O3 – Instructional Support Content

RACCOLTA DEGLI SCENARI DI APPRENDIMENTO BASATI SUL GIOCO PER GLI INSEGNANTI

Dati del documento

Prodotto finale: O3/A1 - Collection of game design based learning sheets targeting teachers

Risultato N. – Titolo : O3 – Instructional Support Content

Responsabile del risultato: South-West University “Neofit Rilski” (Bulgaria)

Partner coinvolti: University of Ljubljana (Slovenia), University of Rijeka (Croazia)

Traduzione: EU-Track (Italia)

Dichiarazione

Questo progetto è stato finanziato dal Programma Erasmus+ dell’Unione Europea.

Il sostegno della Commissione europea alla produzione di questa pubblicazione non costituisce un'approvazione del contenuto, che riflette esclusivamente il punto di vista degli autori, e la Commissione non può essere ritenuta responsabile per l'uso che può essere fatto delle informazioni ivi contenute.

INDICE

INTRODUZIONE

Il principale psicologo del secolo scorso ha identificato il gioco come una delle attività più importanti per lo sviluppo di abilità di vita, indipendentemente dall'età o dallo stadio di sviluppo. Il bambino, attraverso il gioco, adotta rapidamente nuove situazioni e gestisce il cambiamento con facilità. Quando gioca, scopre i concetti base della vita reale e si stabiliscono le prime relazioni fondamentali.

Oggi, i giochi sono più comunemente utilizzati a casa e all'asilo nelle prime fasi dello sviluppo di un bambino. L'apprendimento a scuola è ancora troppo spesso basato sulla trasmissione tradizionale delle conoscenze all'interno di un modello centrato sull'insegnante con studenti passivi. D'altro canto, le teorie dell'apprendimento, sviluppate nel secolo scorso, promuovono nuovi approcci all'insegnamento e all'apprendimento centrati sullo studente, basati sui problemi, diretti verso obiettivi educativi di livello superiore ordinati su livelli tassonomici più elevati, motivazionali e spesso supportati dalle TIC.

L'approccio CODING4GIRLS incoraggerà la partecipazione alle attività di programmazione attraverso un approccio "low entry high ceiling " che richiede all'inizio poche conoscenze da parte degli studenti senza, però, limitare le sfide per la risoluzione dei problemi agli studenti più avanzati. Gli studenti saranno incoraggiati a completare soluzioni parzialmente completate aggiungendo blocchi di codice mancanti o a creare le proprie soluzioni. Le attività sono pianificate in sequenza, da quelle di base con un solo concetto di programmazione a quelle più avanzate. Le attività di apprendimento preparate in Snap! sono state focalizzate sugli elementi che identificato i giochi preferiti dalle ragazze e sulle attività derivanti da problemi del mondo reale.

Le schede di apprendimento preparate presentano in modo conciso le informazioni che aiuteranno gli istruttori ad integrare i *serious game* proposti e un apprendimento basato sul *design thinking* nelle loro pratiche di insegnamento. Tali schede sono sviluppate sulla base dell'apprendimento attivo e basato sul gioco il design di apprendimento attivo di CODING4GIRLS e includono informazioni per ogni attività da sviluppare per migliorare le abilità di programmazione delle ragazze e dei ragazzi. Sono disponibili le seguenti informazioni:

- Obiettivo formativo generale della corrispondente attività di apprendimento
- Concetti inclusi nell'attività di apprendimento
- Obiettivi di apprendimento specifici
- Risultati di apprendimento attesi
- Utilizzo graduale dell'approccio di apprendimento CODING4GIRLS basato sulla progettazione di giochi
- Metodi di valutazione per la verifica delle conoscenze sviluppate
- Domande per avviare una discussione tra gli studenti nel contesto collaborativo della classe.

Sono state preparate 21 schede che corrispondono ad attività di apprendimento. Gli insegnanti possono utilizzare gli scenari e i giochi nella sequenza proposta o selezionarli liberamente in base alle proprie preferenze ed esigenze. Le schede di apprendimento coprono sia la funzionalità generica del *serious game* proposto, compresi i processi di interazione con l'utente e la generazione di feedback, sia le descrizioni di tutte le attività di apprendimento che verranno implementate nel *serious game* proposto.

Le schede di apprendimento sono disponibili in inglese e nelle lingue nazionali dei partner del progetto: bulgaro, croato, greco, italiano, portoghese, sloveno e turco.

SCHEDE DI APPRENDIMENTO

Le schede di apprendimento preparate partono da quelle di base con un concetto di programmazione a quelle più avanzate con più concetti di programmazione. La tabella seguente rappresenta l'ordine delle attività proposte.

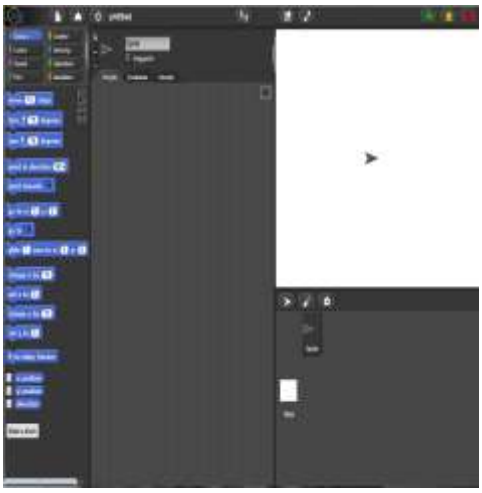
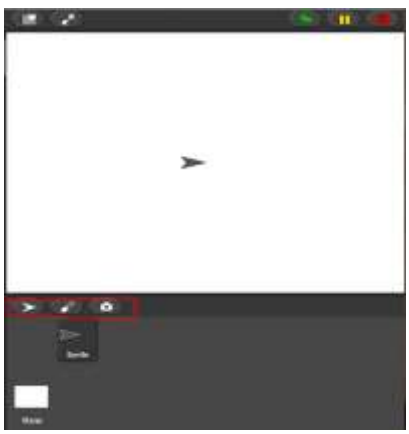
BASIC LEARNING SCENARIOS		
1	**Introduzione all'interfaccia di Snap!** Familiarizzare con l'ambiente di programmazione visiva Snap!	UL
2	**E' tempo di dare vita al tuo *sprite*** Trovare i blocchi di programmazione, connetterli in sequenza, muovere lo sprite, far dire qualcosa allo sprite	UL
3	**Muoversi sulla scena** Realizzare una sequenza significativa di blocchi	UL
4	**Cambiare costume e roteare** Realizzare una sequenza significativa di blocchi	UL
5	**Suoni della fattoria** Aggiungere, importare, registrare e riprodurre un suono	UL
6	**Le vacanze estive di Chameleon – versione semplificata** Familiarizzare con gli eventi, rilevamento del colore, valori booleani, controllare e rispondere a due diversi stati del gioco	UL
7	**Aiutare il principe e principessa a trovare i loro animali** Usare i condizionali, disegnare	UL
8	**Disegnare con il gesso** Usare i cicli, roteare, cambiare lo sfondo	UL
9	**Raccogliere la spazzatura e pulire il parco** Acquisire familiarità con le variabili, duplicare gli sprites, blocchi di codice	UL
10	**Nutrire i gatti** Usare le variabili (dentro/fuori dal ciclio), cicli, numeri casuali, concatenazione di stringhe, operatori, input	UL
11	**Indovinare il numero di gatti in un rifugio** Usare le variabili casuali, variabili di input, condizionali, operatori di comparazione, contatore	UL
SCENARI AVANZATI DI APPRENDIMENTO		
12	**Catturare cibi sani** Usare variabili, condizionali, cicli, punta in una direazione e casualità	UL
13	**Storytelling**	SWU
14	**Disegnare**	UNIRI
15	**Catturare il topo** Usare i cicli, condizionali, variabili	UL
16	**Acquistare cibo per un picnic** Usare variabili, condizionali, operatori	UL
17	**Operazioni**	SWU
18	**Riciclare**	SWU
19.1	**Suonare un piano 1**	SWU

19.2	**Suonare un piano 2**	UNIRI
20	**Test**	SWU
21	**Gioco PACMAN semplificato** Usare l'evento basato sul movimento dell'oggetto, il rilevamento del colore, valori booleani, controllare e rispondere a due diversi stati del gioco.	UL

SCENARI BASE DI APPRENDIMENTO

Scenario di apprendimento 1 – Introduzione all'interfaccia di Snap!

