moltiplicazione

dei blocchi e lasciarli là) e costruiamo un nuovo script per **Sprite1** (▶ Figura 6.4).

Con il comando "Aspetto" *porta dimensione al ( ) %* facciamo occupare al gatto meno spazio sullo stage.
Il blocco "Movimento" *punta in direzione 90* ci permette di reimpostare la direzione del gatto verso destra.

Con il blocco "Controllo" *attendi 1 secondi*, Scratch attende un secondo esatto prima di eseguire il blocco successivo.
Se clicchiamo sulla bandierina verde e testiamo il programma, vediamo che il gatto segue, idealmente, il perimetro di un triangolo.

Figura 6.4: Script per **Sprite1**

Evitare di ripetere il codice

Osservando meglio lo script proposto, ci accorgiamo che c'è un gruppo di blocchi che si ripete. Un vero programmatore non ama le ripetizioni di questo tipo, che spesso sono fonte di errori, e le gestisce in un'altra maniera.

È possibile trasformare lo script precedente in quello di Figura 6.5, che è molto più compatto e leggibile.

Abbiamo usato il blocco *ripeti ( ) volte*, presente nella categoria "Controllo". Questo blocco, che ha una forma simile a una graffetta, può contenere altri blocchi, che vanno inseriti da destra verso sinistra (▶ Figura 6.6).

Figura 6.5: Un blocco che ripete l'esecuzione di altri blocchi

Dal punto di vista informatico, in quest'ultimo script abbiamo utilizzato un ciclo.

Figura 6.6: Come inserire dei blocchi all'interno di un blocco a graffetta

Avvertenza

La direzione degli sprite

Quando usiamo il blocco *fai 10 passi*, lo sprite si muove di dieci passi nella direzione in cui sta puntando.
All'inizio di un progetto il gatto punta sempre in direzione 90 (che per Scratch significa verso destra), ma se usiamo il blocco *punta in direzione...* con un valore diverso da 90 oppure facciamo ruotare il gatto attraverso i due blocchi *ruota di ( ) gradi* o, ancora, usiamo il blocco *punta verso...* la direzione può cambiare. Per convenzione Scratch usa i seguenti valori per indicare le quattro direzioni principali:
- 0 verso l'alto;
- 90 verso destra;
- –90 verso sinistra;
- 180 verso il basso.

7 Sprite e costumi

Figura 7.1: Sprite selezionato

Sprite, non solo personaggi

Gli sprite sono oggetti grafici che appaiono e si muovono sopra lo stage e sono caratterizzati da alcuni attributi (dimensione, la direzione ecc.). Alcuni di questi attributi possono essere modificati durante l'esecuzione del programma, utilizzando i blocchi di istruzioni. Altri si modificano quando il programma non è in esecuzione.

Figura 7.2: Il menu informazioni

Menu informazioni

Quando iniziamo un nuovo progetto, sappiamo che lo sprite del gatto è selezionato perché vediamo il bordo azzurro intorno alla sua miniatura (▶ Figura 7.1).

In alto, a sinistra della miniatura, c'è una piccola «i» bianca su sfondo azzurro. Se ci clicchiamo sopra, appare il **menu informazioni** relativo allo sprite (▶ Figura 7.2). Da questo menu possiamo:

- tornare all'elenco degli sprite A;
- cambiare il nome dello sprite B;
- conoscere le sue coordinate sullo stage C;
- modificare la sua direzione D;
- modificare lo stile di rotazione (definire, cioè, se il costume dello sprite deve apparire riflesso o ruotato o rimanere invariato quando lo sprite cambia direzione) E;
- decidere se lo sprite è trascinabile nel player (cioè se possiamo spostarlo con il mouse quando lo stage viene visualizzato a pieno schermo) F;
- decidere se lo sprite è visibile G.

Figura 7.3: Si può cancellare uno sprite anche dalla sua miniatura

Gestire più sprite

Un programma di Scratch può contenere zero, uno o molti sprite. Proviamo a cancellare lo sprite del gatto, cliccandoci sopra con il pulsante destro del mouse e scegliendo l'opzione "cancella"(▶ Figura 7.3).

Carichiamo quindi un nuovo sprite, prendendolo dalla libreria degli sprite di Scratch.

