Appunti per il CoderDojo Fosso+Sandon

17 maggio 2014

Presentazione del CoderDojo

Facciamo una brevissima presentazione del CoderDojo e ripassiamo i comandi di Scratch più importanti, prima di partire con qualcosa di nuovo.
Dopo il breve ripasso, vi insegneremo alcune tecniche per realizzare progetti e giochi sempre più complessi.

I mentor che vedete intorno vi aiuteranno quando sarete in difficoltà, ma è importante che prima di chiamarli voi proviate a risolvere il problema da soli.

CoderDojo è un movimento senza scopo di lucro nato in Irlanda nel 2011, rivolto a bambini e adolescenti, con il fine di organizzare incontri gratuiti per permettere di familiarizzare e imparare concretamente a conoscere gli strumenti di tecnologia informatica.
In questo momento, in altre città italiane come Roma, Milano, Bologna, altri bambini come voi stanno frequentando un CoderDojo e (compatibilmente con il fuso orario diverso...) anche negli Stati Uniti d'America altri bambini stanno imparando Scratch come voi.

Oggi impariamo ad usare meglio Scratch, uno strumento di programmazione grafico gratuito e open source sviluppato dal MIT, una prestigiosa università americana.
Scratch permette, agevolmente e in modo completamente grafico, di sviluppare programmi e giochi, utilizzando i fondamenti degli algoritmi di sviluppo classici.
Ho usato la parola “algoritmo”, che significa un “istruzioni per far funzionare da sole le macchine”

Introduzione a Scratch

Aprite tutti Scratch. Nel caso non fosse già avviato, l'icona su cui cliccare è quella con il gatto.
Avete tutti la vostra bella schermata iniziale di Scratch?

Intanto verifichiamo che sia in italiano.
Scratch è in più lingue perché viene utilizzato da tutti i bambini del mondo.
In alto a sinistra, vicino alla scritta “Scratch” c'è un mappamondo.
Cliccando là sopra si può **cambiare la lingua**.

Con Scratch si possono creare delle storie, con personaggi animati che si muovono da soli o che controllate voi con i tasti o con il mouse.
Ricordate bene però una cosa: “programmare un computer significa dirgli

esattamente cosa deve fare, passo dopo passo".

Vedete il gatto?
Provate a spostarlo con il mouse. Il gatto si muove sul **palcoscenico**: lo spazio bianco si chiama "stage" in inglese, o palcoscenico.
Tutti i personaggi che vogliamo muovere sullo schermo o i vari scenari, vanno messi qua.
Dopo il palcoscenico, cosa ci vuole? Ci vogliono gli attori! Cioè i personaggi. In inglese si chiamato **sprite**. Il gatto, ad esempio è uno sprite.

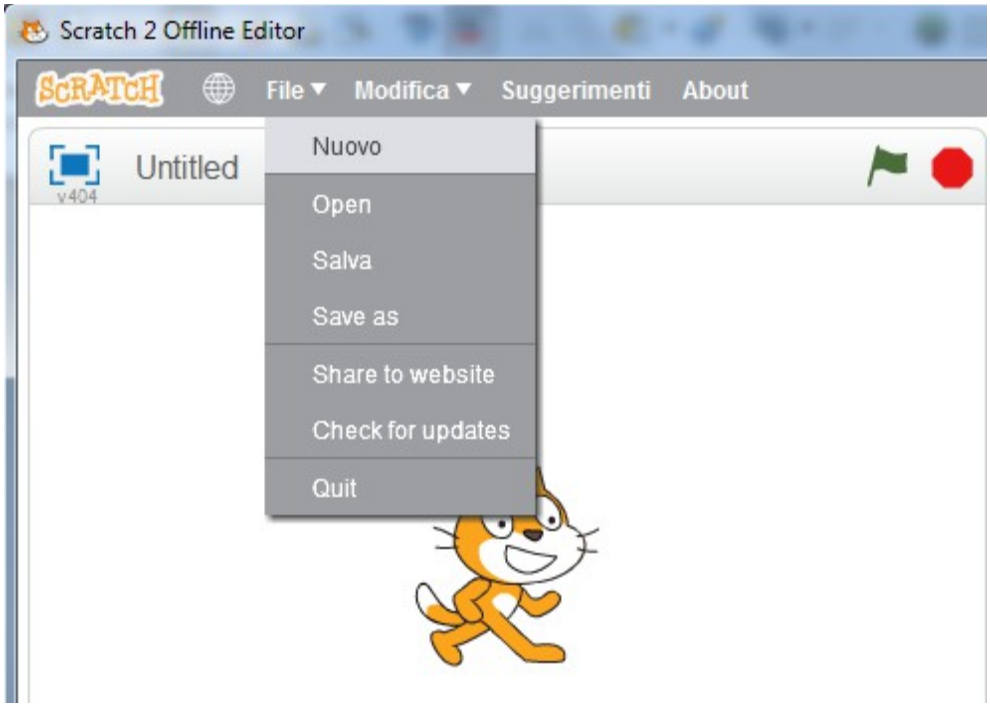
Poi guardate sopra, ci sono due tasti. Uno verde (bandierina) e uno rosso (tondo). Il verde serve per far partire il vostro gioco. Il rosso per fermarlo.

Sempre sopra c'è anche un **riquadro blu**.
Questo serve per vedere il gioco a **tutto schermo**, così vi è più comodo giocarci.

Shakesperare

Proviamo adesso a utilizzare Scratch per presentare un pezzo dell'Amleto di Shakespeare (beh, un po' modificato...).

Ma intanto accertiamoci di partire con un progetto nuovo.
Quindi andiamo nel menu “File”, ci clicchiamo sopra e poi clicchiamo sopra a “Nuovo”.



Adesso cambiamo lo sfondo, quello che attualmente è bianco.
Mettiamoci un bel castello al suo posto.
Come si fa?

Si va sull'icona a forma di cartolina, che si trova in basso a destra, dove c'è scritto “Nuovo sfondo”, e si clicca “Sfondo”.

Adesso potete vedere tutta la libreria di sfondi presenti in Scratch. Ce ne sono

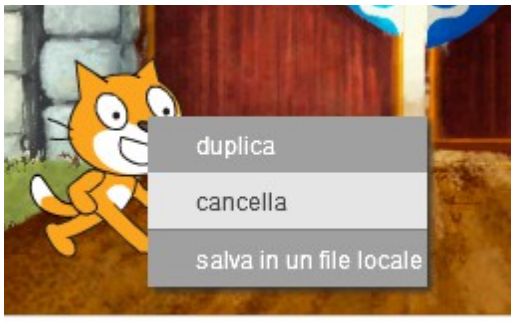
parecchi.
Sono tanti, così conviene sfogliarli cliccando sui menu a sinistra (“tutto”, “interni”, “esterni”), dove sono già suddivisi per temi.

Io consiglio di scegliere un castello, ad esempio il primo.
Prima si seleziona lo sfondo cliccandoci sopra (e appare un bordo blu attorno al fondo scelto), e successivamente si clicca su “Ok” in basso a destra.