Titolo dello scenario di apprendimento	Introduzione all'interfaccia di Snap!
Precedente esperienza di programmazione	/
Risultati di apprendimento	Risultati generali di apprendimento: ● familiarizzare con l'ambiente di programmazione visiva Snap! Risultati specifici di apprendimento: ● lo studente è in grado di aggiungere un nuovo *sprite* ● lo studente è in grado di aggiungere un costume ad uno *sprite* e modificarlo ● lo studente è in grado di centrare lo *sprite* in modo tale che la rotazione lavori in modo appropriato ● lo studente è in grado di aggiungere un nuovo sfondo alla scena e modificarlo
Obiettivo, Compiti e Breve Descrizione delle Attività	Lo studente aggiunge un nuovo *sprite*, aggiunge un costume allo *sprite*, lo modifica e cancella uno di loro. Lo studente crea un nuovo sfondo per la scena, lo modifica ed elimina quello indesiderato. **Obiettivo: Alla fine dell'ora gli studenti disegneranno il loro personaggio favorito ed il suo ambiente di vita, reale o immaginario, per usarlo nel gioco. Per rendere l'attività più motivante per tutti gli studenti, gli *sprite* sono stati scelti in base a degli studi scientifici per adattarli a questo gruppo target.**
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Dimostrazione dell'insegnante Lavoro individuale
Didattica	Lezione frontale Lavoro individuale

Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) Alla fine dell'ora gli studenti disegneranno il loro personaggio favorito ed il suo ambiente di vita, reale o immaginario, per usarlo nel gioco. [Fase 1] Si mostra agli studenti la pagina web dove è possibile trovare Snap! (https://snap.berkeley.edu/). Si mostrano le differenti parti dell'interfaccia: sezioni con i blocchi, sezioni dove possono assemblare i codici/cambiare *i* costumi/aggiungere suoni, stage con lo *sprite*, l'elenco degli *sprite*.  [Fase 2] Si può creare un nuovo *sprite* cliccando uno dei tre bottoni: 

Si può provare a disegnare un nuovo *sprite*: cliccando sul simbolo del "pennello" si apre un *pop-up* dove si può disegnare il proprio *sprite* in modo simile a quello di *Paint*.

Compito per gli studenti: disegna il tuo primo *sprite*. Hai 10 minuti.

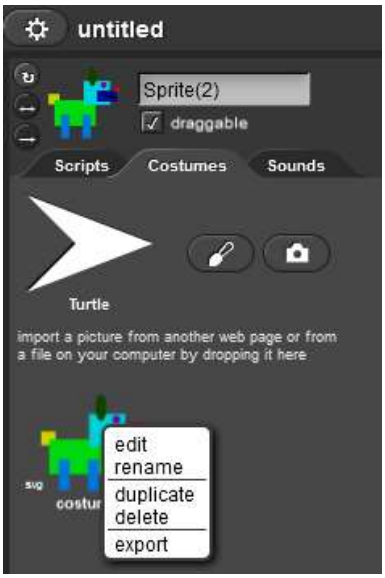
Dopo aver disegnato lo *sprite*, dovresti assicurarti che il centro di rotazione dello *sprite* sia dove vuoi che sia. Per farlo usa questo  .

Compito per gli studenti: centra il tuo *sprite*.

[Fase 3]

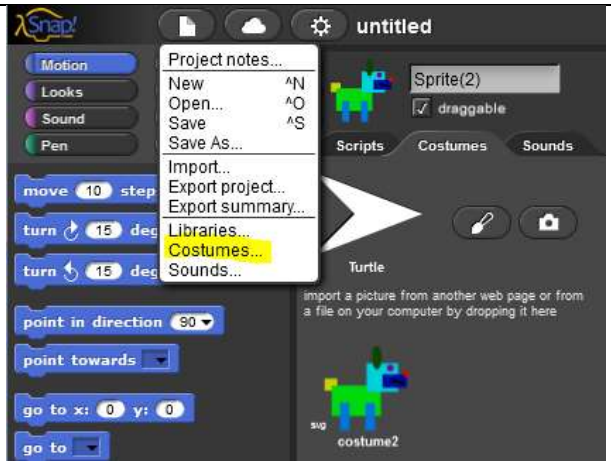
Per modificare il tuo *sprite*, scegli l'etichetta "Costumi", questo è visibile solo quando il tuo *sprite* è cliccato.

Clicca a destra sul costume se vuoi modificarlo e scegli la modifica. Puoi anche duplicare il tuo costume o cancellarlo nello stesso menu.



[Fase 4]

Per importare un costume già esistente, fai clic sull'icona simile ad un "pezzo di carta" e scegli Costumi …

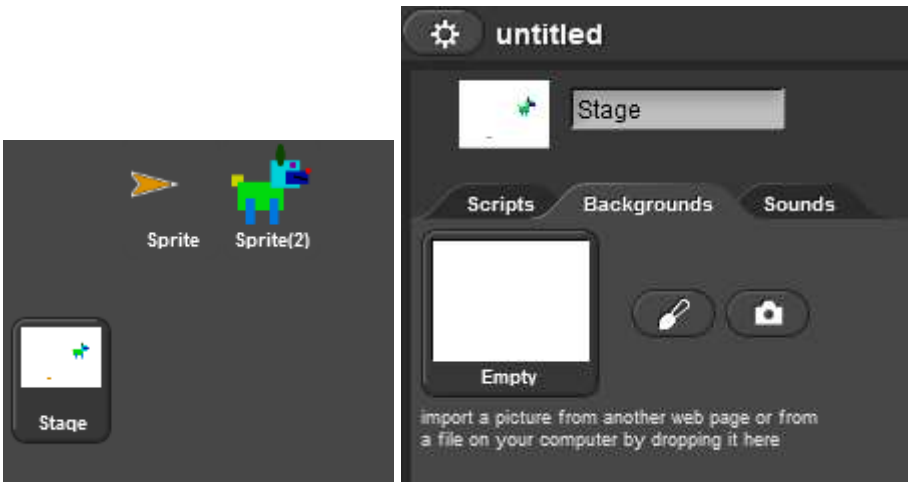


Questa opzione verrà mostrata solo quando si fa clic sul tuo *sprite* sotto lo *stage*.

Compito per gli studenti: seleziona un costume e aggiungilo allo *sprite*

[Fase 5]

Ora hai il tuo personaggio, aggiungi uno sfondo alla scena. Per farlo, prima clicca sullo *stage* invece che sul personaggio. Aggiungi un nuovo sfondo, scegliendolo dalla etichetta “Background”:



Compito per gli studenti: disegna il tuo sfondo.

Compito per gli studenti: tra gli scenari esistenti individuane uno ed aggiungilo importandolo così da averne due.

Compito per gli studenti: modifica il tuo sfondo.

Elimina uno dei tuoi sfondi, in modo che ne rimanga solo uno.