Figura 7.4: Il menu degli sprite ha quattro icone

Per visualizzare la libreria dobbiamo cliccare la prima icona ("Scegli uno sprite dalla libreria") del **menu degli sprite**, che si trova subito sotto lo stage (▶ Figura 7.4).

Nella **libreria degli sprite** (▶ Figura 7.5) ci sono numerosi oggetti grafici già pronti. Se stiamo cercando qualcosa in particolare, possiamo usare i filtri a sinistra.
Carichiamo, dalla Categoria "Persone", lo sprite **Avery Walking** (▶ Figura 7.6). Per caricarlo ci clicchiamo sopra due volte oppure una volta sola e poi clicchiamo su "OK". Una volta che abbiamo caricato lo sprite, verifichiamo che sia selezionato e clicchiamo sulla tab "Costumi" per entrare nell'editor grafico. Nell'editor grafico vediamo che lo sprite **Avery Walking** ha quattro costumi (▶ Figura 7.7). Ogni sprite può avere da uno a più costumi, che ne caratterizzano la rappresentazione grafica. Ciascun costume ha anche un nome, che può essere modificato, e un numero progressivo. Questi dati servono a identificarlo. I costumi di **Avery Walking**, se visualizzati uno di seguito all'altro, danno l'idea che lo sprite muova le gambe per camminare, come nella tecnica dei cartoni animati.

Figura 7.5: la libreria degli sprite

Torniamo quindi in modalità programmazione con un click sulla tab "Script". Costruiamo, per questo sprite, i due script mostrati in Figura 7.8. In entrambi gli script abbiamo usato un blocco "Controllo" *per sempre* che ripete in continuazione i blocchi che vengono inseriti al suo interno. Nel primo script vediamo che, ogni "0.3" secondi (che corrisponde a 0,3 poiché in Scratch il punto equivale alla virgola), lo sprite indossa il costume successivo, grazie al blocco "Aspetto" *passa al costume seguente*. Questo blocco opera in maniera circolare: quando i costumi dello sprite sono terminati, si riparte da quello iniziale.
Il secondo script, invece, serve a controllare il movimento orizzontale della ragazza. Il blocco "Movimento" *porta stile di rotazione a [sinistra-destra]* spiega a Scratch che, quando lo sprite cambia direzione da verso destra a verso sinistra, il costume dello sprite deve apparire riflesso in senso orizzontale.

Figura 7.6: Lo sprite **Avery Walking**

Figura 7.7: I costumi dello sprite

Figura 7.8: I due script per **Avery Walking**

Duplicare uno sprite

Se vogliamo avere due ragazze che camminano sullo stage, possiamo duplicare lo sprite cliccandoci sopra con il tasto destro del mouse e scegliendo l'opzione "duplica" (▶ Figura 7.9). Un nuovo sprite apparirà nell'elenco degli sprite, con un nome leggermente diverso (**Avery Walking2**). Questo nuovo sprite avrà una propria area degli script, completamente autonoma da quella del primo sprite, e potrà quindi essere programmato in maniera totalmente indipendente.
Per verificare quanto sopra, possiamo provare a far camminare questa ragazza più velocemente o più lentamente della prima, modificando i valori nei blocchi *attendi ( ) secondi* e *fai ( ) passi*.