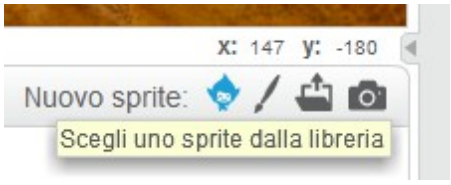
Se il palcoscenico è ancora bianco, bisogna cliccare sopra al nuovo sfondo che compare sulla sinistra (o selezionare lo sfondo e poi andare nel menù “Sfondi”, la linguetta in alto al centro):

Lo sfondo bianco si può cancellare, cliccando sulla croce nera alla sua sinistra in alto.

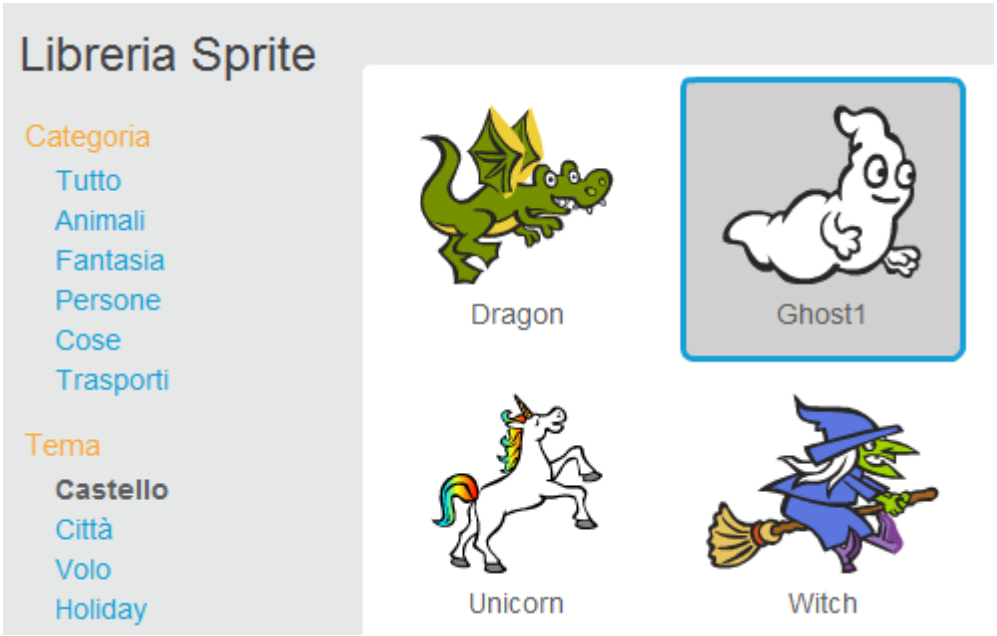
Adesso possiamo cancellare anche il gatto: ci clicchiamo sopra con il tasto destro del mouse e poi scegliamo “cancella”.



Siamo senza sprite. Cosa facciamo?
Intanto ne scegliamo un altro dal menù sprite.

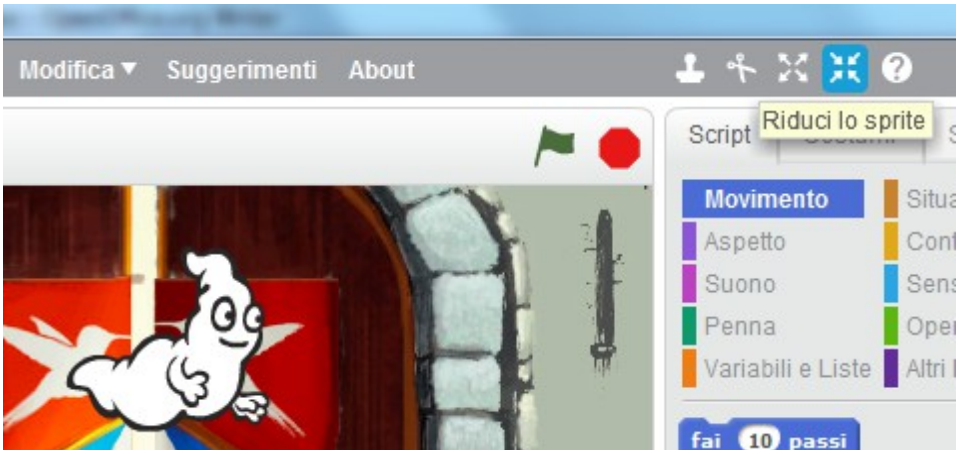


Come nel caso degli sfondi, si apre una libreria di sprite.
Io scelgo il fantasma (Tema → Castello → Ghost1)



Adesso cominciamo a programmare il fantasma.

Intanto possiamo rimpicciolirlo un po' cliccando prima sull'apposita icona in alto e poi andando a cliccare sullo sprite:

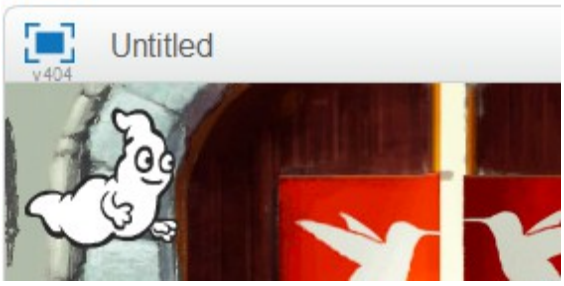


Adesso, accertiamoci di aver selezionato lo sprite del fantasma e poi andiamo sui comando sopra, che sono nel menu “Script”.

Siccome i comandi sono tanti, li hanno divisi in gruppi di comandi simili e ogni gruppi di comandi ha lo stesso colore. Quindi ricordate che i comandi simili hanno lo stesso colore:

Inoltre guardate un po' la forma dei comandi. Sembrano dei blocchi per costruire (tipo Lego). È proprio così. I comandi si incastrano tra loro. Se sono compatibili si incastrano, mentre se non si incastrano vuol dire che non sono compatibili. Insomma, vuol dire che non possono funzionare insieme.

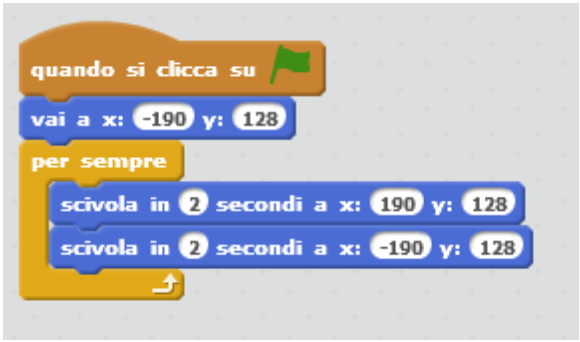
Prima di tutto selezioniamo i comandi di movimento, poi tenendo premuto il tasto sinistro del mouse sopra il fantasma, lo trasciniamo in altro a sinistra:



Adesso guardiamo come i numerini dentro al comando “vai a x... y...” cambiano mentre si muove lo sprite:

Le **coordinate** rappresentano la posizione dello sprite (in questo caso il fantasma) sul palcoscenico.
La x rappresenta la posizione orizzontale.
La y rappresenta la posizione verticale.
Più la x è un numero grande (e positiva) e più lo sprite si trova verso destra.
Più la x è un numero grande (ma col segno meno davanti) e più lo sprite si trova verso sinistra.
Più la y è un numero grande (e positiva) e più lo sprite si trova verso l'alto.
Più la y è un numero grande (ma col segno meno davanti) e più lo sprite si trova verso il basso.
Se lo sprite è nel punto x = 0, y = 0 allora significa che si trova perfettamente al centro del palcoscenico.