	Riflessione e valutazione: Gli studenti sono riusciti a disegnare il loro personaggio e l'ambiente in cui vive? Hanno avuto problemi? Come li hanno risolti?
Strumenti e risorse per l'insegnante	https://snap.berkeley.edu/
Risorse/materiali per gli studenti	Istruzioni per lo studente (C4G1_InstructionsForStudent.docx)

Scenario di apprendimento 2 – E' tempo di dare vita al tuo *sprite*

Titolo dello scenario di apprendimento	E' tempo di dare vita al tuo *sprite*
Precedente esperienza di programmazione	/
Risultati di apprendimento	Risultati generali di apprendimento: ● lo studente sa dove trovare i blocchi di programmazione e come connetterli in sequenza ● lo studente sa muovere lo sprite ● lo studente sa come far dire qualcosa allo sprite Risultati specifici di apprendimento orientati al pensiero algoritmico: ● Creazione di una sequenza significativa di blocchi
Obiettivo, Compiti e Breve Descrizione delle Attività	Lo studente impara dove sono memorizzati i blocchi di programmazione, come trovare quelli appropriati, quali sono le categorie di blocchi e come collegarli in una sequenza.
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Dimostrazione dell'insegnante Lavoro individuale
Didattica	Lezione frontale Lavoro individuale
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) Alla fine dell'ora gli studenti saranno in grado di far muovere il proprio personaggio e fargli dire qualcosa. Si può mostrare loro un esempio di un programma che prepareranno in quest'ora.

[Fase 1]

Per prima cosa diamo un'occhiata ai blocchi di programmazione che sono disponibili per l'uso. Dove sono?

Sul lato sinistro, puoi trovare diverse categorie di blocchi: Movimento, Aspetto, Suoni, Penna, Controllo, Rilevamento, Operazioni e Variabili.

Per primo useremo il blocco 

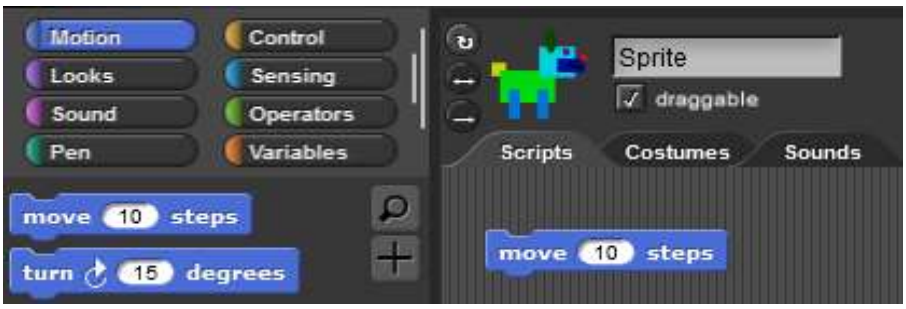
.

Compito per gli studenti: Prima trova il blocco e poi fai doppio clic su di esso. Che cosa è successo?

[Fase 2]

Per iniziare a connettere un blocco in un programma, è necessario trascinare e rilasciare i blocchi 

nella scheda “Scripts”.



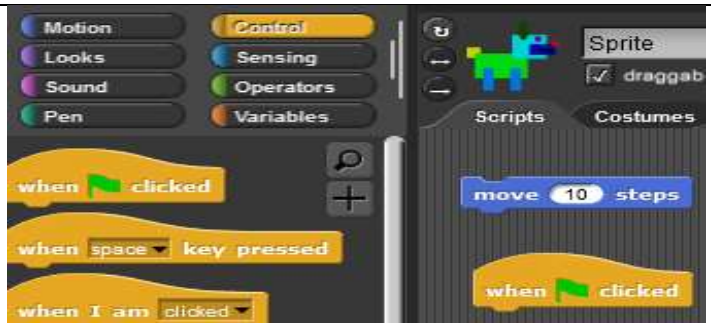
È possibile fare doppio clic sul blocco all'interno della scheda “Script” per eseguire il codice.

[Fase 3]

I programmi in Snap! si avviano di solito facendo clic sulla bandiera verde.

Compito per gli studenti: clicca sui diversi tipi di categorie e cerca di trovare un blocco che avvii il programma quando si clicca sulla bandiera verde.

Soluzione:



Se vuoi che il programma funzioni con una sequenza corretta di passaggi, i blocchi devono essere collegati come in un puzzle. Come questo:

Ora ogni volta che fai clic sulla bandiera verde, lo *sprite* si sposta per 10 passi, ma da una posizione diversa sull'immagine.

[Fase 4]

Se un blocco ha uno spazio bianco, significa che puoi cambiare i numeri o le lettere scritte lì.

Compito per gli studenti: Assicurati che il tuo personaggio si muova per 30 passi alla volta invece di solo 10.

[Fase 5]

Fai dire al tuo personaggio qualcosa. Dove vai a trovare il blocco "dire"? Cerca qual è la differenza tra 

e 

, spiegalo al tuo vicino.

[Fase 6]

Hai trovato entrambi i comandi dire nella categoria "Aspetto" (Looks).

La principale differenza è che con 

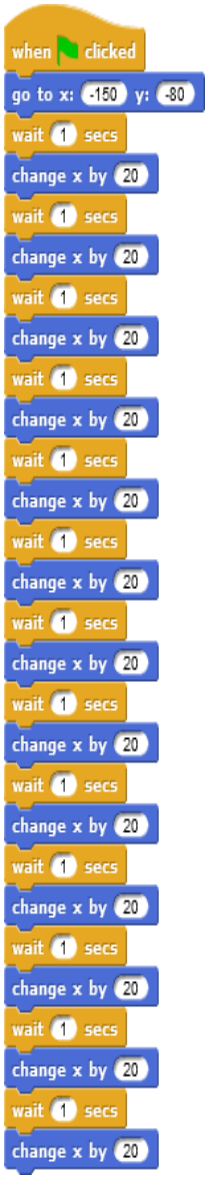
non si dice al programma di "aspettare n___ secondi" prima che il codice continui o smetta di

	dirlo in qualsiasi momento. [Fase 7] Prendi il tuo personaggio precedentemente creato. Trascinalo sullo *stage*, spostalo sul lato sinistro e scrivi un programma in modo tale che il personaggio si muova  dalla sua posizione sul lato sinistro a quello destro della scena. Dopo ogni movimento, il personaggio dovrebbe dire qualcosa. Inserisci più movimenti. **Prova -** Il personaggio finiva esattamente nella stessa posizione ogni volta che il tuo programma veniva eseguito? Riesci a trovare un blocco che assicuri che il tuo personaggio inizi sempre dalla stessa posizione e non esca dalla scena? **Suggerimento per l'insegnante**: se il personaggio esce dalla scena, puoi richiamarlo sul palco facendo clic su di esso con il tasto destro del mouse e scegliendo “mostra”. Il blocco che stai cercando è  . Al fine di determinare quali x e y vanno bene, puoi spostare il tuo personaggio nel punto in cui vuoi che si trovi e facendo clic sulla posizione x e la posizione y (nella parte inferiore della categoria dei blocchi Movimento / Motion) verrà mostrato l’attuale x e y. Devi solo scriverli negli spazi bianchi per fissarli. Riflessione e valutazione: Quante volte il tuo personaggio ha dovuto ripetere la mossa e pronunciare la sequenza per completare l'attività? Il numero è uguale per tutti nella classe? Perché?
Strumenti e risorse per gli insegnanti	**Esempio del programma:** https://snap.berkeley.edu/snap/snap.html#present:Username=spelac

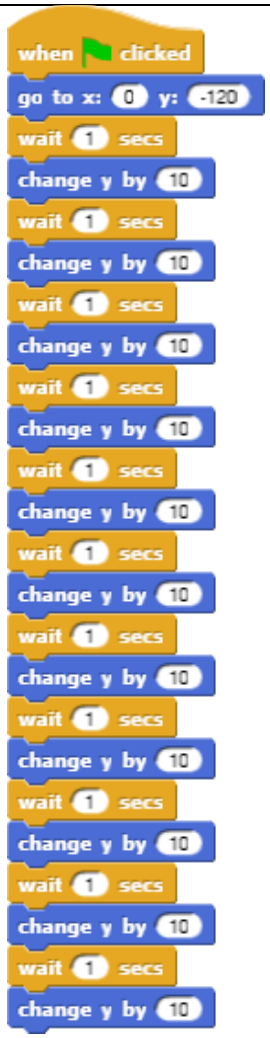
	&ProjectName=C4G_dog_goes_home
Risorse/materiali per gli studenti	• Istruzioni per lo studente (C4G2_InstructionsForStudent.docx) • Se lo studente non ha disegnato il proprio *sprite* e sfondo, si può usare: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_dog_goes_home_tmp