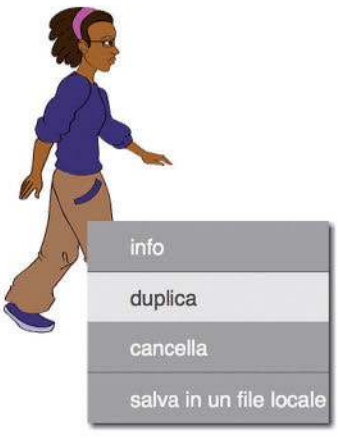


Figura 7.9: Duplicare uno sprite

8 Stage e sfondi

Cosa succede in città

Programmare lo stage

Figura 8.1: La miniatura dello stage è selezionata

Figura 8.2: Carichiamo gli sfondi dall'editor grafico

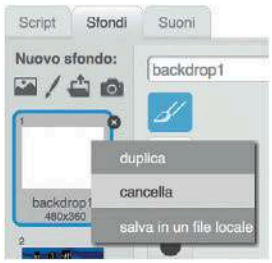


Figura 8.3: Cancellazione dello sfondo bianco

Figura 8.4.a: Script n. 1 per lo stage

Figura 8.4.b: Script n. 2 per lo stage

Lo stage funziona in maniera simile a uno sprite. Può "indossare" diversi sfondi e può essere programmato, anche se possiede un set (insieme) di istruzioni abbastanza ridotto.
Vogliamo costruire un programma che ci illustri come funzionano i servizi pubblici di una città, in due semplici schermate. Nella prima schermata si vede un autobus che corre, nella seconda un camion attrezzato che pulisce l'asfalto.
Iniziamo un nuovo progetto, cancelliamo il gatto e selezioniamo lo stage cliccando sopra la sua miniatura (▶ Figura 8.1).
Clicchiamo sulla tab "Sfondi" e carichiamo, dal tema "Città", gli sfondi "night city with street" e "urban2" (▶ Figura 8.2).

Possiamo cancellare lo sfondo bianco "backdrop1" che non ci serve (▶ Figura 8.3).
Torniamo in modalità programmazione (tab "Script") e, sempre con lo stage selezionato, costruiamo i due script di Figura 8.4.
Testiamo il programma premendo alternativamente il tasto A e il tasto B. Lo sfondo cambierà di conseguenza, con un accattivante effetto grafico.

Avvertenza

"Porta" o "cambia"?

In Scratch alcuni blocchi usano il termine "porta", altri il termine "cambia". Esiste una sostanziale differenza tra i due. Il blocco che abbiamo usato, **porta effetto [effetto pixel] a 0**, per esempio, serve a impostare a 0 l'effetto pixel dello stage, qualunque sia il suo valore precedente. Il blocco **cambia effetto [effetto pixel] di 20**, invece, serve a incrementare il valore di questo effetto. Se in precedenza era 0, allora passa a 20. Se era 20, allora passa a 40 e così via.

Aggiungere uno sprite

Dalla categoria "Trasporti" della libreria degli sprite, carichiamo lo sprite **Bus**. Lo selezioniamo e clicchiamo sulla tab "Costumi" e quindi su "Scegli un costume dalla libreria" (▶ Figura 8.5). In questa maniera vediamo tutti i costumi di tutti gli sprite. Carichiamo "street-cleaner-mit" dalla categoria "Trasporti".

Uso dei messaggi di Scratch

I singoli script appartengono sempre o a uno sprite o allo stage. Il miglior metodo per coordinare le azioni di stage e sprite consiste nell'uso dei messaggi. Dalla modalità programmazione, prepariamo tre script per lo sprite **Bus** (▶ Figura 8.6).
I blocchi per gestire i messaggi si trovano nella categoria "Situazioni". Per creare un nuovo messaggio basta scegliere l'opzione "nuovo messaggio" dal menu a tendina di uno di questi blocchi e poi dare un nome al messaggio (▶ Figura 8.7).

Figura 8.5: Caricare altri costumi

Figura 8.6.a: Script n. 1 per **Bus**, che muove il costume "bus"

Figura 8.6.b: Script n. 2 per **Bus**, che muove il costume "street-cleaner-mit"

Figura 8.6.c: Script n. 3 per Bus, nasconde lo sprite durante il cambio della schermata

Figura 8.7: Creazione di un messaggio

Dobbiamo ora selezionare lo stage e ritoccare i suoi due script come mostrato in Figura 8.8. È giunto il momento di studiare i servizi pubblici della nostra città (▶ Figura 8.9).

Figura 8.8.a: Script n. 1 (modificato) per lo stage

Figura 8.8.b: Script n. 2 (modificato) per lo stage

Figura 8.9: Ovviamente possiamo inserire altri sfondi e altri macchinari usando le tecniche imparate

9 L'ordine delle istruzioni

Smistamento frutta

Flusso di esecuzione

Abbiamo visto che Scratch esegue le istruzioni di ciascuno script partendo da quella più in alto e poi spostandosi verso il basso, finché non rimane più alcun blocco.
Abbiamo anche imparato a usare i blocchi *ripeti ( ) volte*, *per sempre* e *attendi ( ) secondi* che modificano il normale **flusso di esecuzione** del programma. Useremo ora altri blocchi di questo tipo, presi anch'essi dalla categoria "Controllo".

