Un concetto importante: la clonazione

La clonazione permette agli sprite di creare delle copie di se stessi mentre il progetto è in esecuzione, utilizzando il blocco qui a lato che trovi nei blocchi Controllo

Quando viene creato, un clone mantiene lo stato dello sprite originale:

- posizione x,y
- direzione
- visibilità (mostra/nascondi)
- colore e dimensione penna
- effetti grafici (colore, fantasma, etc.)
- dimensione, etc.

Quindi, se provi a cliccare su **[Crea un clone di me stesso]** (Controllo), non sembra accadere nulla: è stato creato un clone, ma è esattamente identico e nella stessa posizione di quello originale.

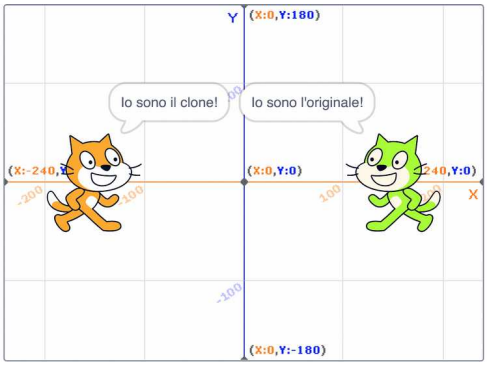
Dopo la creazione, lo sprite originale e i suoi cloni possono evolvere in maniera indipendente, quindi cambiare posizione, direzione, etc.

I cloni condividono il codice dello sprite originale. Quindi, se nel codice c'è un blocco Situazione, come ad esempio [Quando si preme il tasto spazio], tutti i cloni reagiranno alla pressione del tasto spazio.

In particolare, i cloni possono eseguire codice nel momento in cui vengono clonati, utilizzando il blocco qui a fianco. Nota che è l'unico blocco di reazione ad eventi che non si trova nella sezione Situazioni, bensì nella sezione Controllo.

Compito 19

Utilizzando un solo sprite, prova a clonare il gatto e fare in modo che l'originale e la copia si guardino uno con l'altro, come nella figura a fianco. Fai in modo che l'originale e la copia dicano due cose distinte.
https://scratch.mit.edu/projects/296750256/



Una volta eseguito il codice, riesci a dire qual è lo sprite originale e qual è il clone?

Puoi utilizzare il blocco **[Cambia effetto colore di ..]** (Aspetto) per cambiare il colore del gatto.

Quand’è che conviene clonare?

L’esercizio 19 serve solo per prendere confidenza con i cloni. Se avete due attori che hanno comportamenti diversi, clonare non è una buona idea: si fa solo confusione, mischiando il codice di due attori in unico sprite. Se invece avete bisogno di tante copie che si comportano tutte allo stesso modo, la clonazione è la scelta giusta.

Compito 20 – Ballo sincronizzato

Scegli uno sprite che abbia tanti costumi, come “Anina Dance”, “Champ99”, “LB Dance”. Crea un certo numero di cloni e fai in modo che tutti i cloni ballino in maniera sincronizzata.
https://scratch.mit.edu/projects/296455247/

Una volta realizzato, hai incontrato dei problemi?

- *Una delle “copie” non sembra fare nulla?*
  In realtà non è un clone, è lo sprite originale. Se utilizzi il blocco **[Quando vengo clonato]** (Controllo), il codice viene eseguito solo dai cloni, non dallo sprite originale.
- *Le copie non sono perfettamente sincronizzate.*
  I cloni vengono creati uno dopo l’altro, con un certo ritardo temporale: quindi iniziano ad eseguire il codice del blocco **[Quando vengo clonato]** (Controllo) con un certo ritardo.

Per risolvere entrambi i problemi, utilizza questo approccio:

- Al termine della clonazione, aggiungi un ciclo **[per sempre]** (Controllo) che manda un messaggio con il blocco **[Invia a tutti ..]** (Situazioni)
- Invece di reagire alla clonazione, cambia costume ogni volta che ricevi un messaggio, utilizzando i blocchi **[Quando ricevo ..]** (Situazioni) e **[passa al costume seguente]** (Aspetto)

Sincronizzazione e parallelismo

Il blocco **[Invia a tutti ..]** (Situazioni) manda un messaggio a tutti gli sprite e a tutti i cloni, i quali eseguono le azioni associate al blocco **[Quando ricevo ..]** (Situazioni) in **parallelo**. In questo modo, si può ottenere una perfetta **sincronizzazione**.

Migliora il tuo progetto!