Scenario di apprendimento 3 - Muoversi sulla scena

Titolo dello scenario di apprendimento	Muoversi sulla scena
Precedente esperienza di programmazione	• Lo studente sa dove trovare i blocchi di programmazione e come connetterli in una sequenza
Risultati di apprendimento	Obiettivi generali di apprendimento: • realizzare una sequenza significativa di blocchi Obiettivi specifici di apprendimento orientati al pensiero algoritmico: • lo studente posiziona lo *sprite* sulla scena • lo studente cambia la posizione x e y dello *sprite* • lo studente sa ripetere il ciclo • lo studente impara che la direzione del movimento dello *sprite* in n.__ passi è relativo alla direzione verso cui è rivolto lo *sprite*
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione:** lo studente impara come muovere il proprio *sprite* nella direzione x e y sulla scena, programma un codice semplice per risolvere i compiti assegnati, impara come girare il suo *sprite* in una direzione diversa e in che modo ciò influisce sul blocco di n_ passi **Compiti:** sviluppare un programma che muove lo *sprite* nella direzione x e uno nella direzione y ed uno che combina i movimenti nelle direzioni x e y. **Obiettivo:** differenziare tra i movimenti nelle direzioni x e y sulla scena e ripetere il ciclo
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Dimostrazione dell'insegnante Lavoro individuale
Didattica	Lezione frontale Lavoro individuale

Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) Aiuterai diversi animali a raggiungere i loro obiettivi. Per fare ciò, dovrai dare loro le istruzioni su come muoversi sulla scena. [Fase 1] Apri il progetto "*Catch the ball*" e aggiungi il codice al cane in modo che possa catturare la palla. Usa i blocchi  e  per realizzare un'animazione per il cane che si muove verso la palla. Una possibile soluzione al compito: 

	Come puoi vedere, la x cambia quando ti sposti a sinistra o a destra. Se la x è 0, il tuo *sprite* è al centro del palco. Tutto ciò che è alla sinistra del centro, ha bisogno di "-" davanti al numero, e sarà più lontano dal centro se maggiore è il numero. A destra del centro, i valori x sono numeri maggiori di 0. Alla destra del centro, i valori delle x sono numeri più grandi di 0. Suggerimento: se l'esercizio è svolto con studenti più grandi, che conoscono i decimali, i tempi di attesa possono essere più brevi, ad es. 0.1. Se conoscono già cosa sono le coordinate, alcune spiegazioni possono essere omesse. [Fase 2] Apri il progetto "*Help monkey climb the tree*" (Aiuta la scimmia a arrampicarsi sull'albero) e aggiungi il codice alla scimmia per recuperare le banane. Usa i blocchi  e  per realizzare un'animazione per la scimmia che si arrampica sulla palma. Una possibile soluzione del compito:



Come puoi vedere, la y cambia quando ti sposti su o giù. Se la y è 0, il tuo *sprite* è al centro della scena. Tutto ciò che è più alto rispetto al centro ha la y maggiore di 0. Se tu vuoi che il tuo *sprite* sia al di sotto della linea centrale della scena, devi inserire il segno "-" di fronte al numero che definisci quanti passi sei al di sotto della linea centrale

Se vuoi scendere dall'albero, usa 

.

Suggerimento: Se l'esercizio è svolto con studenti più grandi, che conoscono già i decimali, i tempi di attesa possono essere più brevi, ad es. 0.1. Se conoscono il sistema cartesiano, alcune spiegazioni possono essere omesse.

[Fase 3]

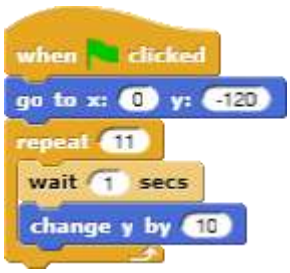
In entrambi le attività dovevi utilizzare in modo intercambiabile due blocchi. Quante volte hai dovuto **ripetere il codice**?

Esiste un modo più breve per scrivere questo codice dicendo al programma di ripetere il tuo codice un dato numero di volte. Questo è definito dal *loop* ossia "ripeti n(volte)__ il ciclo". Puoi usare questo blocco quando la stessa azione o una sequenza di azioni si ripete più volte. Prova a cambiare il codice per entrambe le attività in modo da poter utilizzare il ciclo 

. Il codice che vuoi ripetere deve essere inserito in questo blocco e devi specificare nello spazio vuoto quante volte deve essere ripetuto.

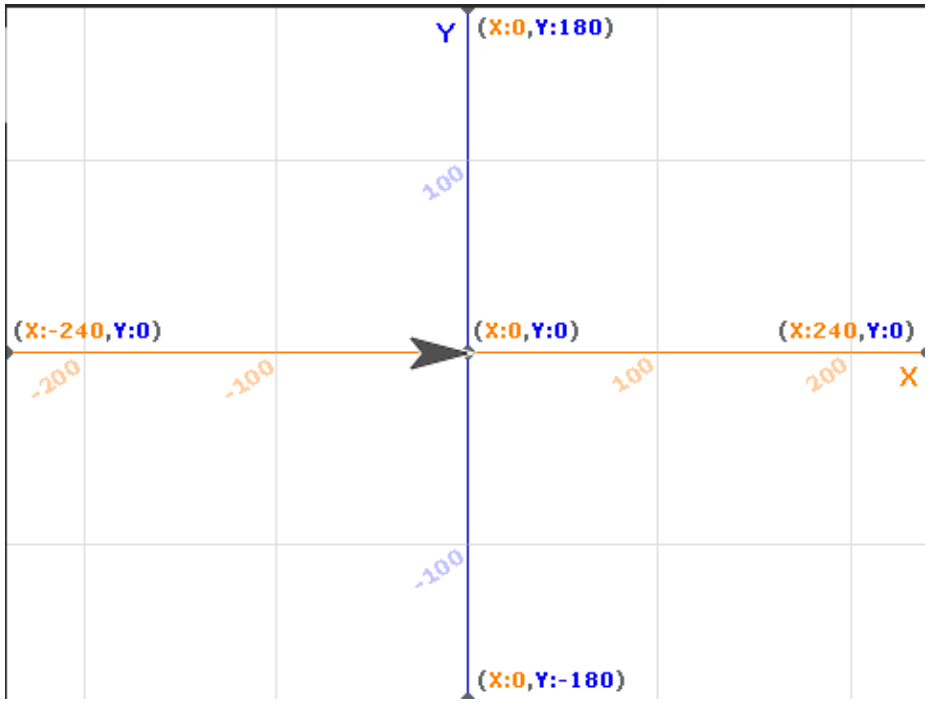
Codice per il cane:

Codice per la scimmia:



Compito: cerca di far correre il cane verso la palla e tornare indietro.

Compito: Cerca di far arrampicare la scimmia sull'albero e tornare indietro.