Smistamento frutta

Vogliamo costruire un programma che simuli il funzionamento di un impianto automatizzato per lo smistamento della frutta. Alcune mele scorreranno su un nastro, e un sensore le riconoscerà in base al colore.
Iniziamo un nuovo progetto e cancelliamo il gatto.
Carichiamo, dalla libreria, lo sprite **Apple** (categoria "Cose") e lo selezioniamo.

Figura 9.1: I due costumi di **Apple**

Clicchiamo sulla tab "Costumi" e duplichiamo il costume "apple", modificandolo un po', per esempio colorandolo di giallo. Ora **Apple** ha due costumi.
Cambiamo nome a questi due costumi, chiamandoli "mela rossa" e "mela gialla" (▶ Figura 9.1).

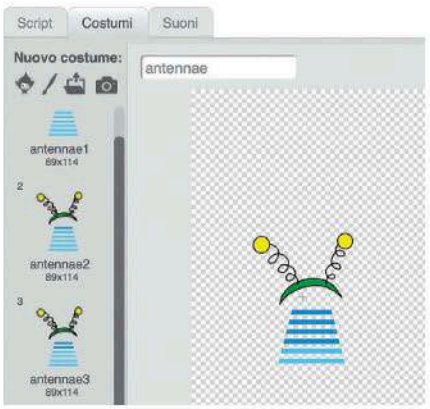


Figura 9.2: Diversi costumi per **Antennae**

Lasciamo lo sfondo bianco (o disegniamone uno a piacere; l'editor grafico è brevemente illustrato nel capitolo 16).

Carichiamo un secondo sprite, **Antennae**, sempre dalla categoria "Cose". Possiamo modificarlo graficamente per aggiungerci una specie di raggio che funzionerà da "sensore ottico" in grado di distinguere tra mele rosse e mele gialle (▶ Figura 9.2).

Programmiamo il "sensore ottico"

Costruiamo due script per **Antennae**. Il primo script serve solamente a gestire l'animazione del sensore (▶ Figura 9.3), nel caso in cui gli avessimo creato più costumi.

Figura 9.3: Script n. 1 di **Antennae**

Il secondo script di **Antennae** è più complesso (▶ Figura 9.4). Contiene, infatti, diverse strutture di controllo annidate, cioè blocchi "Controllo" uno dentro l'altro.

Il blocco *se... allora* esegue i blocchi che ha al suo interno, solo se una certa condizione risulta vera. Questa condizione (che può essere **vera** o **falsa**) viene inserita nella piccola nicchia esagonale (▶ Figura 9.5).

Il blocco *sta toccando [Apple]* è preso dalla categoria "Sensori" ed è un blocco funzione che restituisce un valore "booleano", cioè può assumere solamente i valori **vero** oppure **falso**.

Il *blocco se... allora / altrimenti* è simile a *se... allora*, ma se la condizione nella nicchia risulta falsa, allora vengono eseguiti i blocchi che sono all'interno della seconda "graffetta" (▶ Figura 9.6).

Figura 9.5: Inserimento della condizione nella nicchia

Figura 9.6: Questo blocco ha due graffette

Il blocco **Sensori** *sta toccando il colore...* (▶ Figura 9.7) verifica se è in corso una collisione tra lo sprite **Antennae** e il colore presente nella casellina quadrata. Per cambiare questo colore bisogna cliccare prima sulla casellina e poi sul colore che si vuole controllare.

Noi dobbiamo controllare il colore rosso della mela e il colore giallo della mela.

Programmiamo la frutta

Ci aspettiamo che sul rullo trasportatore passino molte mele, non solo una. Possiamo "duplicare" uno sprite durante l'esecuzione del programma usando la tecnica della **clonazione**.

I blocchi che controllano la clonazione si trovano nella categoria "Controllo".

Costruiamo un primo script per **Apple**, come mostrato in Figura 9.8 Questo script attende che venga premuto il tasto spazio prima di continuare, utilizzando il blocco "Controllo" *attendi fino a quando...* e il blocco "Sensore" *tasto [spazio] premuto*. Dopo che è stato premuto il tasto spazio, lo script clona lo sprite **Apple** ogni secondo.

Il secondo script di **Apple** (▶ Figura 9.9) si attiva ogni volta che questo sprite viene clonato. Ogni clone viene spostato verso destra finché la sua coordinata x supera il valore di 240. A questo punto il clone viene eliminato.

