Scratch

Un software ***open source***, più precisamente un **ambiente di programmazione visuale**, sviluppato da un gruppo di ricerca presso il Multimedia Lab del MIT di Boston.

Consente di programmare il computer per **risolvere problemi** e creare simulazioni, animazioni, storie interattive, grafica, oggetti artistici in generale potendoli condividere nel Web[1].

In pratica usa dei mattoncini simili al Lego o ai pezzi del puzzle (gli Scratch blocks) *per costruire* ***progetti multimediali*** *che mettono assieme immagini, suoni, video ecc. Il suo nome deriva proprio dalla tecnica dei* disk jockey hip-hop *che mixano i dischi facendoli ruotare con le mani e l'obiettivo fondamentale è quello di* ***avvicinare alla programmazione e capire la logica degli algoritmi*** (prima dell'uso di *codice* di un linguaggio rigoroso[2]) *oltre a sviluppare* ***abilità creative*** *nell'uso dei computer.*

La *codifica* dei programmi in Scratch consiste nell'impilare, incastrandoli, oggetti grafici che presentano forma e colore dipendenti dall'istruzione che si vuole usare.

E' disponibile per diversi Sistemi Operativi (Mac, Windows, Linux - distribuzione Ubuntu); è possibile scegliere tra diverse lingue, tra cui l'italiano, per i nomi dei mattoncini che poi servono a costruire i ***programmi****.*

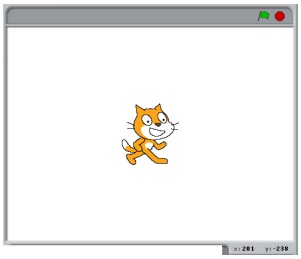
Il suo uso è abbastanza semplice, ma per una introduzione più facile potrebbero essere utili i due videotutorial online di Romolo Pranzetti:

Tutorial 1 per una illustrazione a partire dal download http://www.comeweb.it/doc/scratch/1/scratch_1.swf

Tutorial 2 prosegue con l'illustrazione delle procedure http://www.comeweb.it/doc/scratch/2/Scratch_2.swf

Termini dell'ambiente di lavoro Scratch

Sprite o folletto: in *grafica informatica* (cioè con uso di computer) lo sprite è una figura bidimensionale che può essere spostata rispetto allo sfondo. Nei progetti Scratch sono gli oggetti grafici (il disegno in figura o altro) su cui agiscono i programmi.



Stage o palcoscenico: zona dove ciò che si crea con Scratch, "prende vita" cioè dove viene visualizzata l'esecuzione

Script: insieme di comandi e procedure

singolo comando:
muovi di 10 pixel verso destra

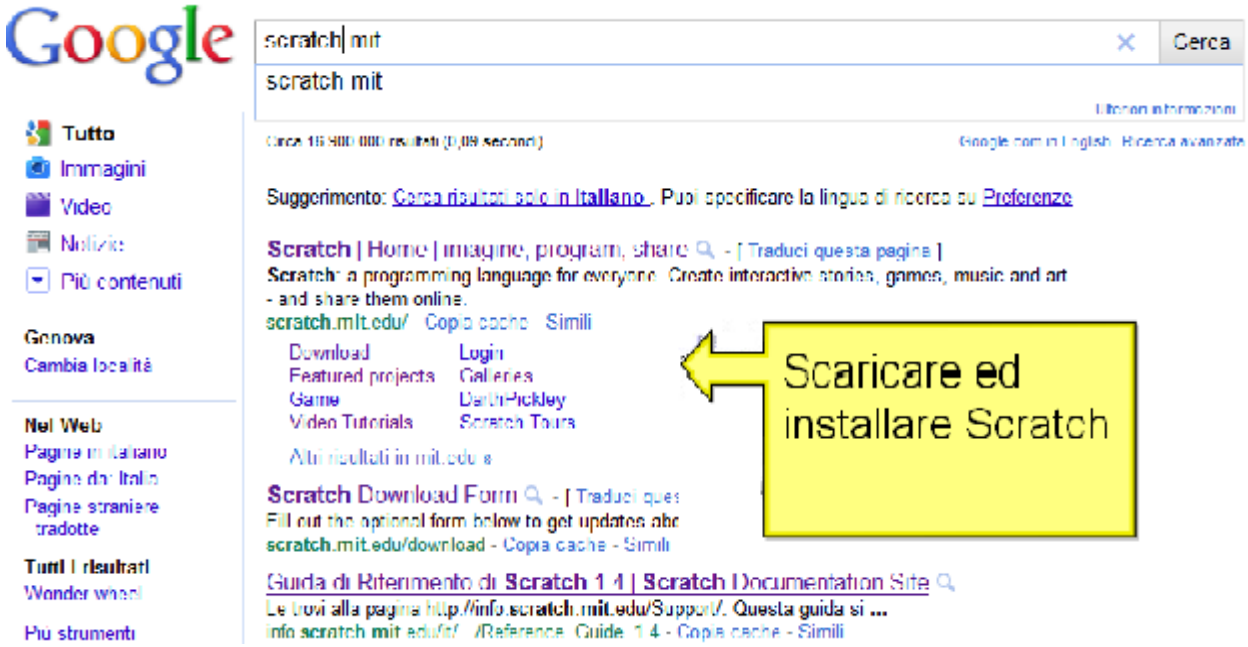
Scratch **block** o tessere: gli oggetti grafici *o mattoncini* che servono a costruire i **programmi** e si distinguono in **hat** o cappello (block che può essere inserito solo all'inizio dello script) **stack** o catasta (block che si incastra con altri: il primo ad essere incastrato nella sequenza di incastri è l'ultimo a poter essere disinserito) **reporter** o corrispondente (block di diversa forma/colore che deve essere inserito, nell'area di input, all'interno di un altro block)

1 Il ***World Wide Web*** (in sigla) ***WWW***, più spesso abbreviato in ***Web***, anche conosciuto come **Grande Ragnatela Mondiale**, è un **servizio** di Internet che permette di navigare ed usufruire di un insieme vastissimo di contenuti multimediali.

2 Un *codice* è un sistema di segni dotato di un proprio **alfabeto**, regole **sintattiche** (di scrittura) e regole **semantiche** (di interpretazione).

Scaricare ed installare Scratch

Con un motore di ricerca, ad esempio **Google**, si impostino come termini di ricerca ***scratch mit*** come esemplificato in figura e si selezioni la parola **Download**



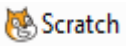
Si apre, nella fistra del browser, la **home page** del sito del MIT:

Selezionando il pulsante per il ***download***, si apre una seconda pagina; a fondo pagina, senza obbligo di compilare moduli, si prema sul pulsante

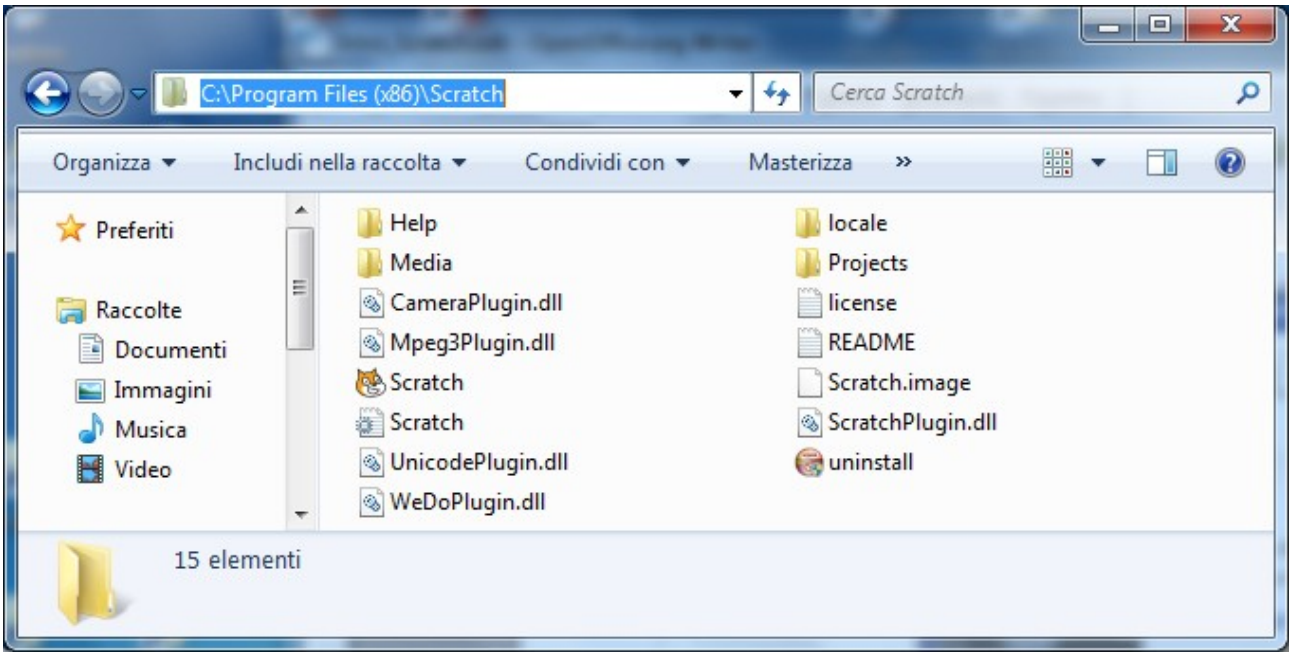
nb: l'iscrizione è essenziale solo se si vuole pubblicare nel web, condividendo i propri progetti

Scegliendo, eventualmente, la lingua ***italiano*** appare la finestra per il download dell'**installatore** dell'ultima versione, potendo selezionare il Sistema Operativo adatto.

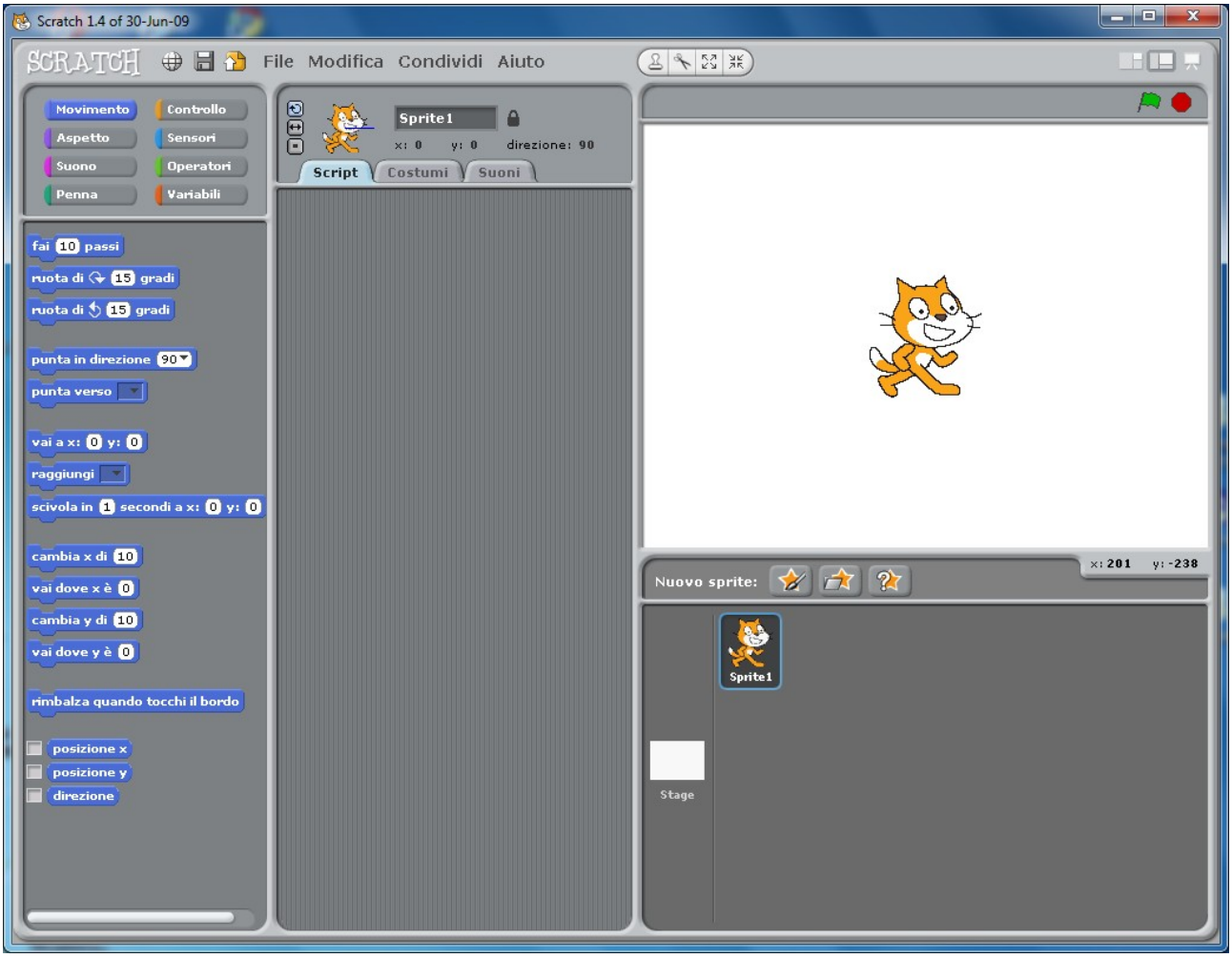
Con doppio click sul file ***eseguibile*** selezionato, si apre una finestra che richiede il **percorso** dove salvare la cartella **Scratch** che conterrà l'***ambiente di lavoro*** per la programmazione (si consiglia di accettare la scelta proposta dove sarà autoestratto).