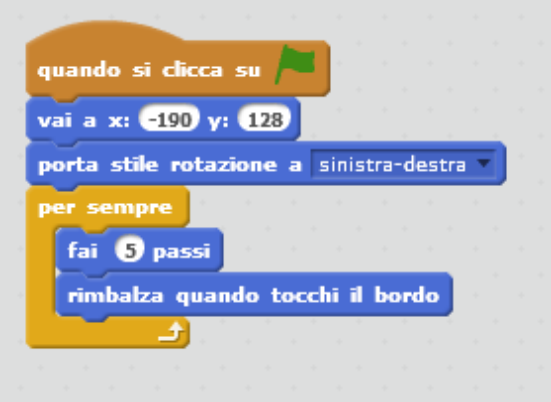
Ma torniamo al nostro fantasma: adesso lo facciamo muovere da sinistra a destra, programmandolo così:



Quando si fa partire il gioco, il fantasma si posiziona in alto a sinistra, poi “per sempre” (“per sempre” è un ciclo di controllo), scivola prima verso destra e poi torna in dietro verso sinistra.
Ricordatevi che per cambiare i valori numerici dentro agli spazi bianchi bisogna prima cliccare con il tasto sinistro del mouse dentro sopra lo spazio bianco e poi

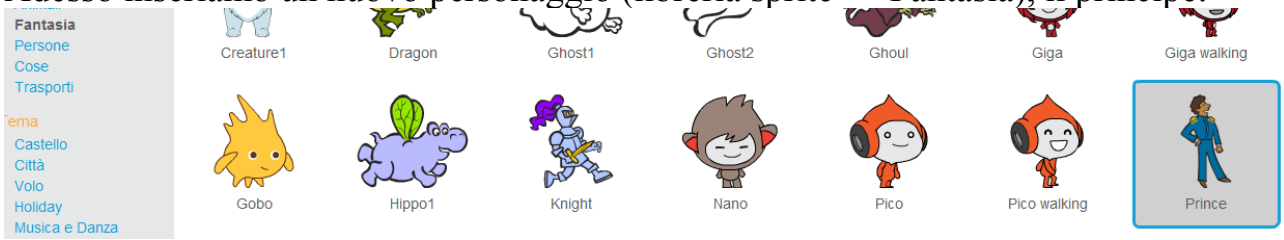
digitare quello che si vuole mettere dentro con la tastiera.

Il fantasma si muove, ma quando torna indietro sembra andare un po' in retromarcia... Quindi è meglio modificare la sua programmazione in questa maniera:

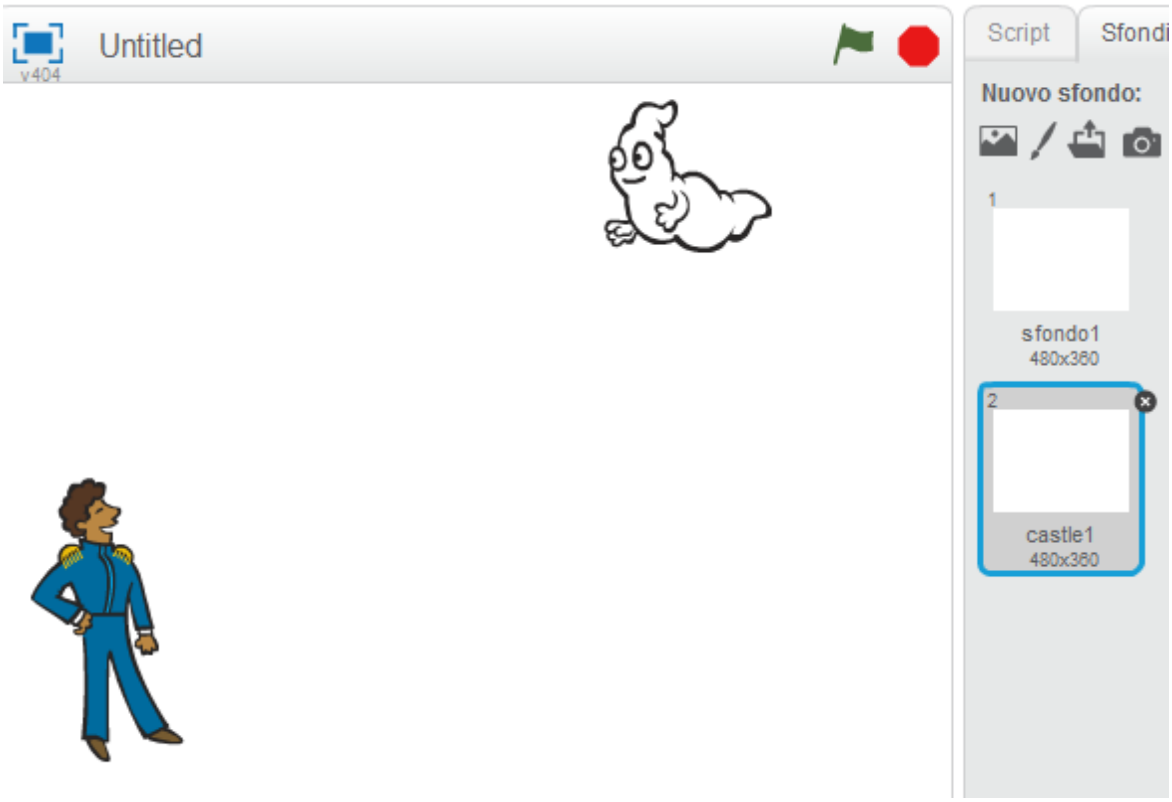


Importante: per provare gli effetti della nuova programmazione, bisogna sempre fermare in gioco cliccando sull'ottagono rosso e poi farlo ripartire cliccando sulla bandierina verde.

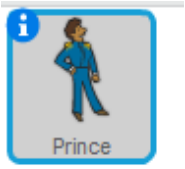
Adesso inseriamo un nuovo personaggio (libreria sprite → Fantasia), il principe:

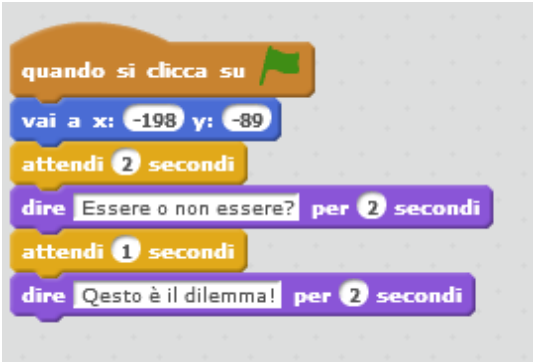


Il fondo però sembra troppo scuro e allora lo eliminiamo e ne prendiamo un altro (quello che volete...).