- Aggiungi più sprite danzanti e clona ognuno di esso.
- Fai in modo che ogni sprite abbia un ritmo diverso
- Aggiungi degli effetti sonori utilizzando i blocchi Musica che trovi nelle estensioni (in basso a sinistra)

Compito 21 – Nevicata

Prova a simulare una nevicata: seleziona lo sprite Snowflake e utilizzando i cloni, prova a far cadere tanti fiocchi.

https://scratch.mit.edu/projects/296426911/

Qualche suggerimento:

- Per sempre, crea cloni del fiocco - Blocco **[crea clone di me stesso]** (Controllo)
- Riduci la dimensione del fiocco al 5% - Blocco **[Porta la dimensione a .. %]** (Aspetto)
- Fai partire i fiocchi dall'alto (attorno alla coordinata 170), facendo attenzione che non tocchino il bordo - Blocco **[vai dove y è ..]** (Movimento)
- Fai partire i fiocchi da una posizione x casuale fra -230 e +230 - Blocchi **[vai dove x è ..]** (Movimento) e **(numero a caso tra .. e ..)** (Operatori)
- Falli cadere fino a quando non toccano il bordo - Blocchi **[ripeti fino a quando]** (Controllo), **<sta toccando bordo>** (Sensori), **[cambia y di ..]** (Movimento)
- Dopo aver toccato il bordo, elimina il fiocco di neve.
  Blocco **[elimina questo clone]** (Controllo)

Variabili locali

Normalmente, quando si crea una variabile, essa viene creata "per tutti gli sprite". Significa che esiste un'unica copia della variabile e tutti gli sprite possono accedere a quell'unica copia per leggerne il contenuto o per modificarlo.

Esiste anche la possibilità di creare una variabile "solo per questo sprite" (locale). In questo caso, la variabile esiste all'interno dello sprite ed è accedibile solo dal codice associato ad esso. Altri sprite non possono accedere alla variabile, ma è comunque possibile creare una variabile locale con lo stesso nome in un altro sprite.

Nel caso della clonazione, ogni clone ha una copia della variabile. Al momento della clonazione, la variabile assume il valore dello sprite originale, ma poi ogni clone può cambiare il contenuto della propria copia, come le proprietà dimensione, direzione, etc.

Nell'esempio qui sotto, abbiamo creato due variabili locali chiamati Nome, una per il gatto, una per il cane. Scratch aggiunge il nome dello sprite nella descrizione della variabile.

I fiocchi cadono tutti insieme! Come faccio ad evitarlo?

Ogni fiocco deve avere la propria velocità, che deve essere inizializzata casualmente alla creazione.

- Crea una variabile chiamata “Velocità”, cliccando su sulla spunta “Solo per questo sprite”
- Quando il fiocco viene clonato, inizializza Velocità ad un valore casuale compreso fra 1 e 3 Blocco **[porta .. a ..]** (Variabili)
- Per fare cadere il blocco, invece di scrivere un numero fisso, metti il blocco **(Velocità)** che trovi fra le variabili.

In questo modo, ogni fiocco avrà la sua velocità e la scena sarà molto più realistica!

Compito 22 – Nevicata con l’ombrello

Aggiungi un personaggio con l’ombrello (disegnalo tu!) e fai in modo che oltre a sparire quando toccano il bordo, i fiocchi spariscano anche quando toccano l’ombrello. Così il nostro pinguino non si raffredderà.

Suggerimenti:

- Dopo aver spostato il fiocco, se sta toccando l’ombrello, chiama il blocco **[elimina questo clone]** (Controllo)

Qualche spunto per proseguire

Sfida 1 – Media (L’esplosione della Morte Nera)

Un buon programmatore è anche un fan di Star Wars! Ti sfidiamo a fare esplodere la Morte Nera (Death Star in Inglese) con tanto di effetti speciali. Se non ti ricordi come esplode la Morte Nera, ecco qui: https://www.youtube.com/watch?v=qniy8aDSFLA

Abbiamo preparato una bozza del progetto qui:
https://scratch.mit.edu/projects/91110167/editor

Abbiamo anche una scheda con un certo numero di suggerimenti.

Sfida 2 – Molto difficile (Strutture 3D)

Con una certa esperienza, è possibile realizzare effetti molto sofisticati. Per esempio, è possibile imitare strutture tridimensionali utilizzando solo sprite.

Per questo esercizio, non abbiamo suggerimenti – solo una scheda da realizzare passo-passo.

Esercizi addizionali su cicli e variabili

Compito 23 – La somma di Gauss

Scrivi un programma che chiede all'utente un numero n e calcola la somma dei primi n numeri.

https://scratch.mit.edu/projects/296792604/

Suggerimenti:

- Crea una variabile *Totale*, inizializzata a zero. Questa variabile conterrà il totale della somma calcolata fino ad ora.
- Crea una variabile *Indice*, inizializzata a zero. Questa variabile conterrà il valore del prossimo numero da sommare.
- Domanda all'utente un numero tramite il blocco **[chiedi .. e attendi]** (Sensori); il numero verrà memorizzato in **(risposta)** (Sensori)
- Ripeti "risposta volte":
  - Somma 1 ad *Indice* – Blocco **[cambia .. di ..]** (Variabili)
  - Somma *Indice* a *Totale* – Blocco **[cambia .. di ..]** (Variabili)
- Alla fine del calcolo, *Totale* conterrà la somma dei primi *n* numeri, dove *n* è la risposta

Digressione storica

C'è un aneddoto molto famoso collegato alla somma dei primi *n* numeri, collegato alla figura di Carl Friedrich Gauss. Si veda [1] per un breve riassunto e [2] per una spiegazione più completa.