La cartella **Scratch** contiene, in particolare, il file ***eseguibile*** 

ed altre sottocartelle utili ai nostri progetti: file di help (aiuto), progetti già realizzati ma modificabili ecc..



Doppio click sul file ***eseguibile*** Scratch visualizza l'interfaccia mostrata in figura:



Nel ***frame*** di sinistra si vedono i comandi veri e propri suddivisi per categorie:

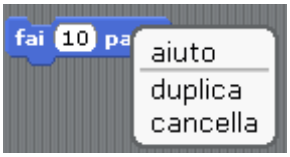
Tali comandi possono essere trascinati nell'area centrale (**Script**) dove possono essere direttamente eseguiti con un click sul comando stesso.

Tale esecuzione si osserva nel riquadro a destra: lo **stage**. Il protagonista è lo **sprite** che visualizza l'esecuzione dei comandi.

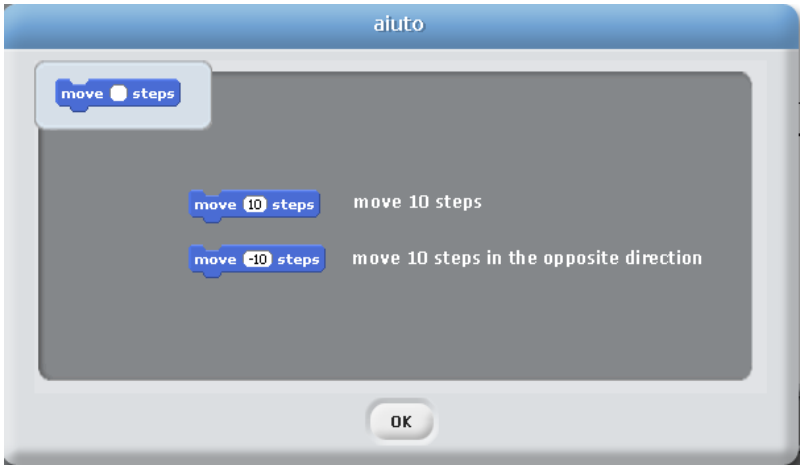
Sopra lo stage, sulla sinistra, si vedono 4 pulsanti utili:

Duplica, Cancella, Espandi sprite, Riduci sprite

Per cancellare o duplicare il comando, trascinato nell'area di Script, si può anche ricorrere al **menù contestuale** che appare alla pressione del tasto destro del mouse sul comando stesso. La prima opzione "aiuto" apre una



finestra di aiuto personalizzata a seconda del comando:



Per **salvare con nome** un progetto:

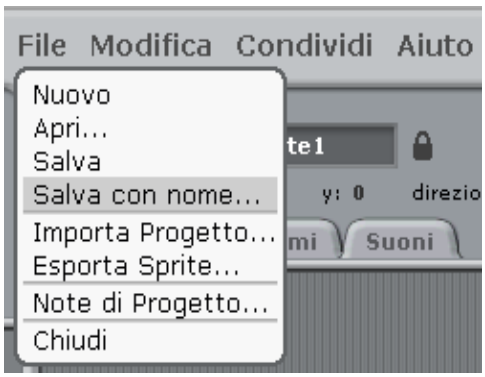
Con uso icona:

premendo il tasto destro del mouse dopo aver selezionato l'icona 

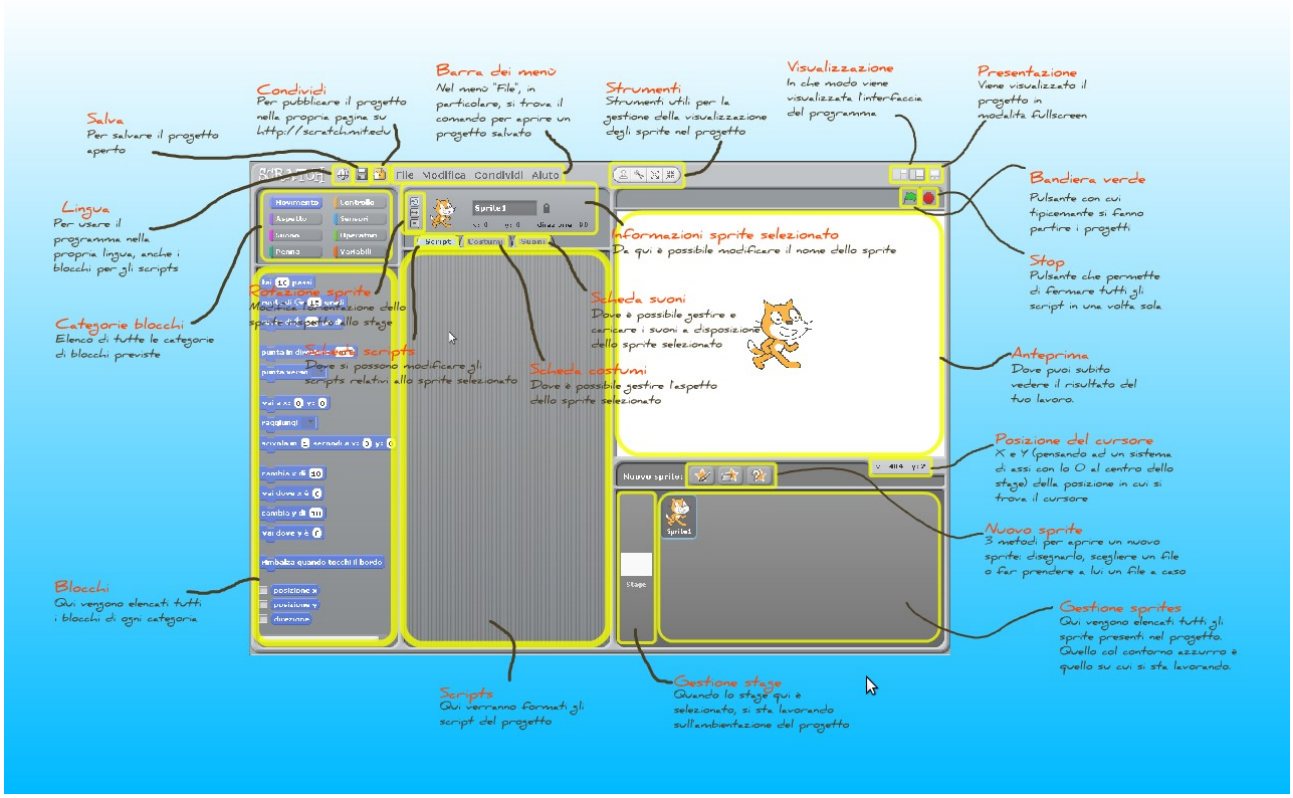
si imposta il nome del file (si consiglia di non modificare il percorso di *default*) e si preme OK.

Con uso comando da barra dei menù:

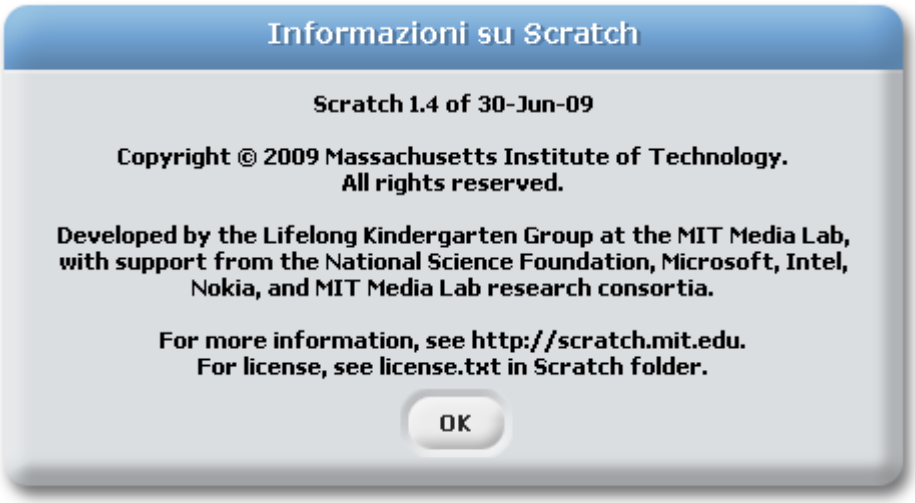
con percorso **File** → Salva con nome ...



Interfaccia di Scratch



Nella **barra dei menù**:



Pagina Aiuto sul sito … nella sottocartella Help\en**index.html**

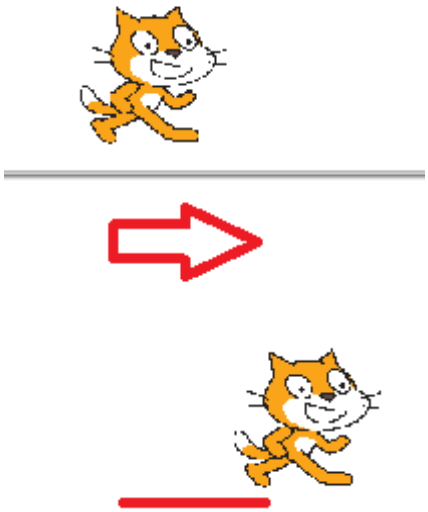
Schermate di aiuto … nella sottocartella Help\en**allscreens.html**

Esecuzione singolo comando

Esempio:

Si trascina nell'area degli **Script** il comando 

si modifica il valore nella tessera scrivendo 100



Facendo click sulla tessera, lo sprite si sposta in avanti di 100 pixel

100 pixel

Esempio con uso del pulsante **Duplica**: 

Si trascina nell'area degli **Script** il comando 

si modifica il valore nella tessera scrivendo 100
si seleziona il pulsante Duplica e si trascina il tampone sulla tessera da copiare 

al click si realizza la duplicazione, potendo spostare la nuova tessera con attenzione a non incastrarla con la precedente e si modifica il valore nella nuova tessera scrivendo -100

al click sulla prima tessera lo sprite avanza di 100 px,

al click sulla seconda tessera lo sprite torna nella posizione originaria indietreggiando di altrettanti pixel

nb: se si incastrano le due tessere, 

al click (sull'una o l'altra) non appare nessuno spostamento infatti i **comandi vengono eseguiti in sequenza** ed è **visualizzato l'effetto solo al termine dell'esecuzione**: in questo caso lo sprite resta fermo.

Esecuzione passo-passo

Si incastrano uno dopo l'atro i comandi desiderati costruendo un *blocco (pila di tessere)* con attenzione alla **SEQUENZA** delle azioni che verranno attuate dallo sprite

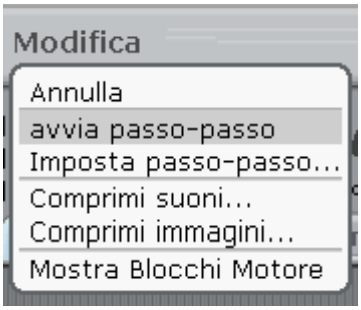
Ad esempio si imposta la sequente **SEQUENZA** :

dalla categoria 

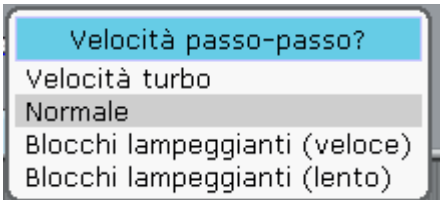
si personalizza la tessera 

scegliendo un'attesa di 10 secondi

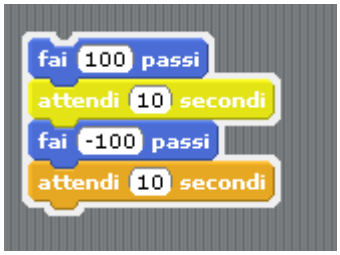
Si predispone l'esecuzione **passo-passo**



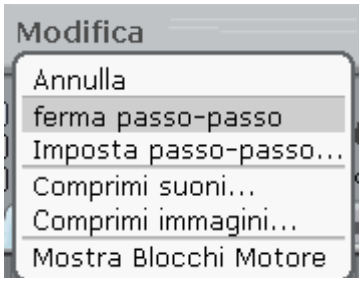
potendo impostare ad esempio la velocità Normale



Al click sul *blocco* l'esecuzione del singolo comando è evidenziata da colore di sfondo giallo della tessera relativa



Si può fermare l'esecuzione **passo-passo**



al termine del test di esecuzione infatti tale metodo serve per verificare il flusso di programmazione in fase di ***debugging***[3] del programma.

