

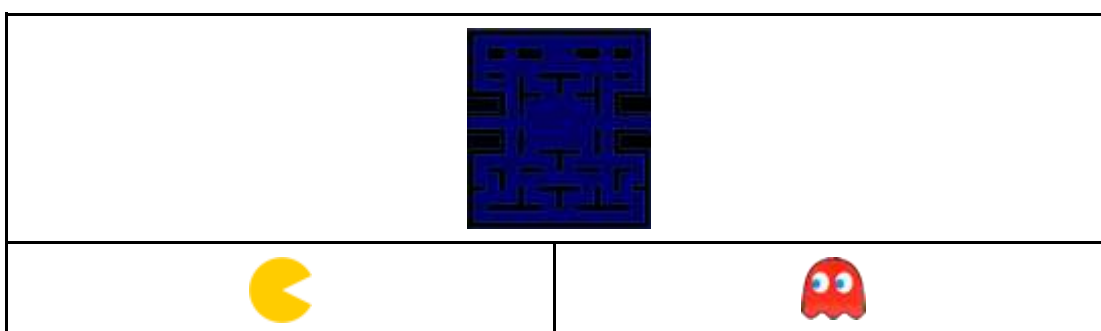
Scenario 21 – Il gioco semplificato di PacMan

Compito: programma per semplificare il gioco di Pacman. Il personaggio principale (Pacman) si muove attorno al labirinto e raccoglie le stelle. All'inizio c'è una stella in una posizione a caso, quando Pacman la raccoglie una nuova stella appare in un'altra posizione all'interno del labirinto. Pacman è inseguito da un fantasma che si muove casualmente. Il gioco termina quando il giocatore raccoglie 20 stelle o se pacman tocca il fantasma.

1. Apri il file modello:

https://snap.berkeley.edu/project?user=zapusek&project=pacman_template

All'interno puoi trovare un'immagine del pacman, del fantasma rosso, della stella e dello sfondo che rappresenta un labirinto:




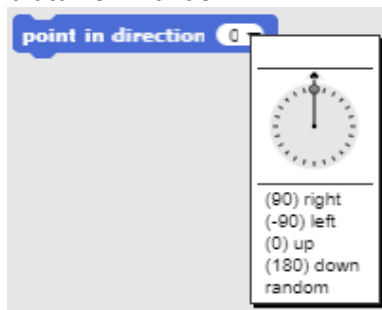
2. innanzitutto dobbiamo programmare il movimento del pacman. Il giocatore può controllare il suo movimento con i tasti freccia. Dobbiamo considerare che Pacman non può muoversi attraverso i muri.

Codificheremo il movimento usando blocchi multipli



Definiremo i tasti appropriati.

Ad ogni passo dovremo girare il Pacman nella direzione corrispondente e poi fare una mossa. Possiamo far girare un oggetto usando il blocco  e selezionare la direzione usando il menu a discesa. Non è necessario digitare gli angoli esatti delle direzioni perché Snap! ci aiuta nominandoli.




Pacman non può muoversi arbitrariamente, perché è limitato dai muri. Le pareti sono dipinte di nero ed i percorsi permessi sono dipinti di blu. Occorre implementare questo movimento: se pacman è sul percorso blu può muoversi, ma se tocca il percorso nero deve fermarsi. Ogni passaggio deve essere combinato con questa condizione.

Codifica il movimento nell'area blu del labirinto usando questi blocchi



Se scopriamo che tocca il colore nero, sappiamo che ha toccato il muro. In questo caso, dobbiamo spostarlo indietro in un'altra direzione in modo che non rimanga bloccato al muro.


I valori positivi all'interno del blocco  sposteranno l'oggetto in avanti, i valori negativi lo sposteranno all'indietro.


Aggiungi un altro blocco »Se« per verificare se pacman sta toccando il colore nero. In questo caso lo spostiamo di 5 passi in una direzione opposta.



- Successivamente vogliamo codificare il comportamento della stella. In un gioco semplificato di Pacman c'è solo una stella alla volta. Quando il giocatore la raccoglie, una nuova stella appare in una posizione casuale. Per ogni stella il giocatore ottiene un punto, quando ha un totale di 20 punti il gioco termina.

Nel gioco ci saranno un totale di 20 stelle. Implementeremo le stelle come cloni dell'oggetto stella originale.

Usiamo i cloni in situazioni in cui vogliamo usare diverse istanze dell'oggetto originale.

Il nuovo clone viene creato con il blocco . Nel menu a discesa selezioniamo l'oggetto che vogliamo clonare.

Ora dobbiamo cliccare su una stella e codificare cosa succede quando viene creato il clone dalla stella originale. Questo viene fatto usando il blocco .

Quando viene creata una nuova stella, quindi inizia come un clone, deve posizionarsi in un punto a caso sullo schermo. L'altezza e la larghezza dello sfondo sono predefinite in Snap! Ed è compreso tra -140 e 140 pixels. Se assegniamo un valore a caso tra questi due numeri per la posizione x e y del clone, otteniamo il posizionamento casuale della stella. Usa la combinazione dei blocchi:  e .

Il prossimo problema che si presenta è quando il computer seleziona tali numeri dall'intervallo in cui la stella è posizionata su un muro e Pacman non riesce ad arrivarci.

Sappiamo che il muro è rappresentato da un colore nero, possiamo usarlo per verificare se una stella è posizionata su un muro o meno. Se questo è vero, possiamo creare un altro clone ed eliminare questo.

Questo verrà ripetuto fino a quando il clone verrà collocato in una posizione sul percorso blu. Codifica usando questi blocchi:



Il giocatore può muovere liberamente Pacman nel labirinto, quindi dobbiamo controllare se ha toccato il clone in un ciclo continuo. Per rilevare se l'oggetto ha toccato un altro oggetto, possiamo usare il blocco **touching ?** e dal menu a discesa selezionare l'altro oggetto.