Adesso selezioniamo lo sprite del principe e lo programmiamo:

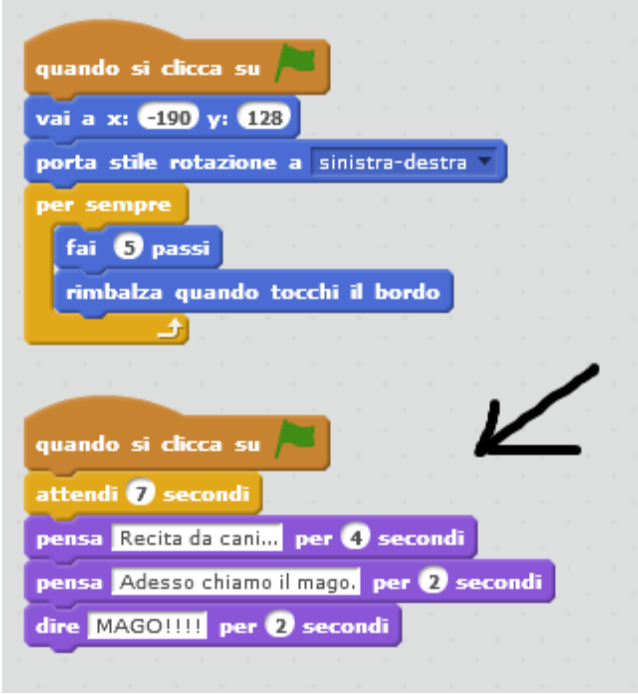




Vedete come Scratch esegue i comandi in maniere sequenziale, cioè in ordine uno dietro l'altro.
È possibile inserire delle pause (attendi... secondi).
Adesso calcoliamo quanto tempo ci mette il principe Amleto: quando il gioco inizia aspetta 1 secondo, poi dice “essere o non essere” per 2 secondi, poi attende 1 altro

secondo, poi dice “questo è il dilemma” per 2 secondi.
Cioè: 2 + 2 + 1 + 2 = sette secondi in totale.

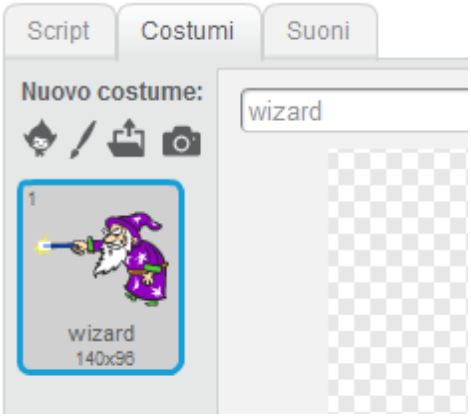
Ricordiamoci questo numero e torniamo a programmare il fantasma, aggiungendogli un blocco di programmazione (che parte dopo i famosi 7 secondi!):



Quindi il mago entra in gioco dopo 7 + 4 + 2 + 2 secondi = 15 secondi.

Prendiamo un nuovo sprite, il mago:

Andiamo su costumi:



E lo facciamo girare verso sinistra:

E torniamo su “script” (del mago).

Il mago all'inizio del gioco è nascosto. E rimane nascosto per i primi 15 secondi.
Poi compare nell'angolo in basso a destra.
Attende 2 secondi, e poi lancia un incantesimo sul principe (che dura 3 secondi).
Dopo 15 + 2 + 3 secondi (20 secondi) il principe dovrà sparire!

Intanto proviamo se il mago funziona, poi torniamo a programmare il principe.
Aggiungiamo questo blocco di programmazione al principe:

Proviamo di nuovo il gioco e vediamo che il mago ha fatto scomparire il principe.

Ma cambiamo il finale, aggiungiamo un rospo!
Andiamo tra gli sprite e cerchiamo se c'è un rospo:

Adesso programmiamolo: facciamo che rimane nascosto per 20 secondi e poi appare al posto del principe.

Programmiamo il rospo e vediamo il risultato.

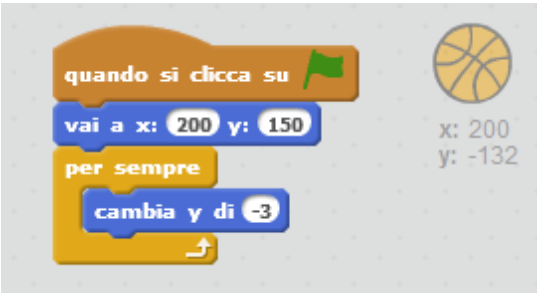
Tennis contro Basket

Iniziamo un nuovo progetto: andiamo a cliccare sul menù File e scegliamo Nuovo.
Eventualmente salviamo il progetto su cui stavamo lavorando, con File → Save As.
Cancelliamo lo sprite del gatto e ci ritroviamo con lo sfondo totalmente bianco.
Scegliamo adesso un nuovo sfondo.
Io prendo un muro di mattoni, perché ricorda le partite di basket all'aperto.
Come nuovo sprite, scegliamo il pallone da basket (Libreria sprite → cose):

Adesso facciamo entrare in gioco un nuovo fattore: la gravità (in versione molto moto semplificata).
La forza di gravità è la forse che attira tra di loro due oggetti, a seconda della distanza e della loro massa.
Gli oggetti cadono perché la nostra terra è molto grande e attira a se tutti gli oggetti che le stanno intorno.
Quindi la prossima volta che fate cadere un vaso di vostra mamma, potete dar colpa alla forza di gravità!

Intanto rimpiccioliamo un po' il pallone...

Quando si inizia il gioco, facciamo che il pallone da basket sta nell'angolo in alto a destra, e “per sempre”, comincia a cadere:



Se facciamo partire il gioco vediamo che il pallone cade finché non arriva per terra.
Adesso facciamo un piccola modifica: vogliamo che, quando il pallone arriva a terra, ricominci a cadere dall'alto:

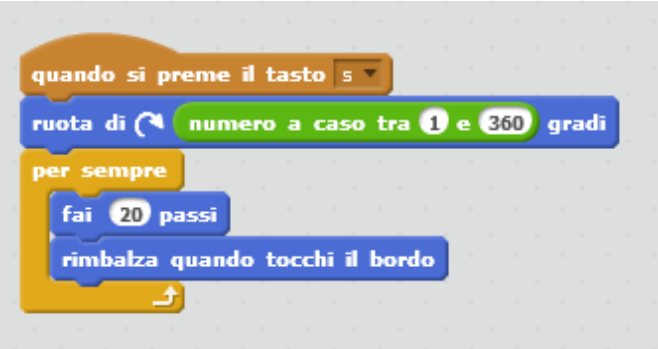
Con questa modifica il pallone ripete il suo ciclo di caduta ogni volta che tocca il pavimento.
Proviamo pure.
Ora aggiungiamo un nuovo sprite al gioco.
Prendiamo una pallina da tennis (Cose → Tennis Ball).


Tennis Ball

La rimpiccioliamo in maniera adeguata e cominciamo a programmarla.
Vogliamo poterla lanciare verso il pallone da basket.