3 Con il termine **debuggare**, in informatica, si intende cercare e correggere gli erori. L'uso del termine *bug*, che in inglese indica genericamente un piccolo insetto, è legato ad un curioso aneddoto

Programmazione: SEQUENZA con esecuzione dall'inizio alla fine

Si incastrano uno dopo l'atro i comandi desiderati costruendo un *blocco (pila di tessere)* con attenzione alla **SEQUENZA** delle azioni che verranno attuate dallo sprite

Ad esempio si imposta la sequente **SEQUENZA** :

dalla categoria 

si personalizza la tessera 

scegliendo cosa far dire allo sprite e per quanto tempo.

Effetto del **programma**:

non si muove per 2 secondi

si muove in avanti di 200 passi

è tornato alla posizione di partenza senza dire nulla

Incastrare una tessera sull'altra corrisponde ad inserire istruzioni all'interno di una **funzione** che, al lancio dell'applicazione (il click sul *blocco*), viene **eseguita dall'inizio alla fine** dall'unità centrale (CPU).

In realtà il click sul blocco determina una **interpretazione** dei comandi: una traduzione simultanea[4] al momento dell'esecuzione in linguaggio comprensibile dal computer.

---

4 I progetti di Scratch sono **applet Java**, perciò possono essere inseriti in pagine html e sono **interpretati** da qualsiasi browser. Si può vedere un piccolo esempio online all'indirizzo *http://nilocram.free.fr/spip/spip.php?article12 con associazione nomi animali italiano-inglese*

Programmazione *event-driven*: esecuzione quando si clicca sulla bandiera verde

Si incastrano uno dopo l'atro i comandi desiderati costruendo un *blocco (pila di tessere)* con attenzione alla **SEQUENZA** delle azioni che verranno attuate dallo sprite anche **quando** si verifica l'**evento** associato al click sulla bandiera verde.

La stessa **SEQUENZA** già esemplificata :

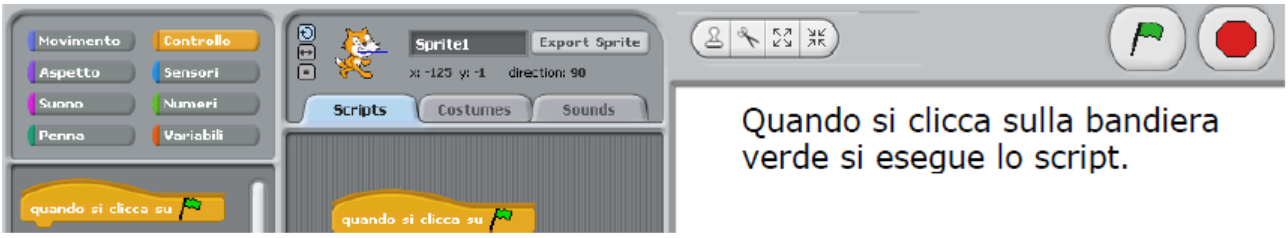
Dalla categoria **CONTROLLO**
trascina 

e incastralo sopra al blocco

Dalla categoria **CONTROLLO**
trascina 

e incastralo sotto al blocco

Ottenendo il seguente programma o **script**



Programmazione: continua a ripetere (loop infinito)

Si incastrano uno dopo l'atro i comandi desiderati costruendo il *blocco (pila di tessere)* con attenzione alla **SEQUENZA** delle azioni che si vuole che lo sprite **RIPETA PER SEMPRE**.

Ad esempio si imposta la sequente **SEQUENZA** :

ricorrendo alla categoria 

per inserire la tessera 

Dalla categoria **CONTROLLO** trascina **PER SEMPRE**.

Trascina lo stack di tessere all'interno di **PER SEMPRE**

Per trascinare uno stack selezionalo dalla tessera più in alto

Per **eseguire** il programma: un click sul blocco.

Al click sul blocco la **sequenza dei comandi viene eseguita per sempre** ed è **visualizzato**, come **effetto**, lo sprite che parla e, con ritardo di 2 secondi, avanza e rimbalza quando tocca il bordo (capovolgendosi quando tocca il bordo destro)

Per fermarlo, clicca sul pulsante di **STOP**, in alto a destra nella schermata.

Premere 

se si vuole **evitare l'inversione** dello sprite quando tocca il bordo.

Programmazione *event-driven*: continua a ripetere quando si clicca sulla bandiera verde

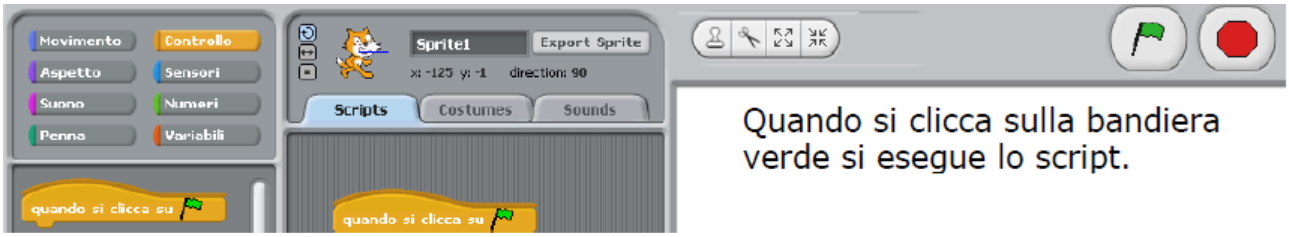
Si incastrano uno dopo l'atro i comandi desiderati costruendo un *blocco (pila di tessere)* con **RIPETIZIONE CONTINUA** delle azioni che verranno attuate dallo sprite anche **quando** si clicca sulla bandiera verde.

Lo stesso blocco "per sempre" già esemplificato :

Dalla categoria **CONTROLLO** trascina

e incastralo sopra al blocco

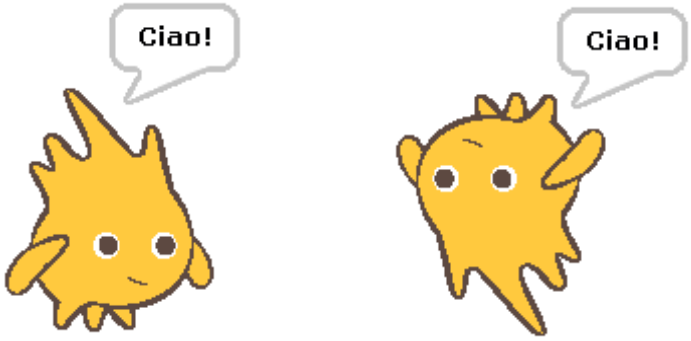
Ottenendo il seguente programma o **script**



Quando si vuole interrompere l'esecuzione si clicca su

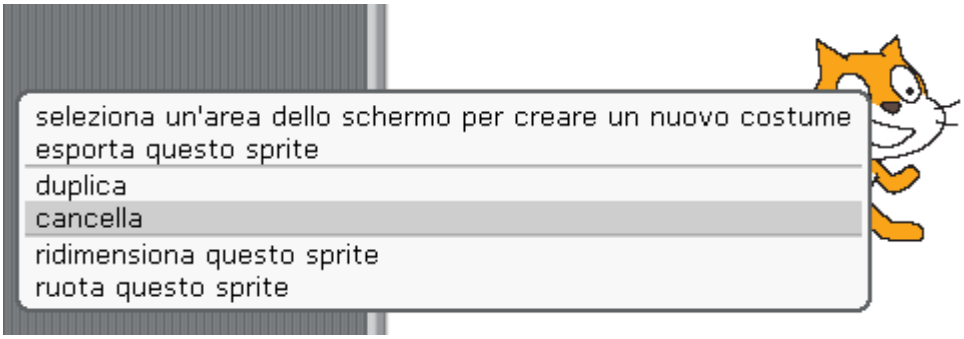
Si provi a **cambiare lo sprite**

e si verifichi l'effetto del programma:



Per **cambiare lo sprite**:

- si cancella lo sprite di default:



- si seleziona uno nuovo sprite da file:

- si seleziona la sottocartella **Fantasy** e si preme OK;

Si sceglie, selezionando 

e si preme 

Aggiungere animazioni

Passando da un COSTUME all'altro si può animare lo sprite.

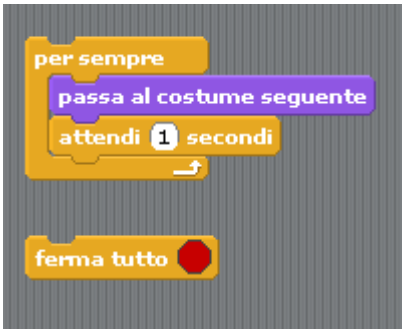
Per aggiungere un **costume** si clicca sulla linguetta **Costumi** poi si preme il pulsante **Importa** per ogni nuovo costume (ad esempio le immagini gobo2 e gobo3 in Fantasy)

Ora si clicca sulla linguetta **Script** e si crea uno script che passa da un costume all'altro:

Si controlli l'effetto dell'animazione realizzata con un click sul *blocco*.

Per fermare l'esecuzione continua, si può trascinare la tessera 

e cliccarla in alternativa alla pressione del pulsante di stop



Programmazione *event-driven*: aggiungere animazioni quando si preme il tasto spazio

Si incastrano uno dopo l'atro i comandi desiderati costruendo un *blocco (pila di tessere)* con **RIPETIZIONE CONTINUA** delle azioni che verranno attuate dallo sprite anche **quando** si preme la barra spaziatrice sulla tastiera

Lo stesso script che passa da un costume all'altro:

Dalla categoria **CONTROLLO**
trascina 

e incastralo sopra al blocco

Ottenendo il seguente programma o **script**

Quando si preme la barra spaziatrice sulla tastiera, inizia l'esecuzione dello script d'animazione.

Si sostituisce al primo costume  il secondo 

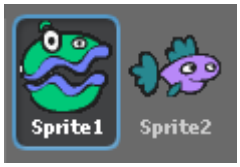
L'effetto risultante è una *bocca che si apre e chiude* con movimento che **appare continuo** se il ritardo è sufficiente breve da non essere percepito dall'uomo.

Per fermarlo, clicca sul pulsante di **STOP**, in alto a destra nella schermata. 

Programmazione multitasking ed *event-driven*: esecuzione contemporanea di più script quando si preme il tasto spazio

Si realizzano due **script** in **loop** infinito: ognuno agisce su uno sprite (un diverso disegno di pesce) cambiandone ogni 2 secondi il colore (con parametro di variazione pari a 50). Si vuole anche **modificare lo sfondo** scegliendo tra quelli disponibili il fondo del mare (underwater)

Si scelgano due sprite a cui associare i due script e si selezioni il primo:



Si incastrano uno dopo l'atro i comandi desiderati costruendo un *blocco (pila di tessere)* con **RIPETIZIONE CONTINUA** delle azioni che verranno attuate da entrambi gli sprite **quando** si preme la barra spaziatrice sulla tastiera

Dalla categoria Controllo si trascina nell'area di Scripts **per sempre**

Si crea la **SEQUENZA** seguente:

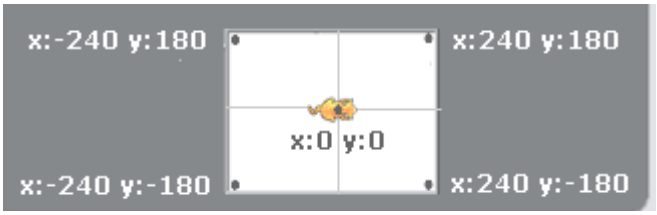
trascinandola dentro il **loop**

Si seleziona il secondo sprite compiendo le stesse operazioni

Prima del ciclo, si imposta la posizione iniziale scegliendo per entrambi gli sprite lo stesso valore dell'ascissa ma diversi valori dell'ordinata

Per il primo sprite

Per il secondo sprite



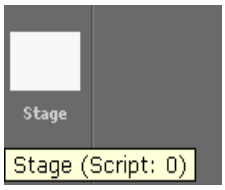
Per entrambi si inizializza il colore con nessuna variazione ad ogni **nuovo lancio (**Scratch **manterebbe i valori precedenti**).

Dalla categoria Controllo si trascina nell'area di Scripts il block di tipo **hat** inserendolo appunto come *cappello* all'inizio

script per il primo sprite

script per il secondo sprite

Modificare lo sfondo



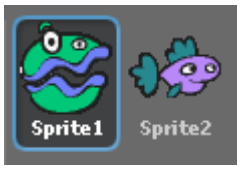
Per modificare lo sfondo, con un click sull'icona associata alla personalizzazione dello stage (ambientazione del progetto), posizionata sotto l'area dello **stage** stesso.