Quando Pacman raccoglie il clone, al giocatore verrà assegnato un punto. Implementiamo questo impostando una variabile del contatore di punti all'interno di Pacman. Quando il clone ed Pacman si scontrano, il clone invierà un messaggio al pacman, così saprà che deve aumentare il contatore dei punti.

Clicca su Pacman e crea una nuova variabile per contare i punti. Impostala a zero, perché il giocatore non ha punti all'inizio del gioco.

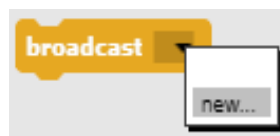
In Snap! gli oggetti possono scambiarsi messaggi per avvisare che è successo qualcosa.



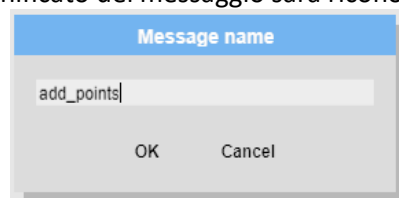
Questo si fa con il blocco

Quando il messaggio viene trasmesso, tutti gli oggetti su una scena lo ricevono. In ogni oggetto possiamo decidere quale sarà la sua risposta. Se non programiamo nulla, non accadrà nulla.

Creiamo un nuovo messaggio cliccando su una piccola freccia all'interno del campo di input del blocco e selezioniamo "nuovo"



Quindi dobbiamo nominare un messaggio. Vogliamo scegliere un nome di facile memorizzazione, quindi il significato del messaggio sarà riconosciuto dal suo nome.



Completa il codice della stella clone in modo che verificherà costantemente se il clone si è scontrato con Pacman. Quando ciò accade, dovrebbe inviare un messaggio *aggiungi_punti*. Dopo aver inviato un messaggio, il clone deve inviare un comando per creare un nuovo clone ed eliminarsi.

Per programmare questa funzione si utilizzano questi blocchi:



Vediamo come possiamo reagire al messaggio ricevuto nell'altro oggetto. Nel gruppo

"Controllo" abbiamo un blocco eventi **when I receive** che inizia la sua esecuzione quando viene ricevuto un messaggio specificato. Possiamo selezionare un messaggio specifico usando un menu a discesa.

In un codice Pacman, programma l'incremento del contatore punti, quando il messaggio viene ricevuto usando questi blocchi **when I receive** e **change by 1**.

Qui possiamo anche codificare le condizioni della fine del gioco. Se aumentiamo i punti di 1 e il punteggio totale è uguale a 20, il gioco è finito. Codifica questa funzione usando questi blocchi



- Ora dobbiamo programmare un fantasma. Il fantasma deve muoversi in modo casuale attraverso il labirinto, non deve oltrepassare il muro, ma deve rimbalzare e continuare in una direzione a caso. Se tocca Pacman, il gioco è finito.

Consideriamo come possiamo far muovere un fantasma in una direzione casuale dopo aver toccato il parete. Può muoversi nelle seguenti direzioni: sinistra, destra, su e giù.

Usando **point in direction 90** possiamo far girare l'oggetto. Possiamo scegliere la direzione desiderata inserendo l'angolo di svolta:

- 0 gradi = SU,
- 180 gradi = GIÙ,
- 90 gradi = DESTRA,
- 270 gradi = SINISTRA.

Vogliamo trovare un modo in cui un blocco punto in direzione verrebbe casualmente impostato su uno di questi valori: 0, 90, 180 or 270.

Il problema è che in Snap! possiamo ottenere un valore casuale da un intervallo e non da un elenco di valori. Se studiamo questi numeri, possiamo notare che tutti sono multipli del numero 90, perché: $0 \cdot 90 = 0$, $1 \cdot 90 = 90$, $2 \cdot 90 = 180$ and $3 \cdot 90 = 270$.

Possiamo ottenere un numero casuale dall'intervallo 0 a 3 con l'uso del blocco **pick random 1 to 10** se moltiplichiamo questo numero selezionato casualmente per 90, otteniamo uno di questi numeri: 0, 90, 180, 270. Questo è esattamente ciò che vogliamo!

Il movimento del fantasma e la verifica se ha toccato il muro devono aver luogo durante il gioco, fino a quando il giocatore raggiunge 20 punti o il fantasma tocca Pacman. Abbiamo già programmato la prima opzione che termina il gioco ora prendiamoci cura della seconda.

La condizione di uscita per spostare il fantasma è quando si scontra con Pacman. Se utilizziamo il ciclo di ripetizione continua, possiamo impostare la condizione su quando tocca il pacman. Fino a quando questo non è vero, il fantasma si muoverà e quando toccherà il Pacman, il *loop* si fermerà e i blocchi posizionati dopo il *loop* saranno eseguiti.

Il movimento fantasma è abbastanza semplice. In ogni iterazione del loop deve muoversi di 1 passo in una data direzione.

Possiamo verificare se il fantasma ha toccato il muro con il blocco di colore sensibile. Le pareti sono nere e, come abbiamo fatto con Pacman, possiamo testare ripetutamente se il fantasma sta forse toccando il colore nero. In questo caso, dobbiamo spostarlo di 1 passo indietro (in modo che non rimanga bloccato sul muro) e cambiare direzione in modo casuale come descritto sopra.

Il ciclo si fermerà quando la condizione sarà vera. Questo accadrà quando un fantasma si scontrerà con pacman. Dobbiamo fornire feedback e terminare il gioco.

Codifica questa funzione usando i seguenti blocchi