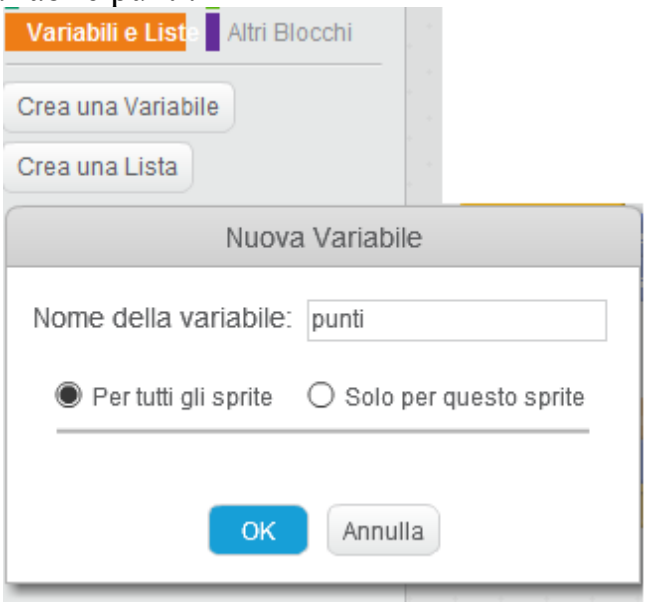
Intanto facciamo che si posizioni subito all'inizio del gioco:

Poi facciamo che quando viene premuto il tasto “s”, la palla da tennis si lanci in una direzione casuale:



Proviamo un po'.
Adesso inseriamo un nuovo comando: facciamo che premendo il tasto “spazio”, la palla da tennis torni nella posizione iniziale:

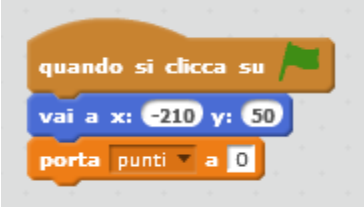
Adesso creiamo la variabile punti:



Una variabile è una cella di memoria in cui il computer tiene da annotato una particolare informazione (un nome, una password, il punteggio di un gioco, ecc.).

In questo caso noi terremo nota di quante volte la pallina da tennis colpisce il pallone da basket.

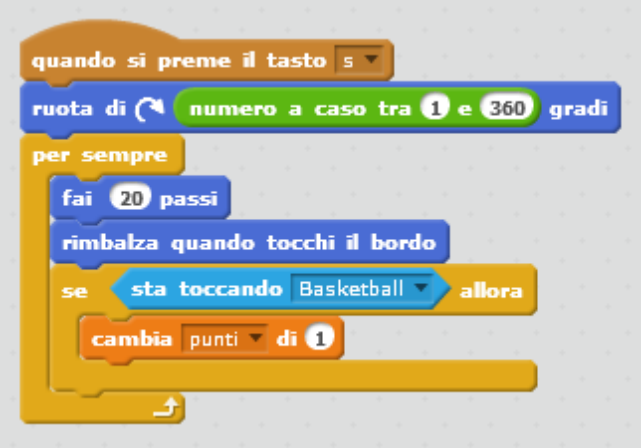
Dopo aver creato la variabile, modifichiamo il primo script:



cioè azzeriamo la variabile quando inizia il gioco (all'inizio del gioco il punteggio è zero).

Inseriamo poi questo frammento di script:

all'interno dello script creato in precedenza:



E facciamo un ulteriore aggiunta, che dice a Scratch di interrompere la corsa della pallina da tennis e di riportarla al punto iniziale quando questa tocca il pallone da basket.

Riassumendo, se la pallina da tennis tocca il pallone da basket succedono le seguenti cose:

- il punteggio aumenta di 1 punto;
- il movimento della pallina da tennis si arresta;
- la pallina da tennis viene riportata al punto di partenza e rimane ferma (in attesa che venga premuto “s” di nuovo).

Aggiungiamo ora un nuovo elemento: un timer.
Praticante un conto alla rovescia che parte da 200 (o 1000, o altro numero che volete) e che arriva a zero.
Quando arriva a zero possiamo decidere cosa fargli fare.

Siccome questo timer riguarda tutto il gioco, lo mettiamo all'interno degli script dello sfondo.
Selezioniamo pertanto la programmazione dello sfondo:

e creiamo una nuova variabile (“timer”):

Adesso programmiamola così (siamo sempre nella programmazione dello sfondo):

Vediamo in dettaglio:
Quando si clicca sulla bandierina verde (partenza del gioco), il timer viene impostato a cinquecento. E poi mano a mano viene decrementato di 1 unità.
Quando il timer arriva a zero, allora “arresta tutto”.
Cioè il gioco si ferma.

Adesso cosa sono quei “200” del timer? Secondo? Minuti? Boh, non si capisce bene.
Anzi, viaggiano veloci o lenti a seconda del computer, forse.
Facciamo qualcosa di più preciso e facciamo che 1 unità di timer corrisponda ad un secondo.
Basta introdurre un piccolo ritardo:

Oh, adesso è tutto più preciso. È proprio un bel timer in secondi.
Ma è triste che quando il timer si azzera, il gioco finisca. Facciamo che quando il timer si azzera, lo sfondo cambi e il gioco vada avanti.

Intanto inseriamo un nuovo sfondo:

Sempre dal menù sfondi, eliminiamo lo sfondo bianco:

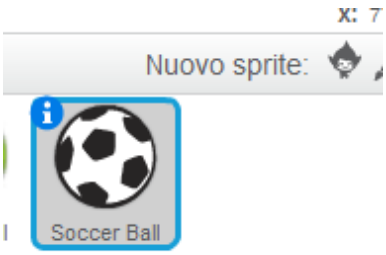
Poi portiamo il numero dei secondi del timer a 10:
Togliamo “arresta tutto” e dal menù “Aspetto”, mettiamo “passa allo sfondo seguente”.
Aggiungiamo un comando per riportare il timer a 10 secondi ogni volta che arriva a

zero (praticamente lo facciamo ripartire):

Adesso aggiungiamo un effetto sonoro ogni volta che il pallone da basket viene colpito dalla pallina da tennis.
Selezioniamo la pallina da tennis:

e aggiungiamo un comando nella sua programmazione:

Adesso andiamo su “Nuovo sprite” e aggiungiamo un pallone da calcio:



E lo programmiamo così (prima possiamo rimpicciolirlo a piacere):

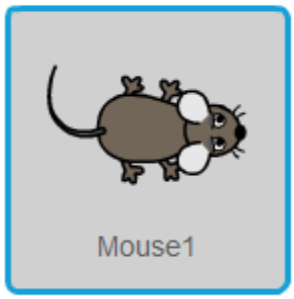
- lo facciamo partire in alto a sinistra;
- lo facciamo ruotare a caso da 1 a 360 gradi, in maniera che ogni volta parta in una direzione diversa (come abbiamo fatto per la pallina da tennis;
- gli diciamo di fare 15 passi per sempre (quindi va più veloce della pallina da tennis);
- ogni volta che tocca il bordo deve rimbalzare;
- e ogni volta che tocca la pallina da tennis cambia direzione ruotando a caso da 1 a 360 gradi.

E con questo abbiamo finito!