Come effetto si caratterizza la zona centrale con informazioni sullo stage e non sugli Scpript

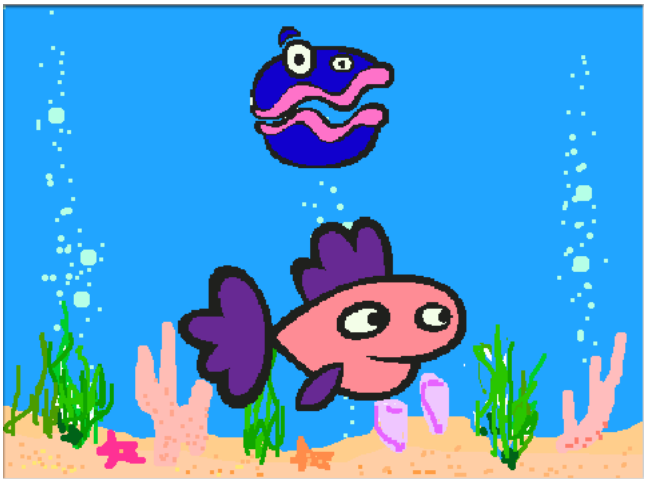
Selezionando l'etichetta **Sfondi** ed il pulsante **Importa** si possono aggiungere nell'elenco altri sfondi tratti dai molti già disponibili e/o personalizzabili.

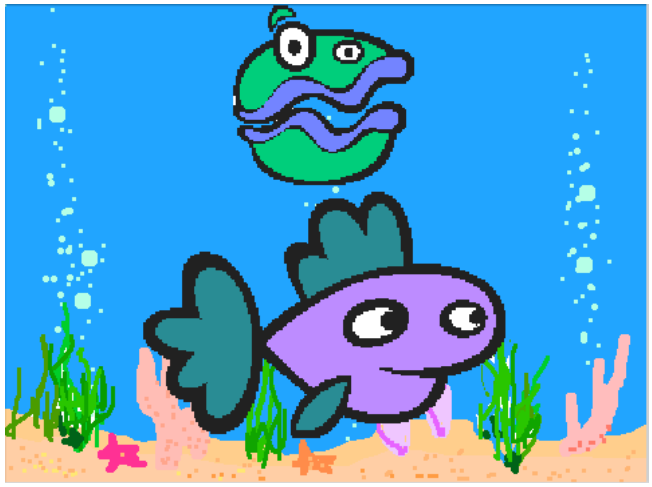
Si seleziona con un click quello desiderato tra quelli che appaiono nell'elenco

Per tornare a caratterizzare la zona centrale con la visualizzazione degli Script basta selezionare uno degli sprite



L'effetto dunque, **quando** si preme la barra spaziatrice sulla tastiera mostra quattro variazioni di colore degli sprite che poi si riavvicendano sempre uguali:



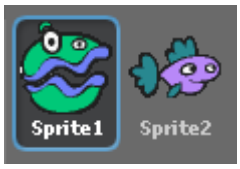


In figura: la prima variazione di colore e il ritorno a quella iniziale

Programmazione multitasking ed *event-driven*: esecuzione contemporanea di più script quando si clicca sulla bandiera verde

Si realizzano due **script** in **loop** infinito: ognuno agisce su uno sprite cambiandone ogni 2 secondi il colore (con parametro di variazione pari a 50) e la dimensione (aumento del 30%).

Si scelgano due sprite a cui associare i due script e si selezioni il primo:



Dalla categoria Controllo si trascina nell'area di Scripts **per sempre**

Si crea la **SEQUENZA** seguente:

trascinandola dentro il **loop**

Si seleziona il secondo sprite compiendo le stesse operazioni

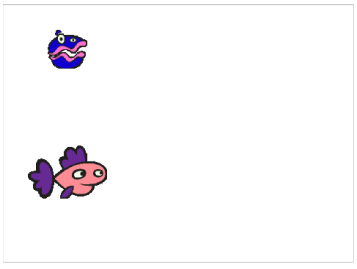
Dalla categoria **CONTROLLO** trascina

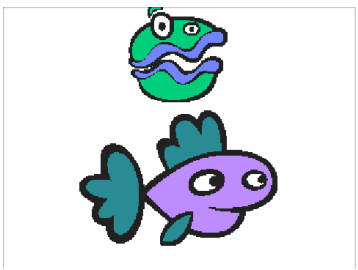
all'inizio di entrambi gli **script** per lanciarne **contemporaneamente** l'esecuzione

Se si interrompe e si rilancia l'esecuzione dello script in figura, si verifica che Scratch **non reinizializza automaticamente** il colore e le dimensioni **al nuovo lancio mantenendo i valori precedenti**.

Una prima soluzione è **inizializzare i valori** come prime istruzioni: ad esempio impostare il valore dell'**ascissa a -200**, reimpostare il colore originario (con nessuna variazione) e le dimensioni ridotte del 30%.

Si potrebbe poi sostituire il loop continuo con un **ciclo che ripete solo un numero finito di volte**: quindi si trascina nell'area centrale il controllo per implementare il costrutto che prevede tale ripetizione per sole 4 volte e si fanno avanzare gli sprite di 50 passi ad ogni iterazione. L'effetto iniziale e finale è mostrato in figura.





In tal caso non si raggiungono le dimensioni limite che non permettevano successive modifiche

Programmazione multitasking ed *event-driven*: esecuzione contemporanea di più script con animazione a effetto mosaico quando si clicca sulla bandiera verde

Si realizzano due **script** in **loop** infinito: ognuno agisce su uno sprite creato ed **esportato** con nome robot nella sottocartella Fantasy dell'ambiente.

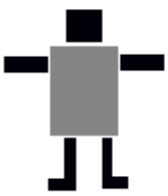
Si imposta diversa la posizione iniziale (uguale valore dell'ordinata) e si sceglie di far inizialmente indietreggiare il primo robot mentre il secondo avanza (dello stesso numero di passi) e viceversa ogni volta che si moltiplicano con **effetto mosaico**.

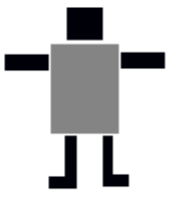
Per vederli avanzare/ indietreggiare si inseriscono ritardi di 1 secondo.

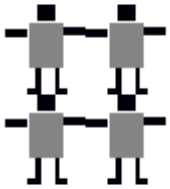
Script associato al primo **robot**

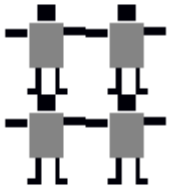
Script associato al secondo **robot**

Al click sulla bandiera verde si allontanano uno dall'altro, poi si avvicinano moltiplicati e così via:







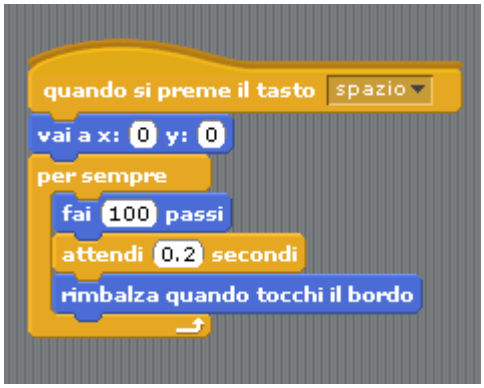


Programmazione multitasking ed *event-driven*: esecuzione contemporanea di più script associati allo sfondo e allo sprite per diverse occorrenze di evento

Si vuole che alla ***pressione dello spazio*** vengano eseguiti **due script**:

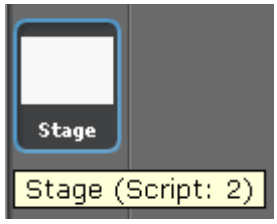
- uno associato allo **sprite** che in loop continuo si vuole avanzi, rimbalzi quando tocca il bordo e torni indietro senza capovolgersi

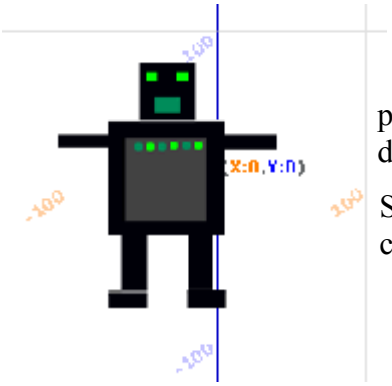
avendo premuto il pulsante 



- l'altro associato allo **sfondo** che si vuole bianco (sfondo1)

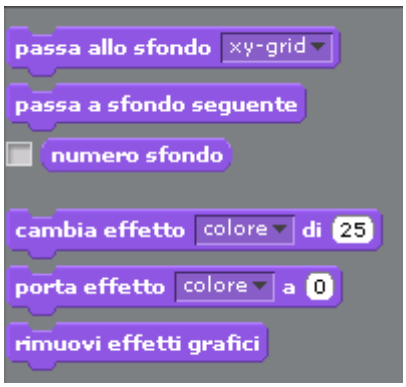
Si vuole poi che alla ***pressione sullo sfondo (stage)*** venga eseguito un secondo **script** anch'esso associato allo **sfondo** che si vuole cambi inserendo la griglia x-y





potendo controllare il numero di pixel dello spostamento

Si noti che selezionando l'icona stage, ed il tipo di comando compaiono sulla sinistra nuove tessere per gestire lo sfondo



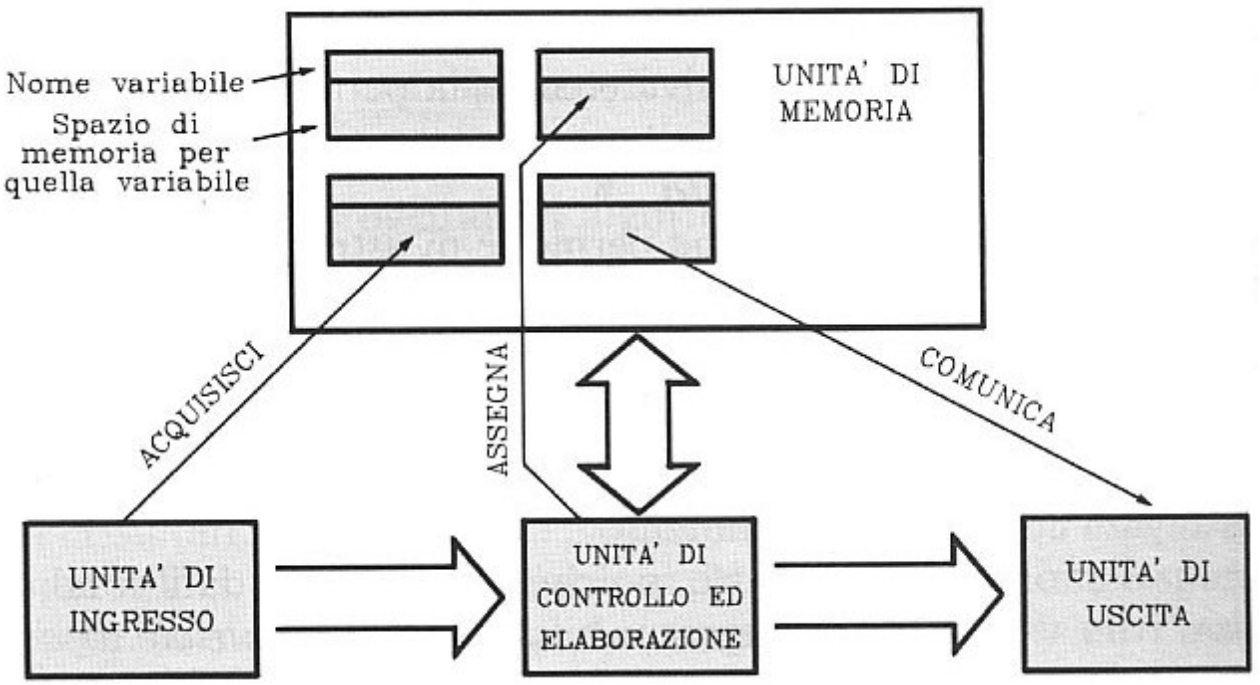
Si vuole infine che alla ***pressione sullo sprite*** simulando la situazione del "colpire il bersaglio" venga eseguito un secondo **script** associato appunto allo **sprite** che lo riporti in posizione centrale e fermi tutti gli scrpit:

Avendo scelto un intervallo di 0.2 secondi per spostare lo sprite di 100 pixel può rimanere difficile "colpire il bersaglio" anche con la guida della griglia: per renderlo più facile si può impostare un tempo di attesa più lungo.