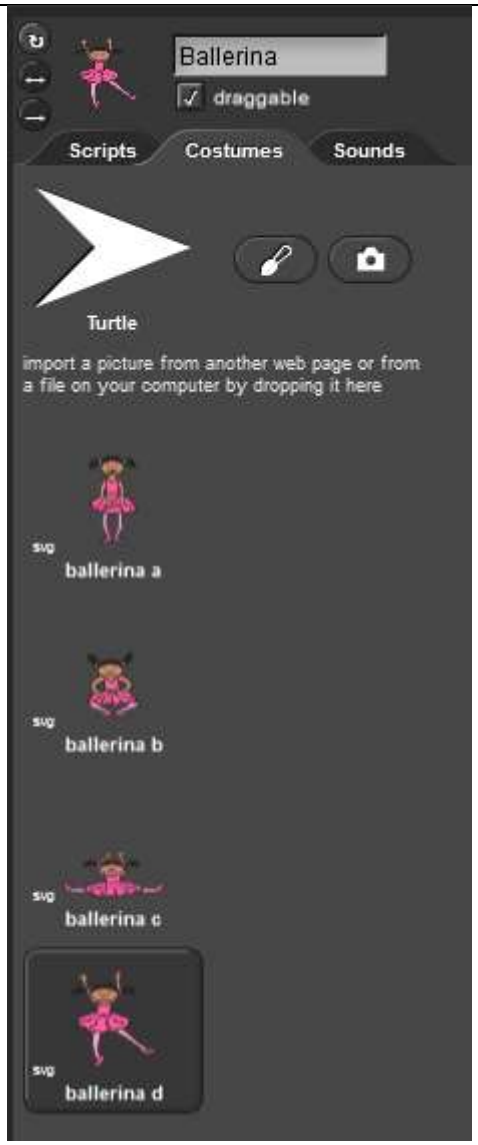
	Cosa ti è piaciuto di più? Puoi aiutarti con la posizione x e y dello *sprite* usando lo sfondo XY Grid in Snap: 
Strumenti e risorse per gli insegnanti	Una possibile soluzione per il progetto *Catch the ball* (Cattura la palla): https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_moving_x Una possibile soluzione per il progetto *Help monkey climb a tree* (Aiuta la sciamma ad arrampicarsi sull'albero): https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_moving_y
Risorse/materiali per gli studenti	• *Catch the ball*: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Catch_the_ball • *Help monkey climb the tree*: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_moving_y

	• Istruzioni per lo studente (C4G3_InstructionsForStudent.docx)

Scenario di apprendimento 4 - Cambiare costume e roteare

Titolo dello scenario di apprendimento	Cambiare costume e roteare
Precedente esperienza di programmazione	Movimento
Risultati di apprendimento	Obiettivi generali di apprendimento: ● realizzare una sequenza significativa di blocchi Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente cambia il costume dello *sprite* per creare un'animazione ● lo studente cambia la rotazione dei personaggi
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione:** lo studente impara come cambiare il costume dello *sprite* per fare un'animazione. Impara anche come cambiare tra i diversi tipi di rotazione dello *sprite*. **Compiti:** sviluppare un programma che cambi il costume dello *sprite*. Occorre impostare in ciascun programma il tipo di rotazione appropriato per ogni *sprite*. **Obiettivo:** sapere come cambiare il costume dello *sprite* e come impostare il tipo appropriato di rotazione.
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Dimostrazione dell'insegnante Lavoro individuale
Didattica	Lezione frontale Lavoro individuale

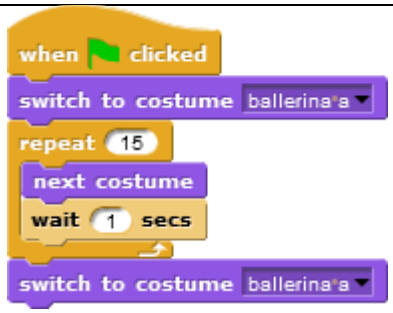
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) Imparerai a realizzare un'animazione per uno sprite in modo che sembri camminare, ballare ... [Fase 1] Apri un nuovo progetto vuoto, fai clic sull'icona che sembra un pezzo di carta bianca e seleziona Costumi "*Costumes*". Fai clic su *ballerina a* e fai clic su Importa "*Import*". Fai lo stesso con la *ballerina b*, la *ballerina c* e la *ballerina d*. Nella scheda Costumi "*Costumes Tab*" del tuo *sprite*, ora hai 4 costumi ballerina. Puoi rinominare lo *Sprite* con *Ballerina*, modificando il testo sopra la scheda Costumi.



Ora torna alla scheda "*Scripts tab*" e prova a creare un codice che inizierà quando si fa clic sulla bandiera verde e per 15 volte al secondo fai cambiare l'aspetto della *Ballerina*. Avrai bisogno del blocco. next costume.

Assicurati che la nostra ballerina inizi e finisca la sua danza con entrambe le gambe sul pavimento. La posizione iniziale e finale non fanno parte della sua danza.

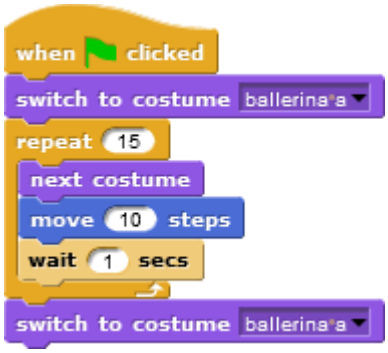
Soluzione:



[Fase 2]

La nostra ballerina non vuole essere sempre nella stessa posizione, quindi fa piccoli movimenti ogni volta che cambia costume. Aggiungi questo movimento alla sua danza.

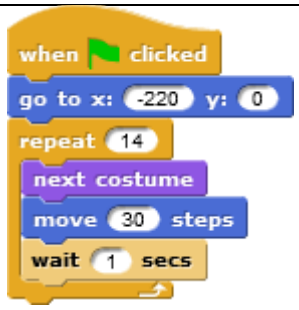
Una possibile soluzione:



[Fase 3]

Apri un nuovo progetto vuoto e importa i costumi di “Avery che cammina”. Aggiungi uno sfondo adatto per far camminare Avery. Crea un’animazione per Avery che cammina dal lato sinistro a quello destro della scena. Prova a capire come animare Avery in modo tale che i suoi passi sembrino collegati come nella vita reale.

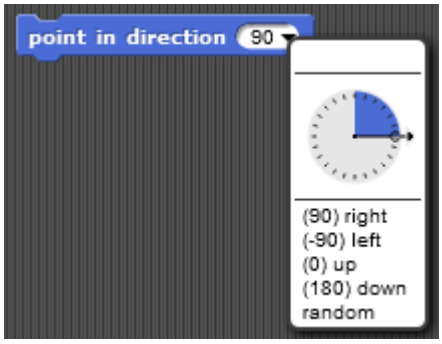
Una possibile soluzione:



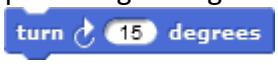
[Fase 4]

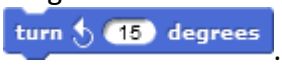
Fino ad ora, hai sempre scritto un programma in cui uno sprite si muoveva solo in una direzione.

In questa attività dovrai far ruotare il topo affinché raggiunga il formaggio. Per farlo ruotare, puoi scegliere:

a) 

dove dici in quale direzione deve guardare o

b) puoi dirgli di girare per un certo angolo in senso orario 

o antiorario 

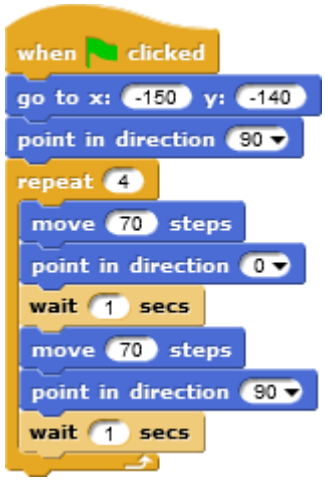
. Un giro completo è di 360 gradi, quindi, se vuoi farlo girare nella direzione opposta a quella in cui si trova ora devi farlo girare di 180 gradi. Se vuoi girare a sinistra, gira di 90 gradi in senso antiorario. Se vuoi girare a destra, gira di 90 gradi in senso orario.

Apri https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Find_cheese.

Scrivi un programma in cui il topo, per raggiungere il formaggio, deve camminare solo sull'area verde. Posiziona il topo nella direzione in cui si sta dirigendo e spostalo di n __ passi. Per vedere come il topo si

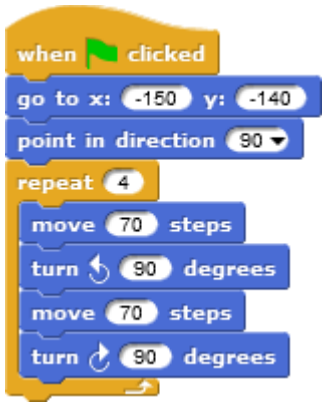
muove, usa “attendere 1 secondo” tra le due stringhe.