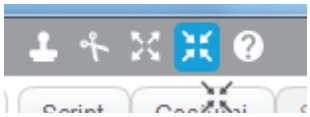
Topo nel labirinto

Adesso iniziamo un nuovo progetto.
Andiamo sul menu File e scegliamo Nuovo.
Se vogliamo salvare il progetto su cui stavamo lavorando, dobbiamo dirlo al programma usando il comando File → Save as. “Save as” significa “Salva come”, ovvero “Salva con il nome che io digiterò”.

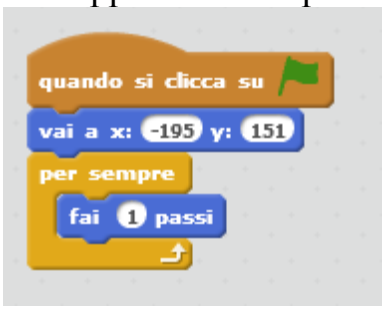
Adesso abbiano lo sfondo bianco e il solito gatto al centro.
Adesso cancelliamo il gatto (tasto destro del mouse e poi “cancella”) e dalla libreria degli sprite scegliamo un nuovo personaggio: un topolino:



Rimpiccioliamo un po' il topo con il solito comando:



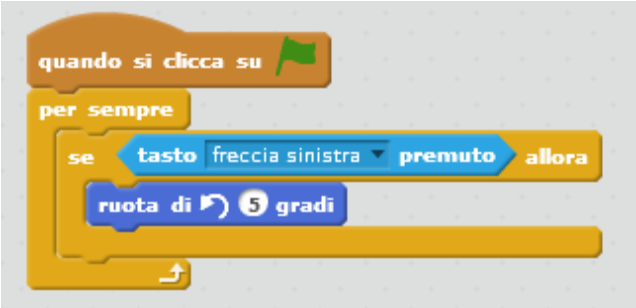
Cominciamo a programmare il topo: vogliamo che all'inizio del gioco lui si trovi in altro a sinistra dello schermo, che rappresenterà il punto di inizio del labirinto.



Adesso programmiamo i suoi movimenti.
Vogliamo che il topo si muova in continuazione, e che noi possiamo ruotarne il movimento in senso orario o anti orario.
Ricordate che questo è un labirinto visto dall'alto.

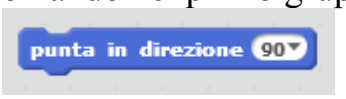
Così controlliamo i movimenti in senso orario (quando si clicca la freccia a destra):

Così controlliamo i movimenti in senso antiorario (quando si clicca la freccia a sinistra):



Adesso proviamo il gioco: clicchiamo sulla bandiera verde e vediamo che il topo inizia a camminare.
Proviamo se i tasti con le frecce lo fanno ruotare.
Tutto ok?

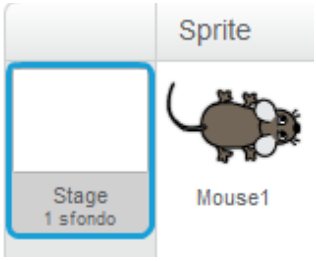
Ora, se premiamo il tasto rosso a fianco della bandierina verde e poi facciamo ripartire il gioco (bandierina verde), vediamo che il topo si posiziona in alto a sinistra (come volevamo), ma è "girato" verso la direzione in cui lo avevamo fermato con il tasto rosso.
Noi invece vogliamo che lui parta sempre rivolta a destra.
Bisogna aggiungere quindi un comando nel primo gruppo di comandi:



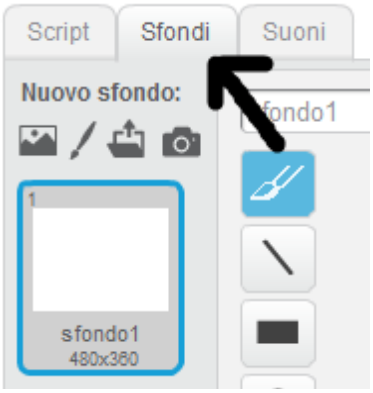
che è un comando di movimento, e lo inseriamo qui:

Adesso proviamo più volte a fermare e far ripartire il gioco per verificare che tutto funzioni correttamente.

Adesso cominciamo a disegnare il labirinto.
Selezioniamo lo sfondo bianco:

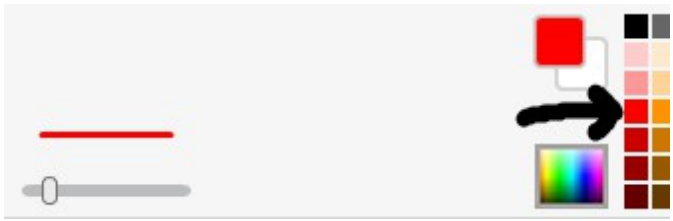


e il menù “Sfondi”:

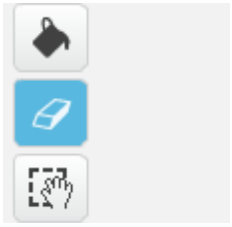


Come strumento di disegno selezioniamo la linea:

e come colore prendiamo il rosso:

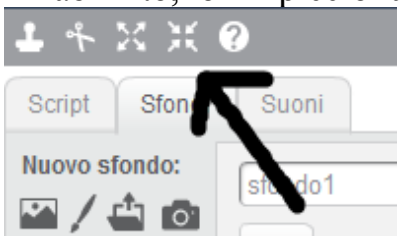


Per cancellare linee rosse (e aprire dei “varchi” nel labirinto), usiamo lo strumento “gomma”:

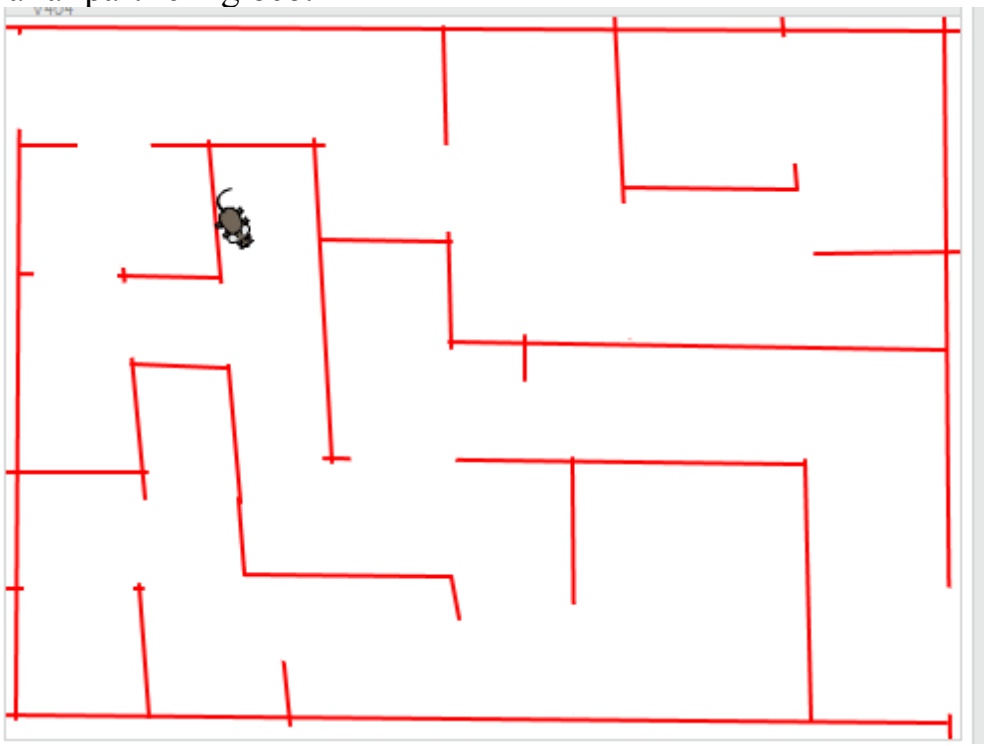


e ingrandiamo la gomma, con il comando a scorrimento che c'è sotto:

Se il topo è troppo grande per il labirinto, lo rimpiccioliamo con il solito comando:



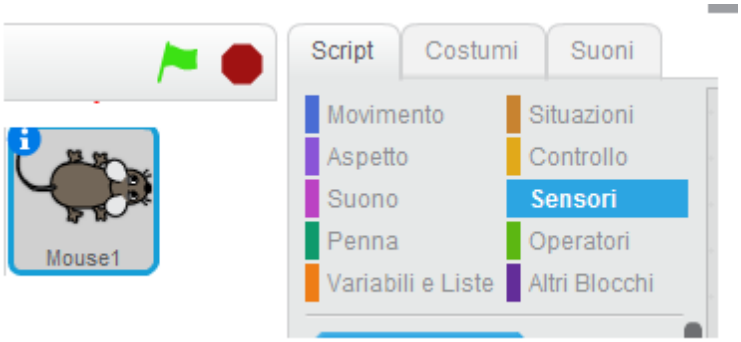
Proviamo a far partire il gioco:



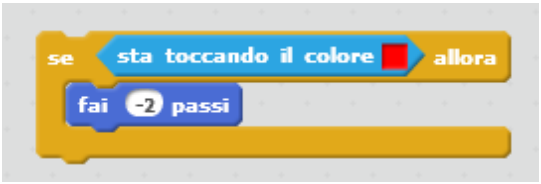
C'è un grosso problema: il nostro topo passa tranquillamente sopra i muri!!!

Dobbiamo dire al programma che quando il topo tocca il muro, si fermi, o torni un po' indietro.
Ma come faccio a far capire al programma che il topo sta toccando il muro?
Questo è un classico problema, in informatica, di “collisioni”.

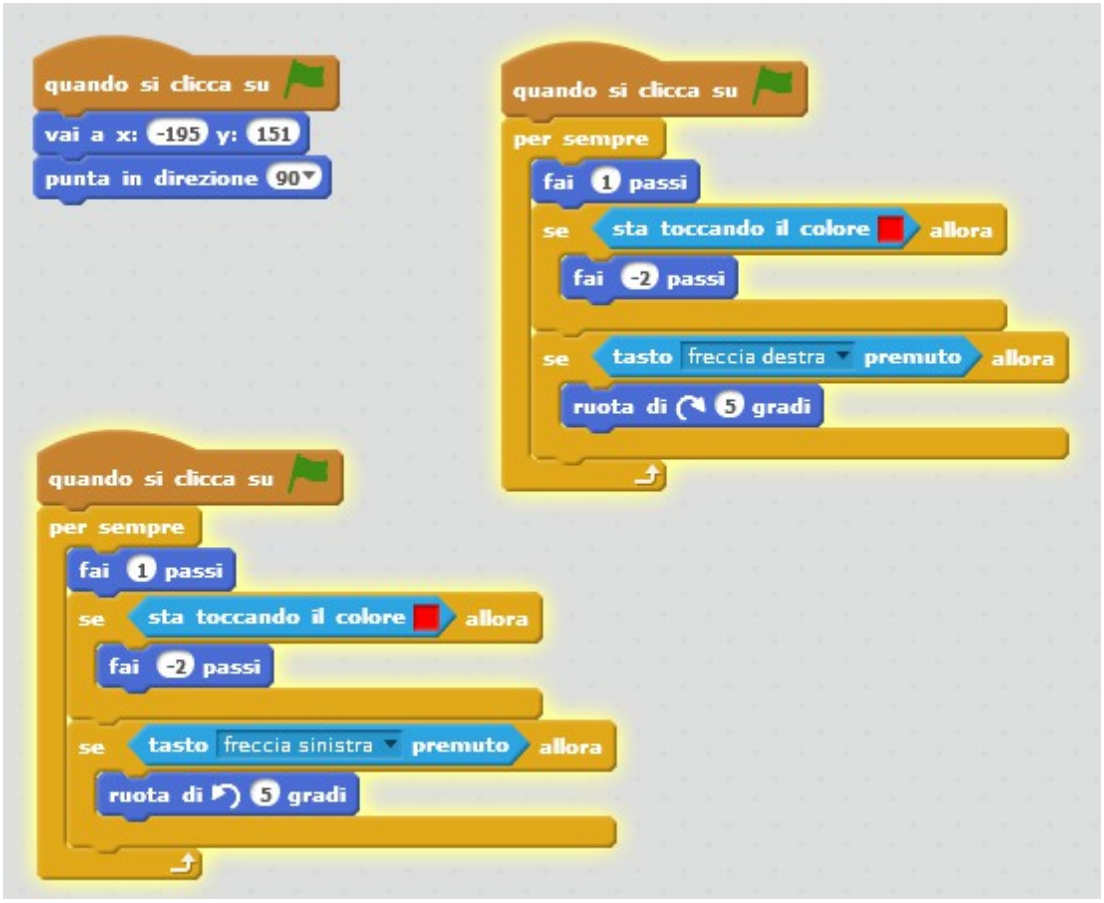
Con Scratch esiste una maniera piuttosto semplice.
Torniamo a selezionare la programmazione del topo e il menù “Script”:



Dobbiamo dirgli che se sta toccando il colore rosso (che è il colore del muro), faccia dei passi indietro (-2) come un gambero:



I comandi del topo vanno modificati così:

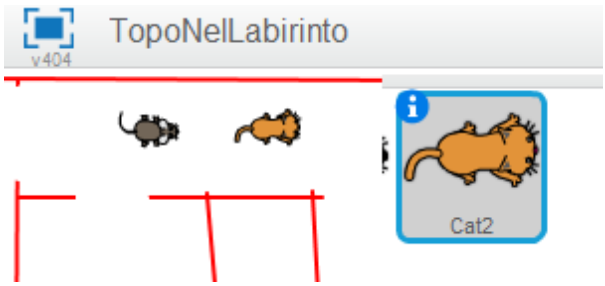


Se il topo va troppo veloce (questo dipende da computer a computer...), si può aggiungere un piccolo ritardo al termine di ogni ciclo:

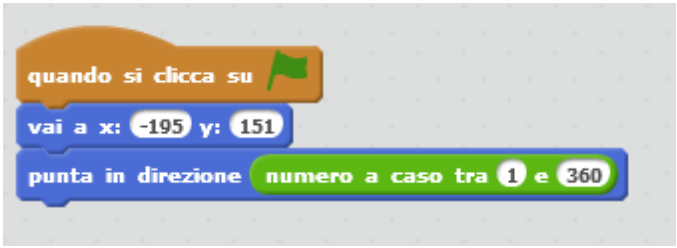
Se il topo si blocca in alcuni passaggi, e meglio selezionare di nuovo lo sfondo e, con la gomma, “aprire” meglio qualche passaggio.