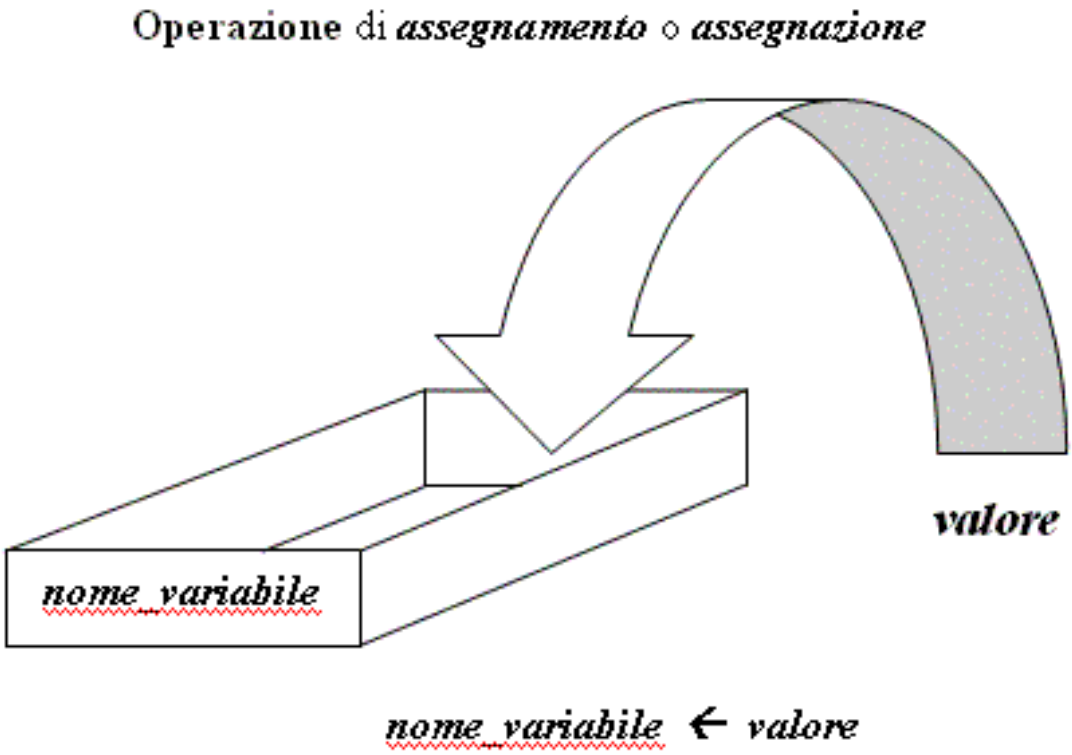
In seguito una diversa soluzione per migliorare il gioco.

Variabili

Una **variabile** è un'area di memoria (RAM) individuata da un **nome** detto identificativo. Tale nome esprime l'*indirizzo* dove sarà posto il valore che si **assegnerà** alla variabile.



Una variabile si può pensare come una scatola aperta dove si possono introdurre dati.



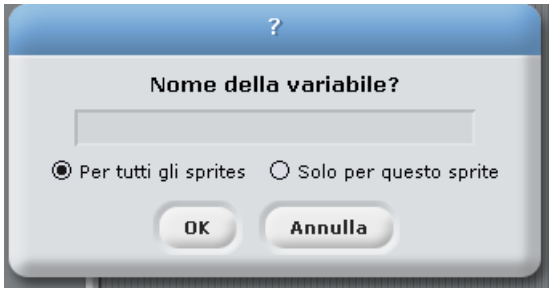
Variabili in Scratch

Nel ***frame*** di sinistra, nella categoria 

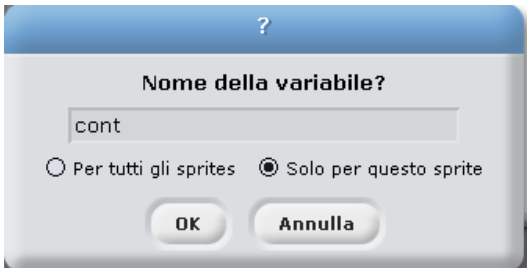
si selezioni 

e si scelga il nome

Si potrà definire la variabile globale o locale cioè visibile da tutti gli sprites o solo dallo sprite attuale:

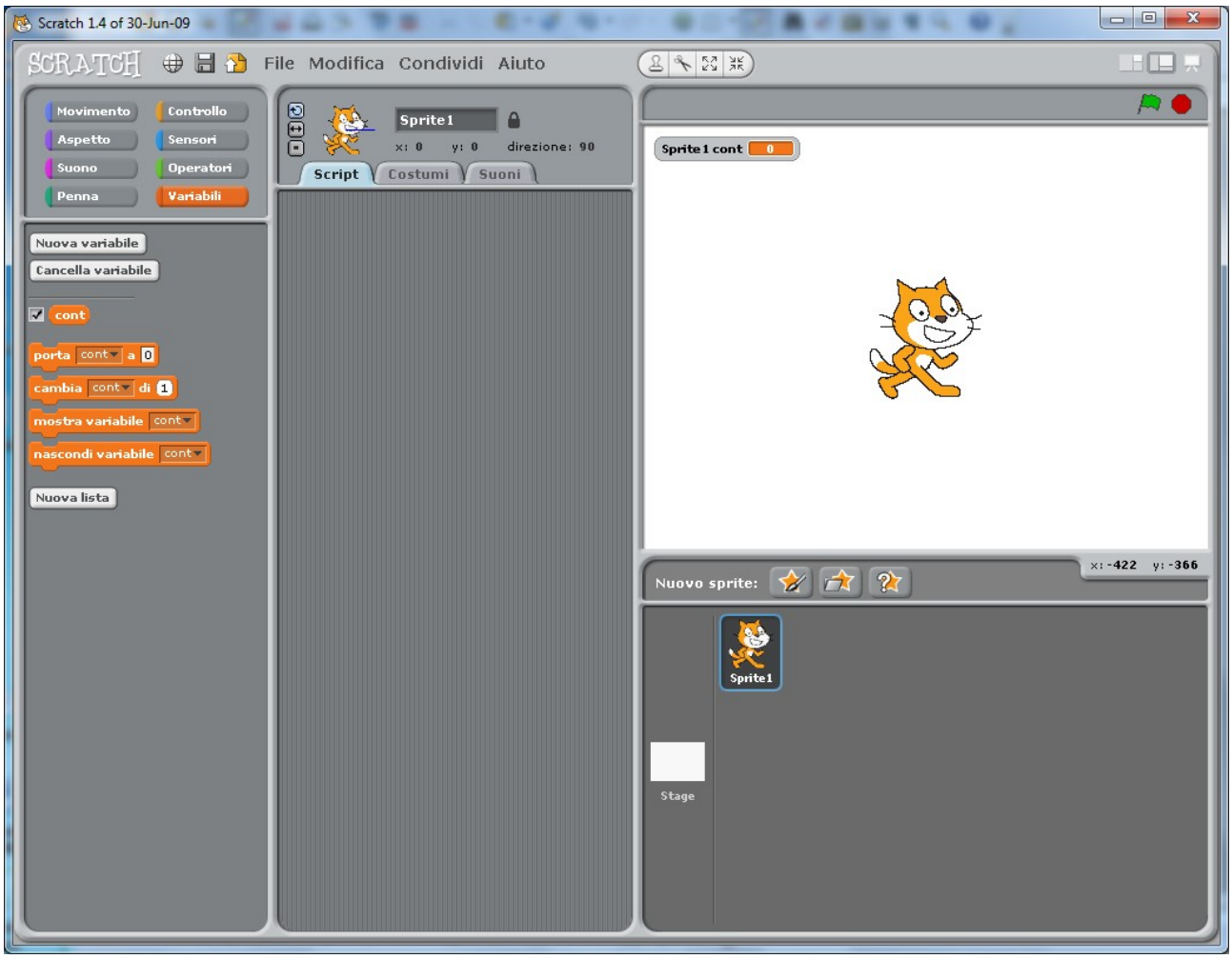


Si consiglia la scelta di variabili **locali**.

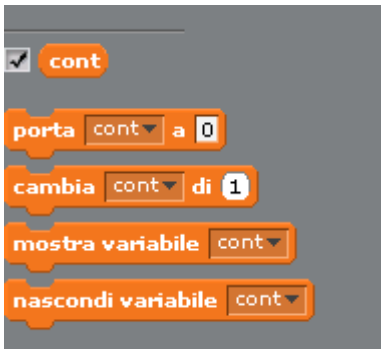


Scelto il nome - ad esempio **cont -** e premuto OK

l'interfaccia si presenta come in figura:



dove, nel ***frame*** di sinistra è comparso, smarcato, il nome della variabile



e tessere per modificarne il valore

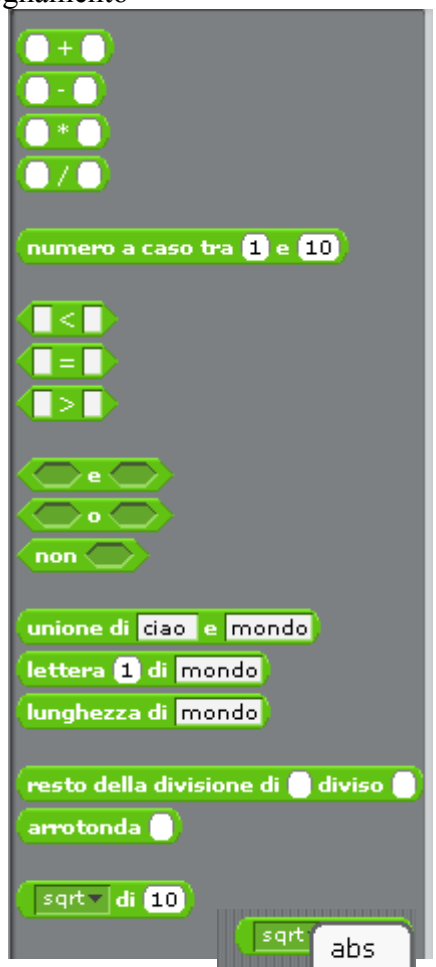
o la visualizzazione

Di default è mostrata nello stage: 

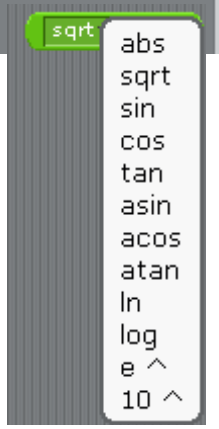
Operatori in Scratch

Una volta definita una variabile, si può usare per impostare formule generali in espressioni che coinvolgono diversi 

oltre alla fondale operazione di assegnamento



- operatori aritmetici
- funzione di generazione di numeri *casuali*
- operatori relazionali
- operatori logici
- funzioni per concatenare, estrarre singoli caratteri da una frase, calcolare la lunghezza (numero di caratteri) di una frase
- operatore resto
- operazioni di arrotondamento
- funzioni matematiche e trigonometriche



Esempio di **operatore relazionale**: 

impostato trascinando nell'area cerntrale Script l'operatore relazionale minore, trascinando all'interno del primo operando la variabile 

spostandosi tra i comandi della categoria 

e 

e digitando come secondo operando il valore 4

Esempio di **concatenazione** di frasi: 

potendo concatenare anche parole e numeri

Esempio di **estrazione pseudocasuale** di un numero tra 1 e 2.8:

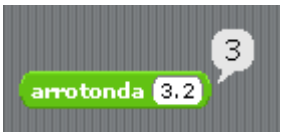
Esempio di **estrazione del terzo carattere** dalla parola mondo: 

Esempio di **calcolo della lunghezza** (numero di caratteri) della parola mondo: 

Esempio di operatore **resto** tra numeri **non divisibili** 

tra numeri **divisibili** 

Esempio di **arrotondamento** per eccesso 

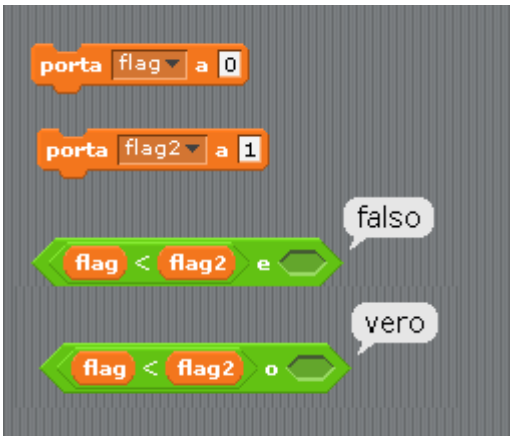
oppure per difetto 

Nell'uso di **operatori logici** l'assenza di operando è per default interpretata come **falso**

infatti la **negazione** di falso è vero 

ed impostando le due variabili flag e flag2 in modo che sia vera la condizione 

sia vera



l'operazione **AND** fornisce risultato falso
(vero solo se entrambi sono veri)

l'operazione **OR** fornisce risultato vero
(è sufficiente che uno sia vero)

Programmazione: costrutto ripetizione con controllo in testa sul numero di iterazioni

Si incastrino uno dopo l'atro i comandi desiderati costruendo un *blocco (pila di tessere)* che prevede: l'inizializzazione di una variabile contatore di nome **conta** (con valore pari a zero) ed una **ripetizione per un numero di volte prefissato** dell'incremento del contatore con attesa di qualche secondo (ad esempio 3) facendo dire allo sprite il valore attuale del contatore.

Nel ***frame*** di sinistra, nella categoria 

si selezioni 

e si scelga il nome **conta** definendo una variabile locale cioè visibile dallo sprite attuale.

Nel ***frame*** di sinistra comparirà la variabile creata e nello **stage** sarà visualizzato il valore di *default* cioè zero:

Come prima istruzione, si inizializza il contatore 

perchè Scratch non azzera automaticamente le variabili al nuovo lancio, mantenendo i valori precedenti.

Quindi si trascina nell'area centrale Script il controllo per implementare il costrutto che prevede la ripetizione per 10 volte:

Il *blocco* che sarà ripetuto si compone ponendo in **SEQUENZA:**

l'incremento di 1 della variabile conta
il comunicare il valore di **conta**
il controllo di attesa 3 secondi

personalizzando la tessera della categoria aspetto trascinando la variabile al posto di Ciao!