Sarebbe quindi possibile ottenere lo stesso risultato utilizzando l'espressione qui a lato,
che corrisponde a: (*risposta* * (*risposta*+1))/2

Il pattern accumulatore

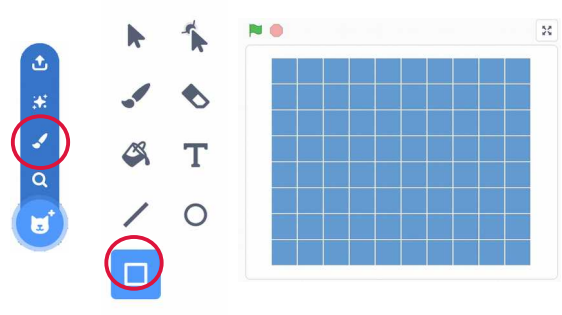
Nell'informatica, alcuni stili di soluzione dei problemi assumono una forma fissa e riproducibile, detta pattern. Ad esempio, quando si devono "accumulare" (sommare, moltiplicare, concatenare, etc.) insieme tanti dati, si utilizza il **pattern accumulatore**, che funziona in questo modo:

- Viene creata una variabile detta **accumulatore**
- L'accumulatore viene inizializzato nella maniera più opportuna
  (ad esempio, zero per una somma)
- I dati vengono letti / generati uno ad uno ed "accumulati" nell'accumulatore
- Al termine, l'accumulatore contiene il valore finale

Compito 24 – Griglia di quadrati

Utilizzando la clonazione, disegna una griglia di quadrati come quella visualizzata qui sotto.

https://scratch.mit.edu/projects/296796297/



Disegna uno sprite a forma di quadrato:

- clicca su costumi
- clicca sul gatto nella colonna dei costumi a sx
- clicca sul **pennello** per disegnare il tuo costume
- clicca sul quadrato nella palette
- traccia un quadrato con il mouse, tenendo premuto il tasto shift sulla tastiera (così sei sicuro di tracciare un quadrato, altrimenti viene un rettangolo)

Nell'esempio sopra, abbiamo tracciato una griglia di *r*=8 righe per *c*=10 colonne. Devi adattare questi numeri a seconda della dimensione del tuo quadrato.

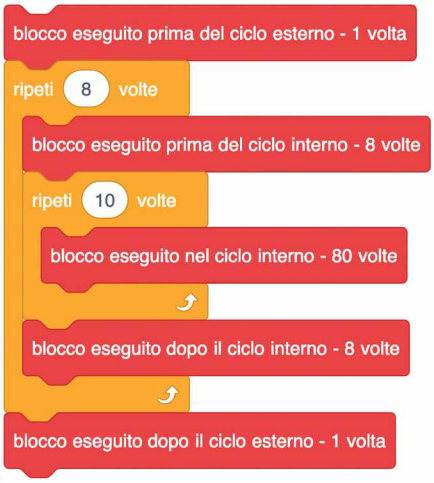
A questo punto, dovrai disegnare *r* righe; per disegnare ogni riga, dovrai tracciare *c* quadrati. Per fare questo, hai bisogno di un **ciclo annidato**.

Cicli annidati

Quando un ciclo è contenuto in un altro ciclo, in gergo informatico si dice che il **ciclo interno** è **annidato** dentro il **ciclo esterno**.

L'intero ciclo interno viene ripetuto tante volte quante sono le ripetizione del ciclo esterno. Quindi, nell'esempio qui a lato, l'intero ciclo interno viene eseguito 8 volte, mentre il blocco del ciclo interno viene eseguito 10 volte per ciclo interno e 80 volte in totale.

E' possibile (ma raro, in Scratch) dover annidare anche più di due cicli.



Una volta completato il disegno, hai 80 oggetti che hanno il proprio stato. C'è un modo molto semplice per rendere questa griglia un editor di disegni basati sui pixel:

- Crea un altro costume, duplicando quello esistente e cambiando il colore del quadrato
- Aggiungi un blocco [quando si clicca questo sprite] (Situazioni) che chiama il blocco **[passa al costume seguente]** (Aspetto)
- Aggiungi un blocco [quando si preme il tasto spazio] (Situazioni) che chiama il blocco **[passa al costume ..]** (Aspetto)

In questo modo, tutte le volte che clicchi su un quadrato, questo cambia colore. Se premi lo spazio, tutti i cloni assumerano il colore di uno dei due costumi.