Proviamo adesso ad aggiungere un secondo personaggio, che si muova da solo nel labirinto!

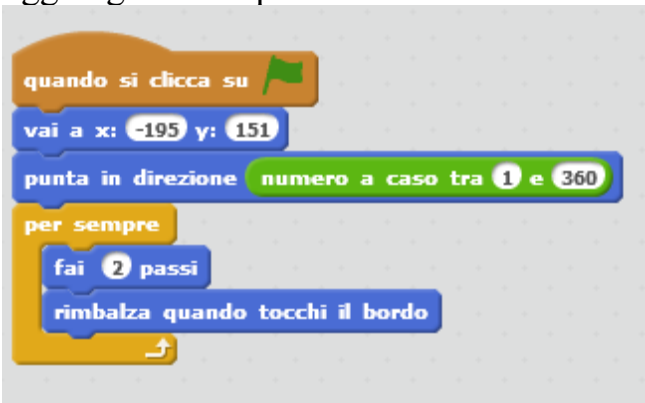
Andiamo nella libreria sprite e selezioniamo il gatto visto dall'alto.
Una volta inserito nel gioco, lo rimpiccioliamo in maniera adeguata:



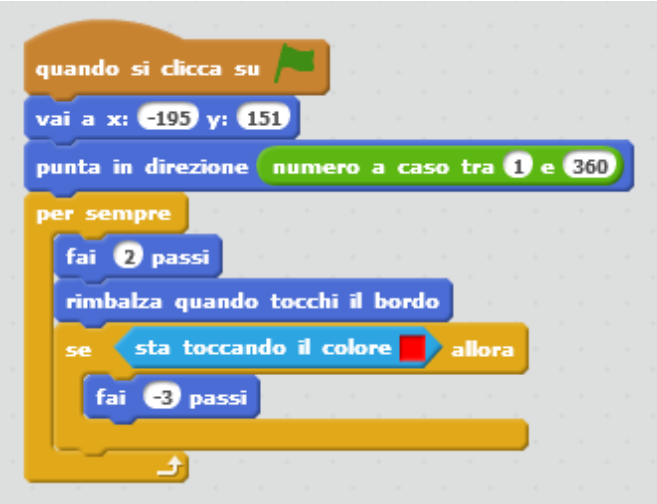
Cominciamo a programmarlo. Anche in questo caso lo facciamo partire all'angolo in alto a sinistra, ruotato in una direzione a caso:



Proviamo se va e poi aggiungiamo un po' di movimento:



Bello, ma anche in questo caso passo sopra ai muri.
Aggiungiamo qualche comando al gatto:

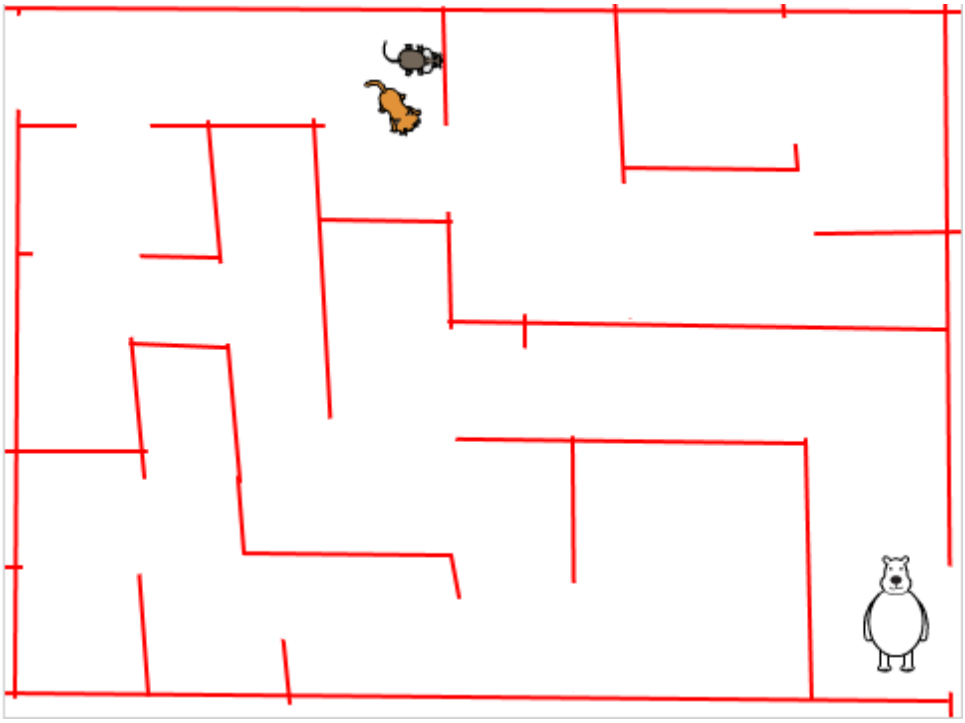


Adesso però il gatto tende a bloccarsi. Cioè se incontra un muro continua a sbatterci attorno.
Come aggirare questo problema?

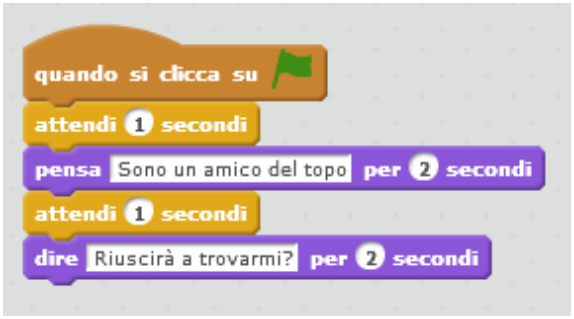
Adesso quando trova il muro, torna un po' indietro e ruota a caso (così non continua a sbattere addosso allo stesso muro).
Quanto ci metterà a trovare l'uscita?
Se lo faccio ruotare di un numero fisso di gradi, va meglio o peggio?
Ad esempio:

E mettendoci altri valori (5... 45... 55...)?

Adesso però torniamo al topo: come faccio a sapere quando arriva alla fine del labirinto?
Ci sono varie tecniche.
Proviamo questa: prendiamo un altro sprite (ad esempio un orso). Lo rimpiccioliamo e lo mettiamo alla fine del labirinto:



E programmiamo l'orso così:



Ecco che l'orso si presenta all'inizio del gioco.
Ma questo non basta.
Aggiungiamo questo:

Possibili espansioni:

- si può fare che quando il topo trova l'orso si passi ad un altro labirinto (in questo caso bisogna preparare un altro sfondo;
- si possono mettere degli oggetti nel labirinto. Se il topo tocca questi oggetti una variabile si attiva e quando il topo tocca l'orso se questa variabile non è attiva l'orso si lamenta che il topo non ha preso quel dato oggetto.