Al lancio dello **script**

lo sprite comincia a contare

e l'esecuzione termina quando arriva a 10:

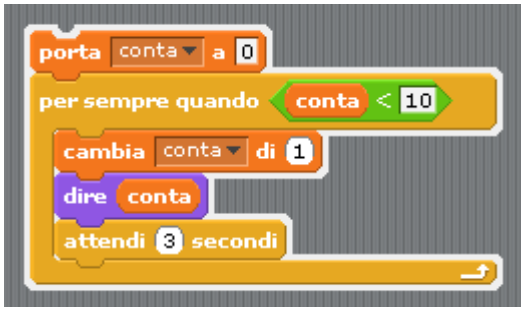
Programmazione: costrutto ripetizione con controllo in testa

Si può risolvere l'esercizio precedente sostituendo al controllo che itera per un numero di volte prefissato, quello che implementa il costrutto **ripetizione mentre è vera una determinata condizione**

impostando come condizione

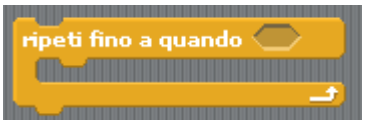
Si incastrino, allora, uno dopo l'atro i comandi desiderati costruendo un *blocco (pila di tessere)* che prevede: l'inizializzazione di una variabile contatore di nome **conta** (con valore pari a zero) ed una **ripetizione mentre è vera una determinata condizione** dell'incremento del contatore con attesa di qualche secondo (ad esempio 3) facendo dire allo sprite il valore attuale del contatore.

Al lancio dello **script**



si ottengono gli stessi risultati.

Si può risolvere l'esercizio precedente sostituendo al controllo che itera per un numero di volte prefissato, quello che implementa il costrutto **ripetizione fino a quando non si realizza una determinata condizione**



impostando come condizione

oppure

Al lancio dello **script**

Come al lancio dello **script**

si ottengono gli stessi risultati.

Programmazione: costrutto alternativa (*binaria*)

Ricercare il **minimo** confrontando il valore di due variabili di nome **a** e **b** inizializzate con **valore casuale** tra 1 e 10. Si comunichi tale valor minimo considerando indifferente il caso di uguaglianza.

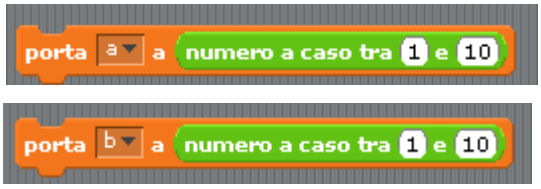
Nel ***frame*** di sinistra, nella categoria 

si selezioni 

e si scelga prima il nome **a** poi **b** definendo due variabili locali cioè visibili dallo sprite attuale.

Nel ***frame*** di sinistra compariranno tali variabili e nello **stage** saranno visualizzati i loro valori di *default* cioè zero:

Si inizializzino le variabili con numero casuale:

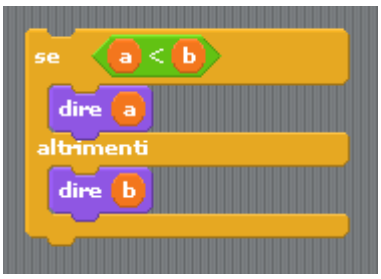


Si trascini nell'area centrale Script il controllo 

per implementare il costrutto alternativa che prevede:

- una condizione da testare
- un'azione da svolgere se la condizione risulta vera
- un'azione da svolgere se la condizione risulta falsa

Come condizione si imposti e si trascini l'operazione relazionale 



come azioni da svolgere **dire** il valore minore personalizzando la tessera della categoria aspetto trascinando le variabili al posto di Ciao!

Al lancio dello **script**

Nello **stage** si visualizza, casualmente:

dove **b** che contiene 3 è minore di **a** che contiene 6

…. provare con lanci successivi

Programmazione: costrutto ripetizione con *controllo in coda*

Dei tre costrutti per realizzare un algoritmo strutturato (***sequenza***, ***alternativa*** o selezione ed ***iterazione*** o ripetizione) quello **iterativo** può assumere due forme: con controllo ***in testa*** o ***in coda*** a seconda che la **condizione** che determina il proseguire o terminare la ripetizione sia **testata prima** di eseguire le azioni da iterare oppure **dopo** averle eseguite almeno una volta. Il ciclo o loop infinito è un caso particolare di iterazione con controllo in testa.

In Scratch (fino alla versione 1.4) non è previsto uno ***Scratch block*** che implementi l'***iterazione con controllo in coda*** ma si può realizzare tale soluzione algoritmica con uno ***Scratch block*** di tipo ripeti ***"per sempre"*** e prevedere l'uscita dal ciclo implementando un'alternativa.

```
inizio
    leggi N
    S ←— ∅
    fai
        leggi A
        S ←  S+A
        N←—N-1
    mentre ( N > ∅ )
    stampa S
fine
```

Ad esempio, volendo implementare il calcolo della **somma** S di una sequenza di N addendi di valore variabile A letto da tastiera, l'algoritmo in pseudocodice è quello mostrato nella figura a fianco, dove si vuole che almeno un addendo venga inserito.

L'algoritmo può essere ripensato come loop infinito e condizione d'uscita dopo aver eseguito almeno una volta (pseudocodice ed affiancato lo script in Scratch) :

```
inizio
    leggi N
    S ← ∅
    mentre ( sempre )
        leggi A
        S ← S + A
        N ← N-1
        se (N = 0)
          scrivi S
          termina
fine
```

Programmazione: costrutto ripetizione con controllo in coda realizzato come loop infinito e condizione d'uscita dopo aver eseguito almeno una volta

Un miglioramento dell'esempio precedentemente proposto, simulando la situazione del "colpire il bersaglio", è inserire una variabile **contatore** per contare quante volte si "colpisce il bersaglio" cioè si riesce a clickare correttamente sullo sprite e comunicare Bravo all'utente se supera una certa soglia di difficoltà fissata (ad esempio 10), sostituendo i due script associati allo sprite con i seguenti:

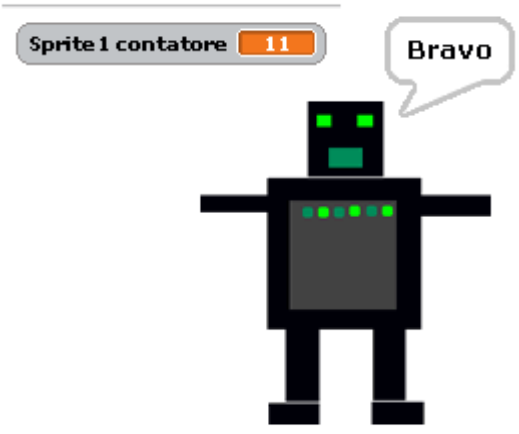
- alla ***pressione dello spazio***

```
{
    posiziona in centro
    contatore ← 0
    mentre (sempre)
        avanza
        attesa
        se (contatore > 10)
            dire Bravo
             ferma tutto
         rimbalza quando tocchi bordo
}
```

- alla ***pressione sullo sprite*** si incrementa la variabile contatore

contatore ← contatore + 1

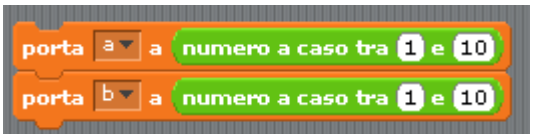
Effetto delle modifiche (nel caso non si sia mai clickato lo sfondo) :



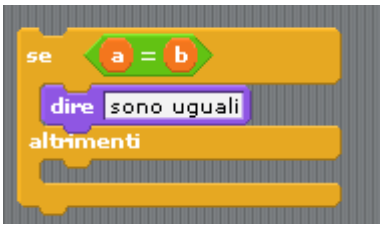
Programmazione: alternativa ternaria

Ricercare il **minimo** confrontando il valore di due variabili di nome **a** e **b** inizializzate con **valore casuale** tra 1 e 10. Si comunichi tale valor minimo concatenandolo alla frase "il minore vale" oppure, nel caso di uguaglianza, si comunichi che "sono uguali"

Come per l'esercizio precedente, si dichiarano e inizializzano le due variabili con numero casuale:



Come per l'esercizio precedente, si trascina nell'area centrale Script il controllo per implementare il costrutto alternativa:



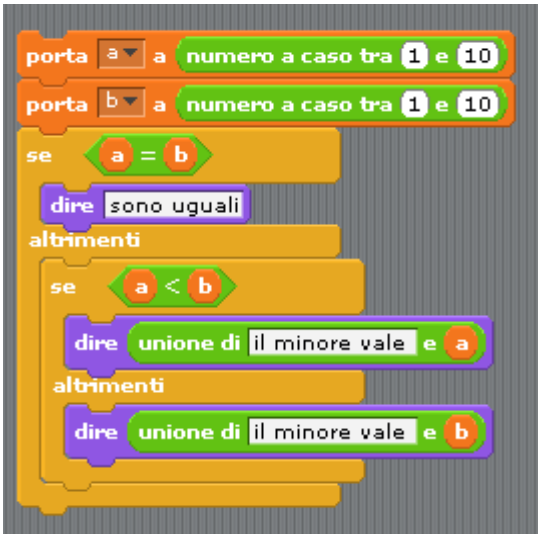
come **condizione** di test è l'**uguaglianza**

come azione da svolgere se la condizione risulta vera la comunicazione che "sono uguali"

come *blocco* da svolgere se la condizione risulta falsa un secondo costrutto alternativa

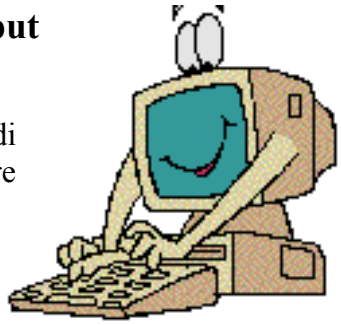
dove si **concatena** la frase desiderata al valore della variabile

Al lancio dello **script**



Nello **stage** si visualizza, con lanci successivi, sia la situazione di uguaglianza sia il caso di valori diversi:

Programmazione: operazioni di input e output



È possibile far interagire lo sprite con l'utente; abbiamo già visto esempi di comunicazione (**operazioni di uscita**) da parte dello sprite con uso di tessere **dire** potendo concatenare frasi e risultati.

Per poter consentire di chiedere informazioni all'utente ed usarle per fornire risposte, si considerino i controlli di tipo 

Esiste la tessera 

personalizzabile nella richiesta all'utente

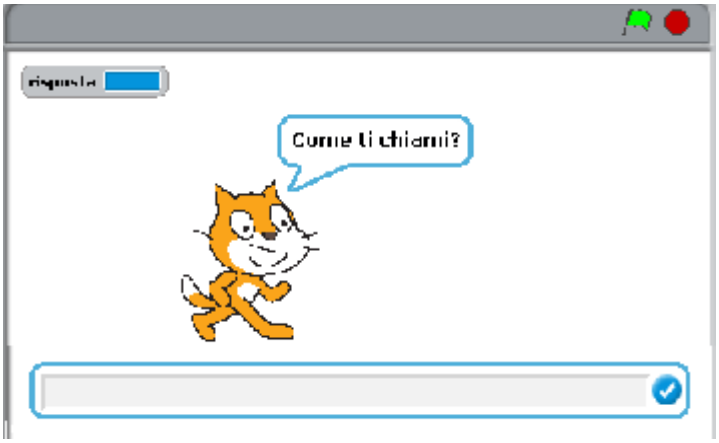
e la tessera 

che può essere smarcata 

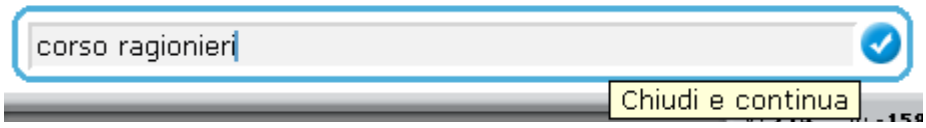
volendone anche direttamente nello stage la visualizzazione 

ed è block di tipo **reporter** cioè può essere inserito in altro come nello **script** esemplificato:

Al click sulla bandiera verde, l'effetto è quello visualizzato:



inserito nella casella di testo un nome o frase il **click sul pulsante di spunta** interrompe l'attesa programmata, e concatenandola ad un saluto

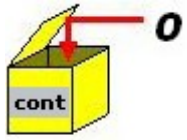


la risposta dell'utente viene sia visualizzata nello stage sia usata per personalizare il saluto – la comunicazione dello sprite.

Progetto

Si disegni[5] un nuovo sprite: una persona stilizzata che inizialmente sia in posizione normale (punti in direzione 90^0) come in figura, poi **continuamente** ruoti via via di 45^0 fino a capovolgersi e ritornare in posizione eretta e a seconda della posizione in cui si trova, ogni 2 secondi, **pensi**.