Ora non ci resta che provare se il nostro impianto di smistamento della frutta funziona (▶ Figura 9.10).

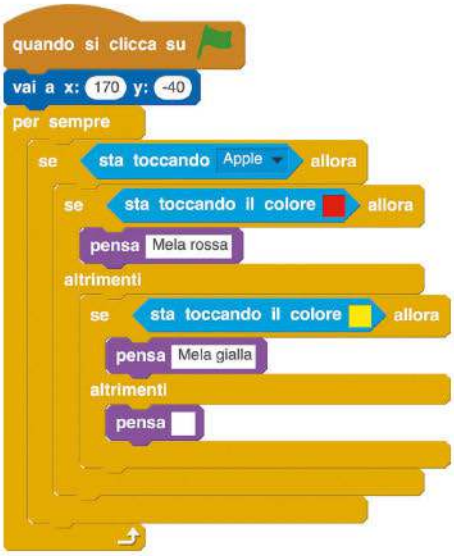


Figura 9.4: Script n. 2 di **Antennae**

Figura 9.7: La casellina va cliccata se vogliamo controllare un altro colore

Figura 9.8: Script n. 1 per **Apple**

Figura 9.9: Script n. 2 per **Apple**

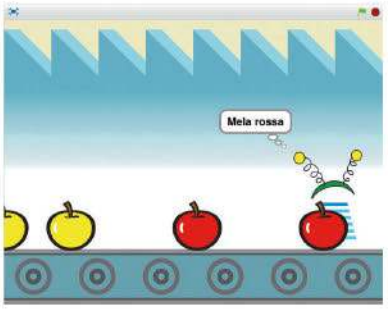


Figura 9.10: L'impianto è in funzione

10 Operatori logici

Figura 10.1: Addizione con due addendi

Svolgere operazioni

Selezioniamo la categoria "Operatori" e vediamo che i primi quattro blocchi di questa categoria sono in grado di eseguire le quattro operazioni fondamentali dell'aritmetica.

Ognuno di questi blocchi contiene due caselline tonde, che accettano quindi solo valori numerici.

Realizzare una somma con Scratch è facile. 7 + 2, per esempio, viene realizzato con il blocco di Figura 10.1.

Nelle caselline bianche, oltre a digitare numeri interi o con decimali (separati dal punto), possiamo trasportare dentro altri operatori.

Costruiamo ora un'espressione più complessa,
per esempio [(3 + 2) * 4] / (5 – 1).
Costruiamo la moltiplicazione (3 + 2) * 4.

Con un operatore di addizione costruiamo 3 + 2	
Con un blocco di sottrazione costruiamo 5 – 1	

Prendiamo un operatore di moltiplicazione (▶ Figura 10.2).

	
A sinistra mettiamo il blocco (3 + 2) e a destra digitiamo 4	
Prendiamo un operatore di divisione	
	
A sinistra mettiamo il blocco [(3 + 2) * 4]	
A destra mettiamo il blocco (5 – 1)	

Figura 10.2: L'espressione è completata

Confrontare due valori

Gli **operatori di confronto** raffrontano i due valori immessi nelle caselline e restituiscono **vero** oppure **falso** (▶ Figura 10.3).

Figura 10.3.a:
Operatore "minore di"

Figura 10.3.b:
Operatore "uguale a"

Figura 10.3.c:
Operatore "maggiore di"

Logica

I blocchi di Figura 10.4 si chiamano **operatori booleani**.

Figura 10.4.a
AND (congiunzione logica). Restituisce **vero** solo se entrambe le condizioni sono vere, altrimenti restituisce **falso**

Figura 10.4.b
OR (disgiunzione logica inclusiva). Restituisce **vero** solo se almeno una delle due condizioni è vera, altrimenti restituisce **falso**

Figura 10.4.c
NOT (negazione logica). Restituisce **vero** se la condizione è falsa e viceversa

Sfere in libertà

Sfere

Per meglio verificare quanto appena spiegato, si può studiare il funzionamento del progetto "Sfere".
In questo programma si utilizza lo sprite **Ball**, che rappresenta una sfera che parte da un punto casuale e si muove in direzione, sempre casuale, sullo stage (▶ Figura 10.5).