Soluzione:



Ora prova a scrivere un programma con una rotazione di 90 gradi.

Soluzione:



[Fase 5]

Come hai visto, il topo ha ruotato in diverse direzioni per raggiungere il formaggio. A volte non vuoi che il tuo *sprite* si capovolga, ma che giri solo a sinistra o a destra in modo da non camminare a testa in giù. Per assicurarti che il tuo *sprite* giri come tu desideri, devi cliccare sull'icona appropriata a sinistra del tuo *sprite*:

	La freccia circolare significa che il tuo *sprite* può girare in qualsiasi direzione (come il topo). La freccia <-> indica che il tuo *sprite* girerà solo a sinistra o a destra (questo è ciò che useresti per non camminare sulla testa). L'ultima freccia -> indica che lo *sprite* apparirà sempre così com'è (potresti usarlo per la scimmia). Prova a riscrivere i tuoi programmi per il cane e la scimmia in modo che prima vadano verso l'oggetto e poi tornino indietro. Assicurati di cambiare correttamente il loro stile di rotazione.
Strumenti e risorse per gli insegnanti	• Soluzioni per il programma Ballerina: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_dancing • Avery che cammina: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Avery_walking • Trovare il formaggio: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Find_cheese_solution
Risorse/materiali per gli studenti	• Trova il formaggio: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Find_cheese • Istruzioni per lo studente (C4G4_InstructionsForStudent.docx)

Scenario di apprendimento 5 - Suoni della fattoria

Titolo dello scenario di apprendimento	Suoni della fattoria
Precedente esperienza di programmazione	• Lo studente sa aggiungere uno sfondo. • Lo studente sa aggiungere un nuovo *sprite.* • Lo studente sa come far dire qualcosa allo *sprite.*
Risultati di apprendimento	Obiettivi generali di apprendimento: • aggiungi un suono dalla libreria multimediale di Snap! • importare suoni da altri media • registrare un nuovo suono • riprodurre il suono quando si preme un tasto Obiettivi specifici di apprendimento orientati al pensiero algoritmico: • lo studente aggiunge l'audio dalla libreria multimediale di Snap! e lo riproduce quando si preme un tasto • lo studente importa un suono dal computer e lo riproduce quando si preme un tasto • lo studente registra un nuovo suono e lo riproduce quando si preme un tasto
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: programmare un gioco semplice in cui il giocatore impara i suoni degli animali premendo determinati tasti. **Compiti**: nella prima fase lo studente deve scegliere lo sfondo della scena. Poi, lo studente deve programmare la contadina seguendo le istruzioni: 1) Se vuoi sentire il cane, clicca il tasto "D"; 2) Se vuoi sentire la mucca, clicca il tasto "C"; 3) Se vuoi sentire la pecora, clicca il tasto "S"; 4) se vuoi sentire il maiale, clicca sul tasto "P"; 5) Se vuoi sentire il cavallo, clicca sul tasto "H". Dopo questo, lo studente deve programmare l'attività della contadina. **Obiettivo**: Agli studenti verrà presentato come aggiungere un nuovo suono e come usarlo. Impareranno anche ad usare il blocco sonoro (“*riproduci il suono [nome_del_suono]*”) e il blocco di controllo (“*quando [il_tasto] il tasto è premuto*”).
Durata delle attività	45 minuti

Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento basato sulla progettazione di giochi
Didattica	Lezione frontale Lavoro individuale
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) **Motivazione-Introduzione** Motiviamo gli studenti giocando (non vedono il codice). L'obiettivo della lezione è rendere il gioco così:  [Fase 1] Nella prima fase occorre determinare lo sfondo del gioco. Lo sfondo deve contenere diversi animali. Abbiamo tre opzioni: 1. gli studenti disegnano loro stessi lo sfondo; 2. gli studenti ricercano immagini gratuite online; 3. forniamo gli sfondi agli studenti (se vogliamo risparmiare tempo). Gli studenti già sanno come aggiungere lo sfondo così procedono individualmente. 

[Fase 2]

Nella seconda fase occorre aggiungere la contadina. Abbiamo le stesse opzioni della prima fase:

1. gli studenti disegnano loro stessi la contadina;
2. gli studenti cercano le immagini gratuite della contadina online;
3. forniamo le immagini della contadina agli studenti (se vogliamo risparmiare tempo).

Gli studenti già sanno come aggiungere un nuovo *sprite*, così possono farlo individualmente.

[Fase 3]

Gli studenti devono programmare le istruzioni per il giocatore. Le istruzioni sono fornite dalla contadina. Gli studenti lo fanno usando i blocchi *Looks/say[string]* e *wait[n]*. Gli studenti già sanno come farlo così possono procedre individualmente.

Successivamente mostriamo agli studenti come aggiungere suono nel gioco. Abbiamo tre opzioni:

1. importare un suono dalla libreria multimediale di Snap!;
2. importare un suono dal nostro computer trascinandolo in Snap!;
3. registrare un nuovo suono in Snap!.

Mostriamo agli studenti tutte e tre le opzioni sotto forma di insegnamento frontale. Successivamente iniziano a realizzare le

attività individualmente (con il supporto dell'insegnante).

[Fase 4]

Gli studenti devono programmare il suono del cane. Quando il giocatore preme il tasto“D”, il cane deve abbaiare. Per prima cosa, gli studenti importano il suono dalla libreria multimediale di Snap! nella scheda dei suoni dello sfondo.

Successivamente, scelgono il suono del cane (*Dog* 1 o *Dog* 2).

Gli studenti devono far riprodurre il suono del cane quando il tasto “D” viene premuto. Devono farlo usando il blocco *Control/quando [il_tasto] tasto è premuto* ed il blocco *Sound/registra suono [nome_del_suono]*.

[Fase 5]

Gli studenti devono scrivere il programma per riprodurre i suoni degli animali. Per prima cosa devono aggiungere i suoni dal loro computer. Lo fanno trascinando i suoni nella scheda dei suoni dello sfondo.

Una volta che hanno importato i suoni, possono cliccare con il tasto destro del mouse sui suoni per rinominarli. Nel nostro caso sono chiamati mucca, maiale, cavallo e pecora.

Successivamente gli studenti devono aggiungere il suono nella sceda script dello sfondo. Lo fanno usando il blocco *Control/quando[il_tasto] tasto è premuto* e *Sound/registra suono[nome_del_suono]*.

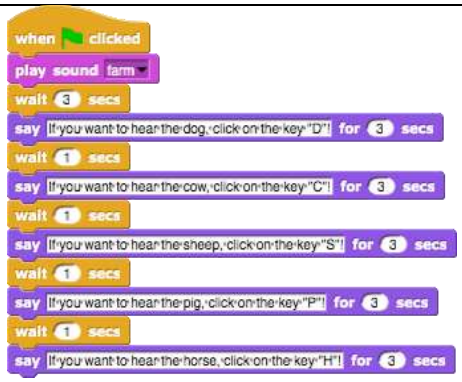
[Fase 6]

Il prossimo passo è programmare il saluto di benvenuto della contadina. Quando il giocatore inizia il gioco la contadina deve dire: “Benvenuti nella mia fattoria”. Dapprima gli studenti devono

registrare il saluto di benvenuto della contadina. Lo fanno con il registratore di suoni (pulsante rosso) situato nella scheda Suoni (della contadina) "*Sounds tab*". Quando registrano il suono devono salvarlo "*Save button*".

Una volta salvato l'audio, possono cliccare con il pulsante destro del mouse per rinominarlo. Nel nostro caso si chiama fattoria.

Ora gli studenti devono aggiungere l'audio negli *scripts* della contadina. Lo fanno usando il blocco *Sound/registra suono[nome_del_suono]*.



Compito aggiuntivo

Gli studenti possono migliorare la fattoria a loro piacimento aggiungendo nuovi sprite (contadino, gallina, trattore, ...) e suoni.