Si faccia uso di una **variabile** di nome **cont** inizialmente uguale a zero



All'inizio della capovolta penserà **ora** e

quando ruota in avanti penserà **giù**

quando ruota all'indietro penserà **sù**

Esiste icona opportuna per disegnare un nuovo sprite:

che può essere solo associato allo script o memorizzato con **Esporta Sprite ...** in una sottocartella dell'ambiente

(da barra dei Menù: File → Esporta Sprite ...)

ad esempio in sottocartella People con nome *uomo*

Possibile **Script**:

Con click sull'icona  si

5 Software per disegnare realizzato come progetto Scratch http://scratch.mit.edu/projects/soccerfirst/1536853

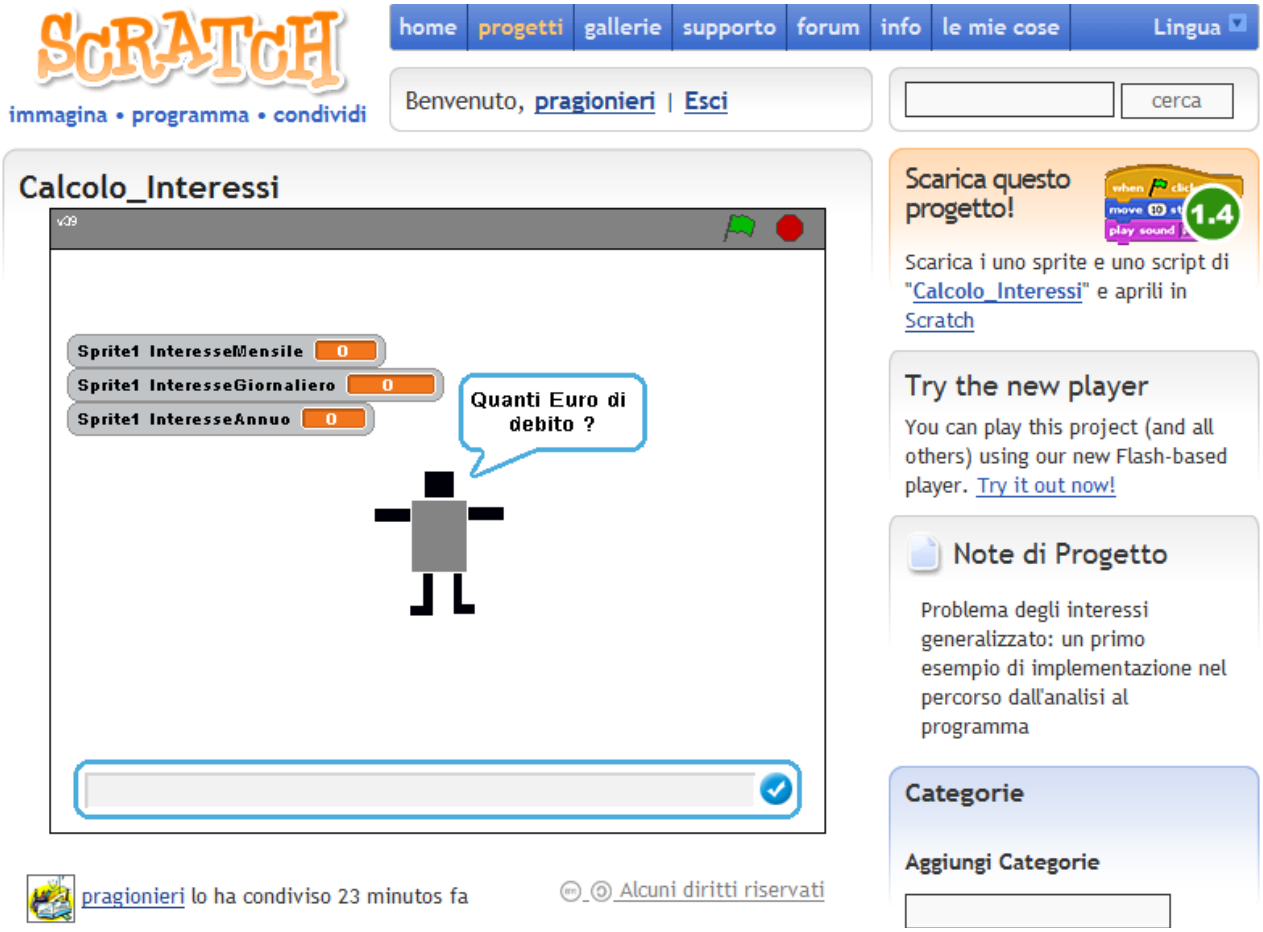
Progetto condiviso in rete

Problema degli interessi generalizzato:
Se all'inizio dell'anno avete un certo Debito con la banca e il tasso d'interesse percentuale annuo praticato dalla banca è TassoAnnuo, quanto spendete di interessi ogni giorno ? E ogni mese ?

online dall'analisi all'implementazione

Si può entrare nel sito http://scratch.mit.edu/, accedere con il prorio account e trovare il progetto nella categoria "I Miei Progetti" potendolo scaricare in locale o eseguire:

un click sul nome del progetto apre la pagina online ed avvia l'esecuzione

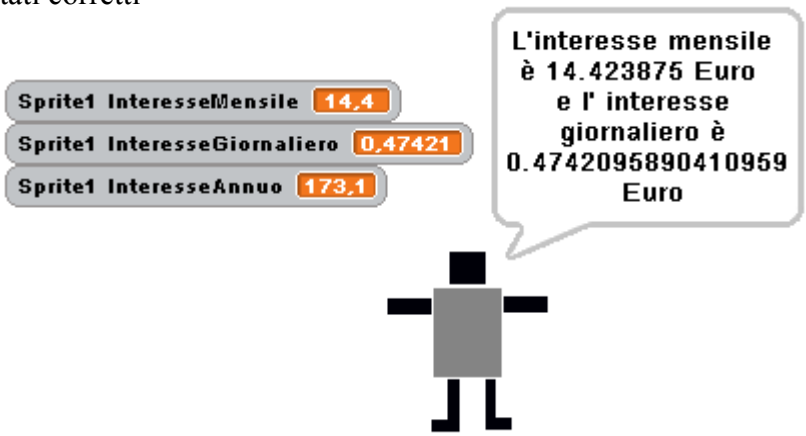


La stessa pagina è accessibile a tutti nel web all'indirizzo:

http://scratch.mit.edu/projects/pragionieri/1772692

Mentre nel test di funzionamento in locale, risulta indifferente usare come separatore delle cifre decimali la **virgola** oppure il **punto**, si verifichi come, eseguendo online, sia essenziale usare la notazione all'inglese cioè il punto. Infatti usando il **punto** come separatore decimale:

si ottengono risultati corretti



Invece usando come separatore delle cifre decimali **la virgola**

i risultati dei calcoli sono **errati** : online si ottiene sempre **zero** poiché i **valori** non sono interpretati come numeri ma come stringhe

Soluzione migliorata, esprimendo i risultati in Euro con solo due cifre decimali alla pagina:

http://scratch.mit.edu/projects/pragionieri/1784757

Oltre a creare pagine che **linkano** all'indirizzo del progetto online[6], si può anche **incorporare** il progetto stesso come ***object*** in una pagina web. Per un esempio si apra la pagina salvata nel sito di classe all'indirizzo:

http://professoressa.altervista.org/Dispense_I/Calcolo_Interessi.html

6 Solo registrandosi si può archiviare un progetto Scratch in modo da poterlo sia condividere con accesso diretto al sito, sia incorporare in una pagina web sia linkare da una pagina web. Si è già annotato, infatti, che il progetto (gli script e tutti gli oggetti multimediali associati) è un'**applet Java**, inserita in pagina html dinamica.

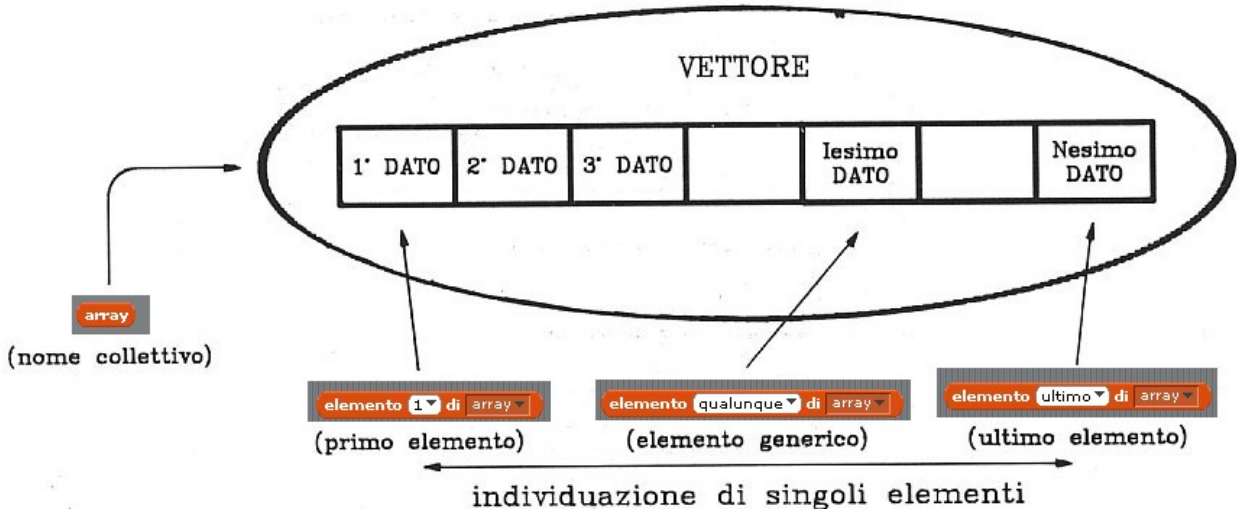
Variabili strutturate: array

Una matrice (**array**) è un *insieme ordinato* e *omogeneo* di dati: è una variabile che contiene uno o più valori in sequenza ordinata. Questi valori si chiamano "elementi" e devono avere lo stesso tipo di dato. I singoli elementi sono individuati tramite indici.

Variabile di tipo strutturato, dunque, in quanto l'indice introduce delle relazioni all'interno dell'insieme, stabilendo un nuovo ordine che non è necessariamente quello dei valori degli elementi componenti. Il tipo dell'indice deve essere scalare, non reale perché, per ogni elemento, deve essere possibile definire "il precedente" ed "il seguente".

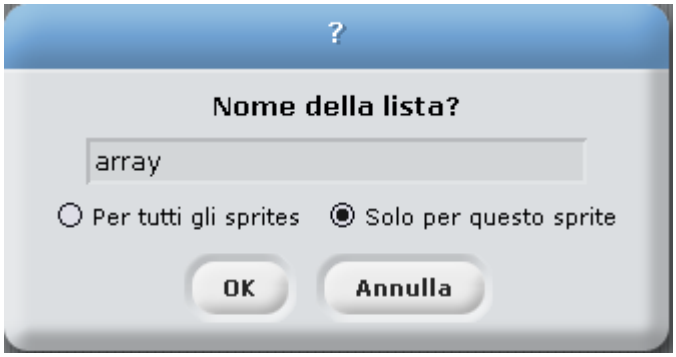
Servono dunque per usare un **unico nome collettivo** per riferirsi a molti elementi.

Il *nome* dell'insieme è l'*indirizzo* della locazione di memoria dove è memorizzato il valore del primo elemento; gli altri elementi della lista occuperanno locazioni di memoria consecutive.



Array in Scratch

Nel ***frame*** di sinistra, nella categoria Variabili si selezioni Nuova lista e si scelga il nome



Si potrà definire, come nel caso di variabile elementare, globale o locale cioè **lista** visibile da tutti gli sprites o solo dallo sprite attuale. Si consiglia la scelta di **liste locali**.

Nel ***frame*** di sinistra è comparso, smarcato, il nome della lista e le tessere relative.