Il primo script (▶ Figura 10.6) fa apparire **Ball** e lo clona in continuazione. I cloni hanno una posizione di partenza e direzione casuali.
Il secondo script (▶ Figura 10.7) gestisce i vari cloni, verificando tre condizioni:

- se il tasto spazio non è premuto, il clone avanza di 8 passi (in caso contrario il clone sta fermo);
- se il clone si trova a destra e in alto rispetto al centro, allora indossa il costume giallo, altrimenti indossa quello azzurro;
- se il clone sta toccando il bordo o se sta toccando il puntatore del mouse, il clone viene eliminato.

Avvertenza

Il blocco "per sempre"

Perché quando verifichiamo una condizione usiamo spesso il blocco **per sempre**? Perché altrimenti il controllo avverrebbe una volta sola e il programma terminerebbe.

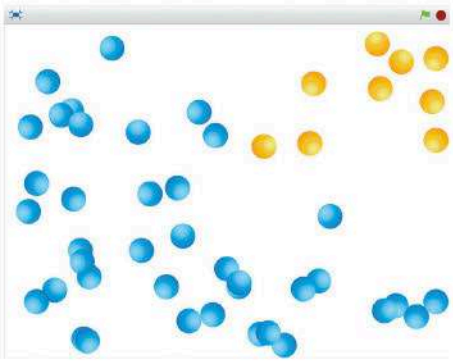


Figura 10.5: Sfere in movimento

Figura 10.6: Script n. 1 di **Ball**

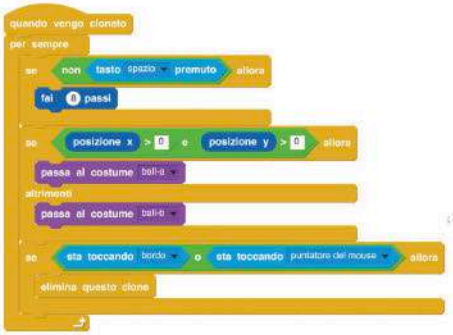


Figura 10.7: Script n. 2 di **Ball**

11 Variabili

Controllo del traffico

In questo capitolo ci caliamo nei panni del nostro sindaco.
Per garantire una maggiore sicurezza agli studenti della città, ha pensato di implementare un sistema di controllo del traffico in prossimità dell'edificio scolastico.
Prima di prendere qualsiasi decisione, comunque, vuole acquisire quanti più dati possibili sul passaggio delle automobili sulle strade che si trovano davanti alla scuola.

Figura11.1: Script per **Convertible3**

Programmazione dello sprite

Iniziamo un nuovo progetto e cancelliamo il gatto.
Carichiamo, dalla libreria degli sfondi, lo sfondo "School2" (categoria "Esterni").
Carichiamo poi, dalla libreria degli sprite, **Convertible3** (sempre categoria "Trasporti").
Creiamo, per **Convertible3**, lo script di Figura 11.1.
Questo script posiziona l'automobile nel punto di coordinate $x = -330$ e $y = -95$. Poi la fa muovere verso destra finché non supera il punto di coordinate $x = 330$ con y che rimane invariata.
Sappiamo che la coordinata x sullo stage assume i valori tra -240 e 240. Abbiamo usato dei valori esterni a questo intervallo in maniera tale da far uscire quasi completamente l'automobile dallo stage.

Figura 11.2: Creazione della variabile

Figura 11.3: Diamo un nome alla variabile

Figura 11.4: Blocchi per gestire le variabili

Contare le automobili con una variabile

Abbiamo ora la necessità di contare i mezzi che passano. Ci serve uno strumento con cui memorizzare un valore che successivamente possiamo anche cambiare.
Le variabili servono a questo scopo. Una variabile è un dato conservato nella memoria del computer.
Per essere correttamente identificata, a ciascuna variabile viene assegnato un nome.
Il valore di una variabile può essere letto e anche modificato.
Per creare una variabile con Scratch selezioniamo la categoria "Variabili e Liste" e clicchiamo su "Crea una Variabile"(▶ Figura 11.2).
Diamo il nome contatore auto alla variabile e lasciamo la scelta su "Per tutti gli sprite"(▶ Figura 11.3). Questo significa che tutti gli sprite possono leggere o modificare il valore di questa variabile.
Dopo la creazione di una variabile la categoria "Variabili e Liste" si popola di nuovi blocchi (▶ Figura 11.4).