Riflessione e valutazione

Gli studenti riepilogano:

- come hanno aggiunto i suoni nel loro codice;
- quali blocchi hanno usato per inserire i suoni nel codice;
- quali blocchi di controllo hanno usato nel loro codice;
- perché e come hanno usato i blocchi dei suoni e i blocchi di controllo.

[Codice finale]

La contadina

Lo sfondo

	
Strumenti e risorse per gli insegnanti	• Attività completa in Snap!: https://snap.berkeley.edu/project?user=tadeja&project=Farm • Sito web delle immagini gratuite: https://pixabay.com/ • Sito web per i suoni gratuiti: https://www.zapsplat.com/ • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Risorse/materiali per gli studenti	• Modello in Snap!: https://snap.berkeley.edu/project?user=tadeja&project=Sounds%20of%20the%20farm_0 • Sito web delle immagini gratuite: https://pixabay.com/ • Sito web per i suoni gratuiti: https://www.zapsplat.com/ • Istruzioni per lo studente (C4G5_InstructionsForStudent.docx)

Scenario di apprendimento 6 – Le vacanze estive di un Camaleonte

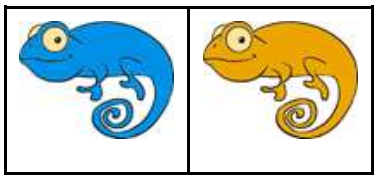
Titolo dello scenario di apprendimento	Le vacanze estive di Camaleonte
Precedente esperienza di programmazione	Non è richiesta nessuna conoscenza precedente
Risultati di apprendimento	Obiettivi generali di apprendimento: ● evento basato sul movimento degli oggetti ● rilevamento del colore singolo o multiplo ● letture di valori *Boolean* nelle espressioni logiche ● definire, differenziare, controllare dinamicamente e rispondere ai diversi stati del gioco Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente implementa il movimento degli oggetti con i tasti freccia utilizzando gli eventi e tenendo conto delle restrizioni, ● lo studente usa il blocco rilevamento colore per ottenere il valore *boolean* per la lettura di rilevamento del colore singolo o multiplo ● lo studente realizza lo stato dell'oggetto che può essere espresso con i colori che l'oggetto sta toccando ● lo studente differenzia fra cinque stati e sa come manifestarli con espressioni logiche ● lo studente realizza che la posizione degli oggetti sta cambiando in modo dinamico e utilizza il *loop* per sempre per controllare ripetutamente lo stato corrente ● lo studente usa le *if sentence* per dare risposte diverse in base alla posizione corrente dell'oggetto
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: programmare un gioco semplice in cui l'oggetto cambi il suo costume in base al colore dello sfondo. **Compiti**: gli studenti devono programmare il camaleonte in modo da cambiare il suo aspetto "*looks*" (costume) e dire dove si trova rispetto a cinque diverse situazioni: 1) quando nuota nel mare, deve cambiare il suo colore in blu e dire "Sto nuotando nel mare", 2) quando si trova

	tra il mare e la spiaggia, la sua pelle diventa blue color sabbia e dice "Sono tra il mare e la spiaggia", 3) sulla spiaggia, diventa color sabbia e dice "Mi sto rilassando in spiaggia", 4) tra la spiaggia e la foresta, diventa metà verde e l'altra metà color sabbia e dice "Sono tra la spiaggia e la foresta", 5) nella foresta, la sua pelle diventa verde e dice "Mi sto rinfrescando all'ombra dell'albero". **Agli studenti verrà illustrato il blocco di rilevamento del colore e gli verrà dimostrato come usarlo nelle espressioni logiche per differenziare tra gli stati di gioco che cambiano dinamicamente e il fornire le risposte giuste**
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento collaborativo, *problem solving*
Didattica	Lezione frontale Lavoro individuale/Lavoro in coppia/lavoro di gruppo
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) Il Camaleonte è andato in vacanza. Gli piace fare il bagno in mare, godersi il relax in spiaggia e quando fa troppo caldo gli piace andare al riparo sotto gli alberi vicini per rinfrescarsi. Poiché è un Camaleonte, cambia colore in base allo sfondo. **[Versione base]** Nella versione base dobbiamo diversificare i due stati. [Fase 1] Chiediamo agli studenti di modificare lo sfondo della scena in modo che sia diviso in due parti dello stesso colore, blu e sabbia, ognuna delle quali rappresenta un luogo diverso. Il colore blu è per il mare e

color sabbia per la spiaggia. Possiamo istruire gli studenti ad includere altri elementi per rendere lo sfondo più realistico, come ad esempio: onde, conchiglie, castelli di sabbia, ombrelloni, ecc. Devono stare attenti a non scegliere oggetti più grandi e interamente colorati con colori diversi rispetto a sfondo. In tal caso, il blocco di rilevamento del colore non sarà in grado di riconoscere in quale parte della scena si trova il personaggio.

[Fase 2]

Devono disegnare un Camaleonte e dipingere la sua pelle in due diverse combinazioni che rappresentano la sua posizione sulla scena:



[Fase 3]

Per prima cosa devono far muovere il camaleonte in quattro direzioni usando i tasti. Possono scegliere la propria combinazione di tasti (ad es. Tasti freccia o WASD).

Dobbiamo avvertire gli studenti di non dimenticare che quel personaggio può uscire dalla scena se non usiamo il blocco appropriato durante la programmazione del movimento, ossia rimbalza se sul bordo (*bounce if on edge block*).

Per rendere il movimento del Camaleonte un po' più realistico, vogliamo che giri a sinistra o a destra per muoversi orizzontalmente

(usando il blocco *punta nella direzione*).

[Fase 4]

Introduciamo agli studenti il concetto dello *sprite* che percepisce il colore (i colori) che sta toccando. Con il blocco "toccare il colore" (*touching color*) possiamo ottenere informazioni in una forma di valori booleani – Vero o Falso se i un dato momento sta toccando colori singoli o multipli. Poiché otteniamo un valore booleano da questo blocco possiamo usarlo all'inizio del blocco "*If sentence*" dove viene deciso se eseguiremo i comandi elencati nel suo codice o meno.

Successivamente discuteremo con gli studenti delle diverse posizioni del Camaleonte sulla scena e come possiamo realizzarle usando il blocco "*touching color* ".

Scopriamo rapidamente che ce ne sono due:

1. È interamente sulla parte blu -> Toccando il colore [blu]
2. È interamente sulla parte sabbia -> Toccando il colore [sabbia]

Quando tocca determinati colori, dobbiamo cambiarne l'aspetto e fargli dire dove si trova. Possiamo variare l'aspetto dello *sprite* cambiando tra i suoi costumi. Questo viene fatto con *Aspetto/passa al blocco costume[opzione] "Looks/switch to costume[option]"* dove selezioniamo uno dei possibili costumi che vogliamo mostrare. Per far parlare il Camaleonte usiamo il blocco Aspetto/dire/testo "*Looks/say[text]*".

Poiché ci sono solo due possibilità, possiamo usare il blocco

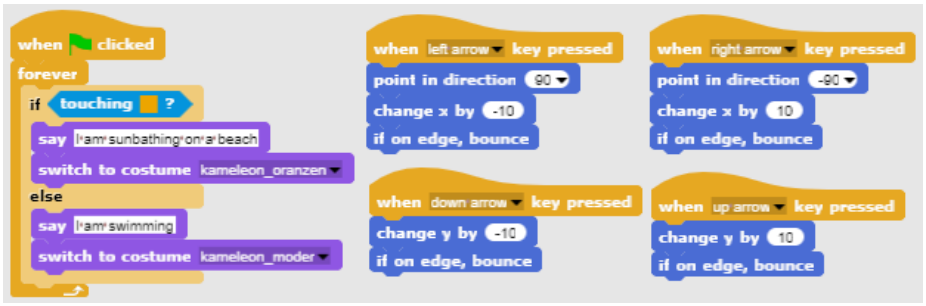
condizionale "if - else".