Operazioni su array in Scratch

Tipiche operazioni su array i cui elementi sono valori numerici (per dimensioni dell'array non nulle)

- **Resettare** cioè porre a zero il valore di tutti gli elementi

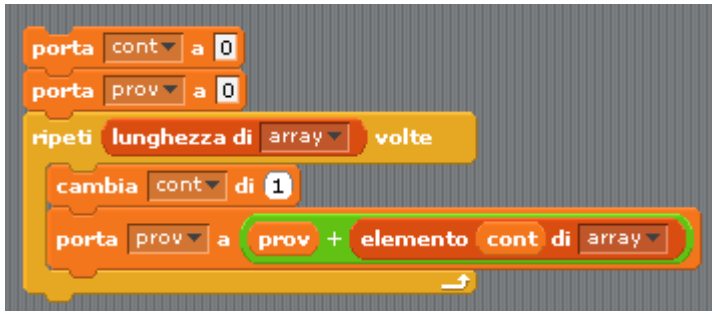
- **Inizializzazione *casuale*** dei valori

- **Inizializzazione** dei valori scelti dall'utente

Con uso di **variabili elementari**

- **Sommare** *(iterativamente)* il valore di tutti gli elementi :

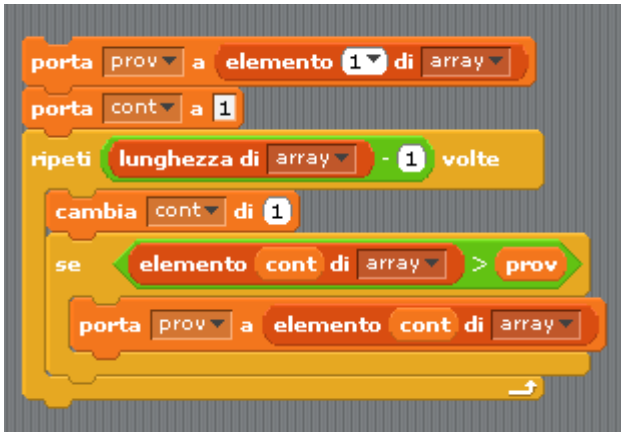
{
cont ← 0
prov ← 0
mentre (*ci sono elementi*)
cont ← cont + 1
prov ← prov + *el_attuale*
}
*ora nella variabile **prov** cè la **somma***
*e nella variabile **cont** il numero di elementi*



- **Calcolare la media** dopo averne calcolato la somma :
prov_*valore medio* ← prov_*somma* / cont

- Ricercare il valore **massimo** tra i valori dell'array in modo ***efficiente***

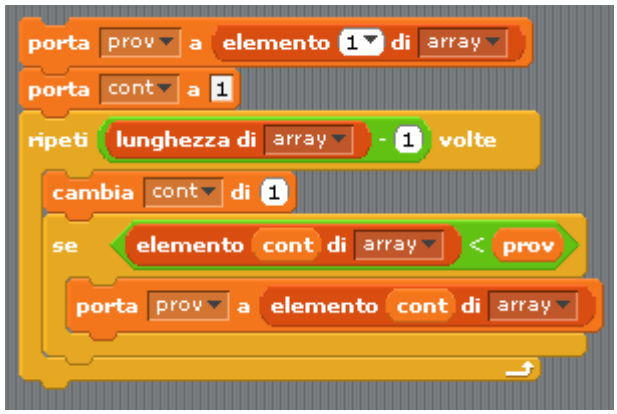
{
prov ← *el_primo*
cont ← 1
mentre (*Nvolte* <= *lunghezza array* -1)
cont ← cont + 1
se (*el_attuale* > prov)
prov ← *el_attuale*
}
*ora nella variabile **prov** cè il **massimo***
*e nella variabile **cont** il numero di elementi*



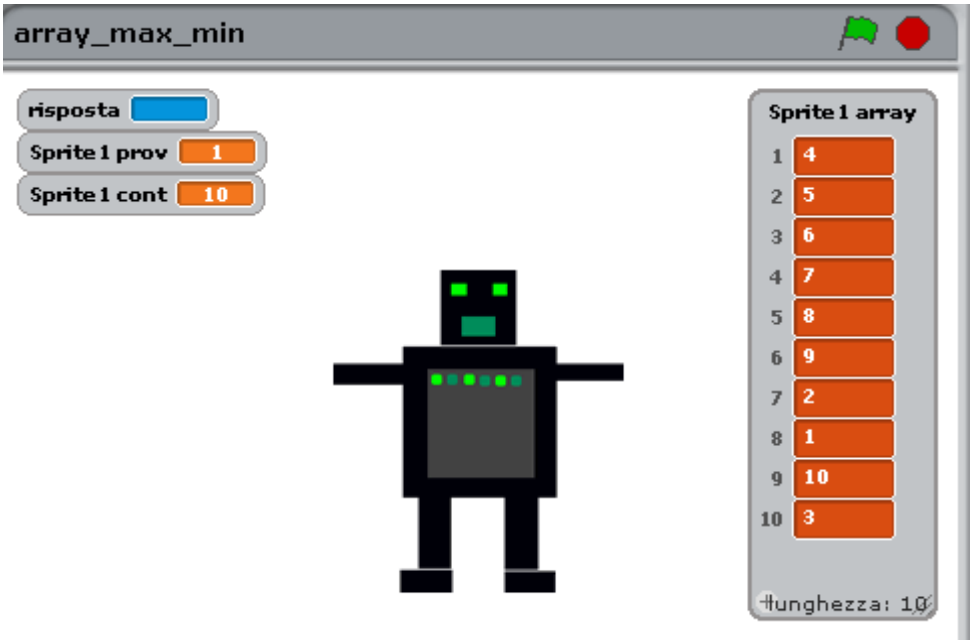
- Ricercare il valore **minimo** tra i valori dell'array in modo ***efficiente***

```
{
  prov ← el_primo
  cont ← 1
  mentre (Nvolte <= lunghezza array -1)
          cont ← cont + 1
          se ( el_attuale < prov)
              prov ←  el_attuale
  }
```

*ora nella variabile **prov** cè il **minimo***
*e nella variabile **cont** il numero di elementi*



Effetto dopo la creazione di un array di 10 elementi, l'inizializzazione con numeri interi digitati dall'utente ed il calcolo del **minimo**



Di seguito un'implementazione **errata** nella ricerca del massimo

se i numeri sono tutti negativi

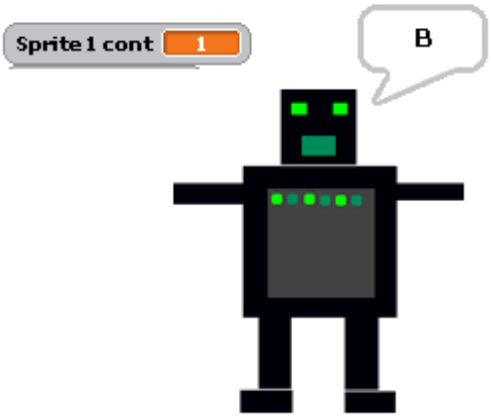
fornisce come **massimo** il **valore zero**

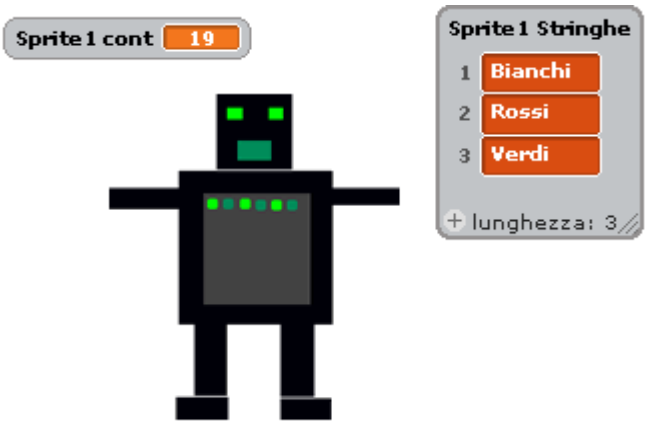
si noti anche la *minor efficienza*:
si compiono tanti confronti quanti sono gli elementi dell'array

Operazioni su stringhe in Scratch

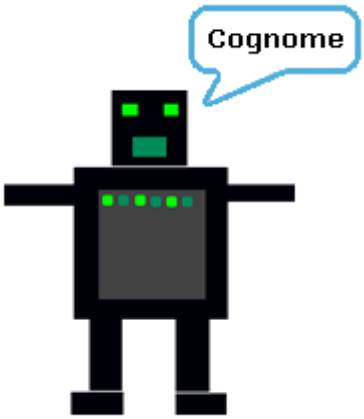
- Inizializzare l'array con ***oggetti stringhe***

- Poter elaborare il **singolo carattere** verificando che le stringhe[7] sono accodate inserendo tra l'una e la successiva un carattere di separazione che non viene visualizzato nella comunicazione dello sprite ma risulta dal conteggio dei singoli caratteri



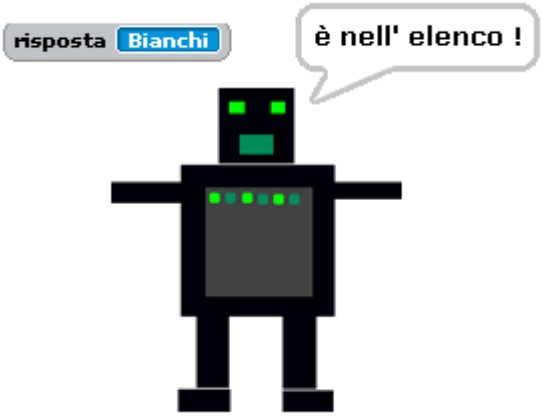


- Poter elaborare il singolo ***oggetto stringa***



esempio di *ricerca e inserimento*

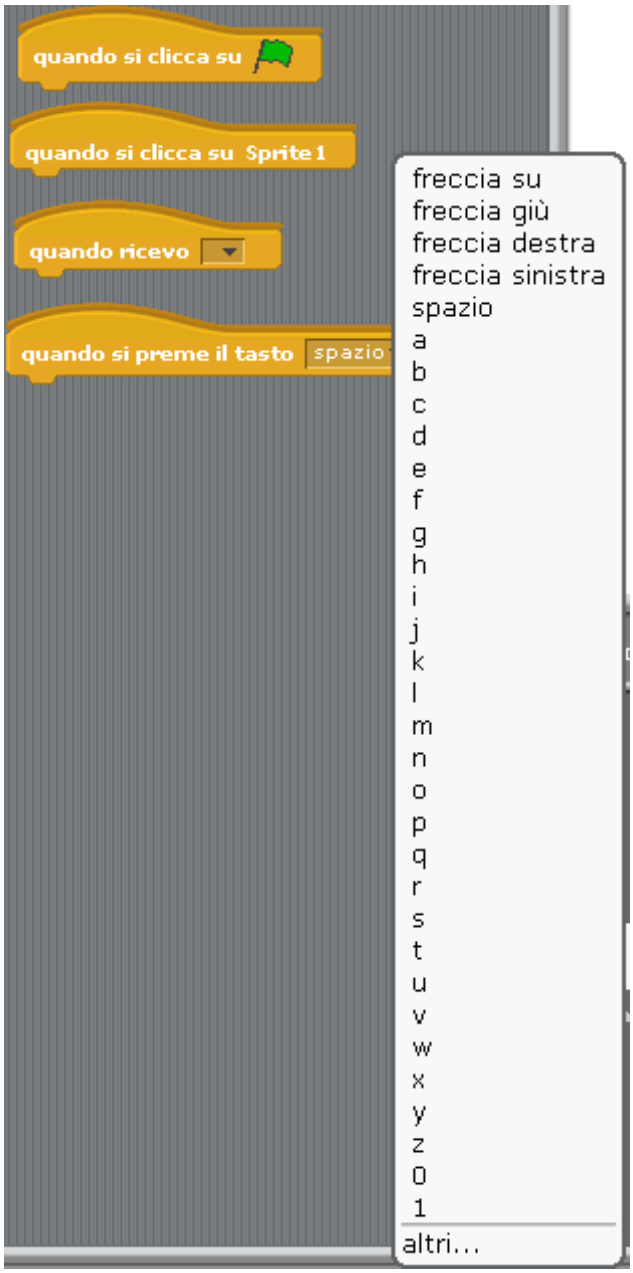
possibile *cancellazione*



7 Stringa : sequenza finita di elementi sintatticamente concatenati in una frase

Programmazione *event-driven*: esecuzione di procedura quando si verifica un determinato evento

Incastrare sopra un *blocco* un controllo del tipo "**quando**"

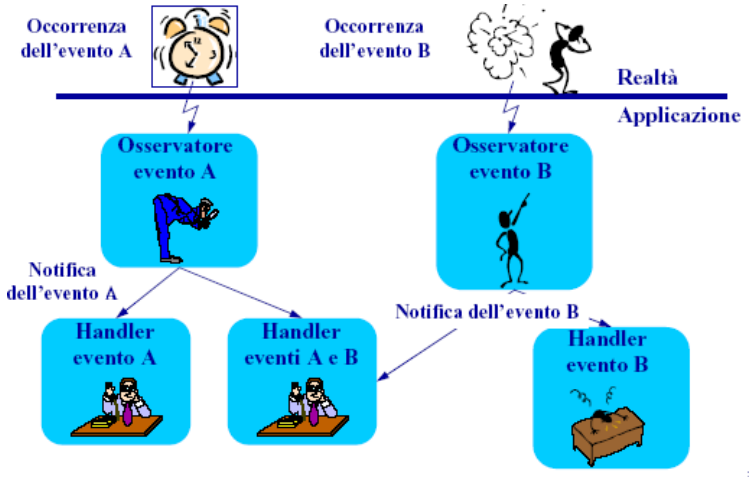


corrisponde ad assegnare un nome alla **procedura-evento** (***handle code***) che costituirà la **risposta** ad un **evento**.

Un ***evento*** è un messaggio che il sistema genera in risposta a un'azione effettuata generalmente dall'utente quando interagisce con l'applicazione.

Gli eventi vengono gestiti dal sistema, che effettua per prima cosa alcune operazioni predefinite, e quindi passa il controllo a una procedura, detta **procedura-evento**, con la quale il programmatore specifica il comportamento dell'applicazione in risposta all'evento.

Gli eventi più comuni sono quelli generati dal mouse o dalla tastiera. Ad esempio la pressione dello *spazio* è l'occorrenza dell'evento (Realtà) a cui è associato un messaggio generato dal sistema a cui l'applicazione risponde eseguendo lo **script**



Si parla di **modello a delega** quando il programmatore può decidere a quale ascoltatore "delegare" la gestione dell'evento.