Possiamo scegliere quale colore controllare e implicitamente l'altro colore cadrà nel caso “else”. Nel codice di esempio abbiamo scelto il colore sabbia:

[Fase 5]

Per le situazioni in cui dobbiamo eseguire determinati comandi per l'intera durata del programma che utilizziamo il ciclo per sempre. Tutto ciò che è scritto all’interno del ciclo per sempre verrà eseguito più e più volte. Discutiamo con gli studenti che nel nostro caso questo è esattamente ciò che vogliamo / ed quello di cui abbiamo bisogno per creare questo gioco.

[Codice Finale]



[Versione completa]

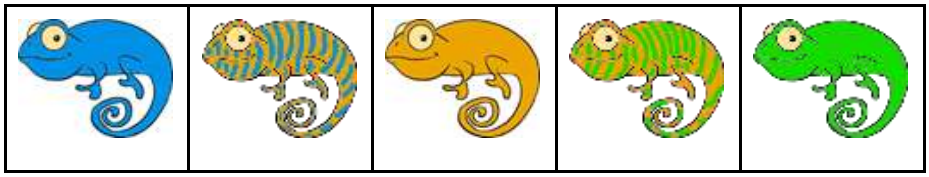
[Fase 1]

Chiediamo agli studenti di modificare lo sfondo della scena in modo che sia diviso in tre parti dello stesso colore, ognuna rappresenta un

posto diverso: il colore blu è per il mare, il colore sabbia per la spiaggia e il verde per la foresta. Possono aggiungere altri elementi per rendere più realistico uno sfondo come: onde, conchiglie, castelli di sabbia, ombrelloni, alberi, ecc ... ma devono fare attenzione che gli elementi aggiunti non siano più grandi del personaggio principale, perché in questo caso il personaggio non toccherà nessuno dei tre colori e la funzione di rilevamento di Snap! non sarà in grado di riconoscere in quale parte della scena si trova il personaggio.

[Fase 2]

Devono disegnare un Camaleonte e dipingere la sua pelle in cinque diverse combinazioni che rappresentano la sua posizione sulla scena:



[Fase 3]

Per prima cosa devono far muovere il Camaleonte in quattro direzioni usando i tasti. Possono scegliere la propria combinazione di tasti (ad es. Tasti freccia o WASD). A questo punto supponiamo che sappiano come farlo. Dobbiamo avvertire gli studenti di non dimenticare che il personaggio può uscire dalla scena se non usiamo il blocco appropriato durante la programmazione del movimento (rimbalzo se sul bordo).

Per rendere il movimento del Camaleonte un po' più realistico,

vogliamo che giri a sinistra o a destra per affrontare la direzione orizzontale che stiamo affrontando (usando un blocco punto in direzione).

[Fase 4]

Presentiamo agli studenti il concetto di un personaggio che rileva il colore (colori) che sta toccando. Con il blocco "toccare il colore" possiamo ottenere informazioni in una forma di valori booleani - Vero o Falso se al momento sta toccando colori singoli o multipli. Poiché otteniamo un valore booleano da questo blocco, possiamo usarlo nella testa della frase If, dove viene deciso se eseguiremo i comandi elencati nel suo interno o meno.

Successivamente discuteremo con gli studenti quali sono le diverse posizioni del Camaleonte sulla scena e come possiamo esprimerle usando il blocco colore che tocca (touching colour).

Scopriamo rapidamente che ce ne sono cinque:

1. È interamente sulla parte blu -> Toccando il colore [blu]
2. È tra la parte blu e quella sabbia -> Toccando il colore [blu] E (*AND*) toccando il colore [sabbia]
3. È interamente sulla parte sabbia -> Toccando il colore [sabbia]
4. È tra la parte sabbia e quella verde -> Toccando il colore [sabbia] E (*AND*) Toccando il colore [verde]
5. È interamente sulla parte verde -> Toccando il colore [verde]

Quando tocca determinati colori, dobbiamo cambiarne l'aspetto e fargli dire dove si trova. Possiamo variare l'aspetto dello *sprite* cambiando tra i suoi costumi. Questo viene fatto con *Aspetto/passa al blocco costume[opzione] "Looks/switch to costume[option]"* dove selezioniamo uno dei possibili costumi che vogliamo mostrare. Per far

parlare il Camaleonte usiamo il blocco Aspetto/dire[testo "*Looks/say[text]*".

Innanzitutto ci occupiamo delle situazioni più semplici in cui il Camaleonte è interamente sullo stesso colore della scena:

Successivamente formiamo un'espressione logica con l'uso dell'operatore logico AND, perché vogliamo verificare se il Camaleonte sta toccando due colori contemporaneamente:

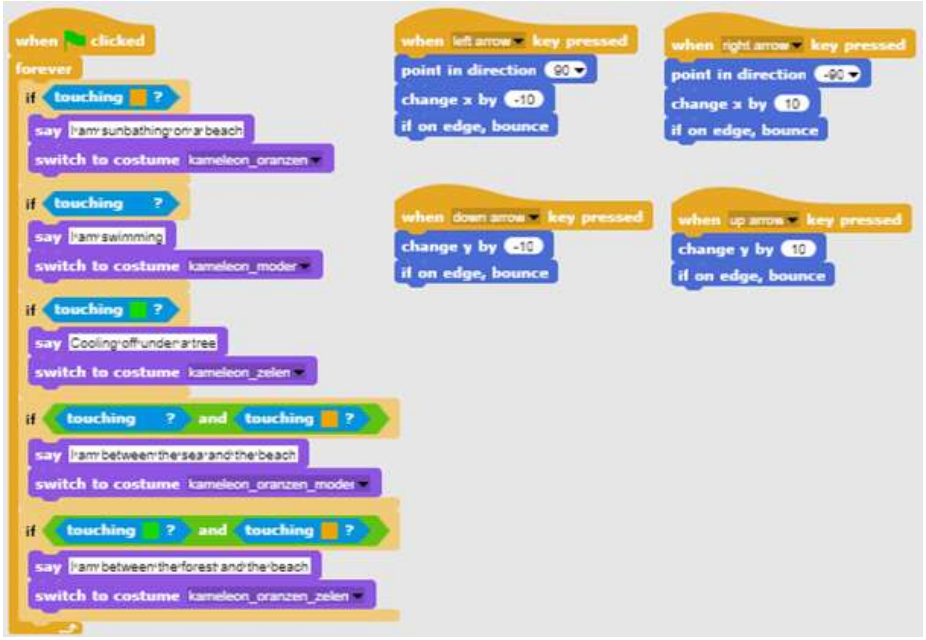
Se uniamo le frasi condizionali sopra riportate e le inseriamo in un blocco evento "clic su bandiera verde", notiamo che queste condizioni verranno verificate esattamente una volta. Li aiutiamo a notare che, poiché controlliamo il movimento del personaggio principale, la posizione del Camaleonte cambierà continuamente durante il gioco. Questo è il motivo per cui dobbiamo controllare costantemente tali condizioni non solo una volta, ma per tutto il tempo!

[Fase 5]

Per situazioni in cui dobbiamo eseguire determinati comandi per l'intera esecuzione del programma che utilizziamo,- il ciclo per sempre. Tutto ciò che è scritto sll'interno del ciclo per sempre verrà eseguito più e più volte. Discutiamo con gli studenti che nel nostro caso questo è esattamente ciò che vogliamo / e quello di cui abbiamo

bisogno per creare questo gioco.

[Codice finale]



[Gli studenti aggiustano il codice]

Per semplificare questa attività, possiamo preparare in anticipo parte del codice in un file modello ed istruire gli studenti a completarlo.

Gli studenti che hanno seguito il percorso suggerito hanno già imparato a spostare l'oggetto con le chiavi. Quindi possiamo includere il codice di movimento in un file modello. Possono modificare le impostazioni dei tasti dai tasti freccia per una disposizione personalizzata (ad es., WASD).

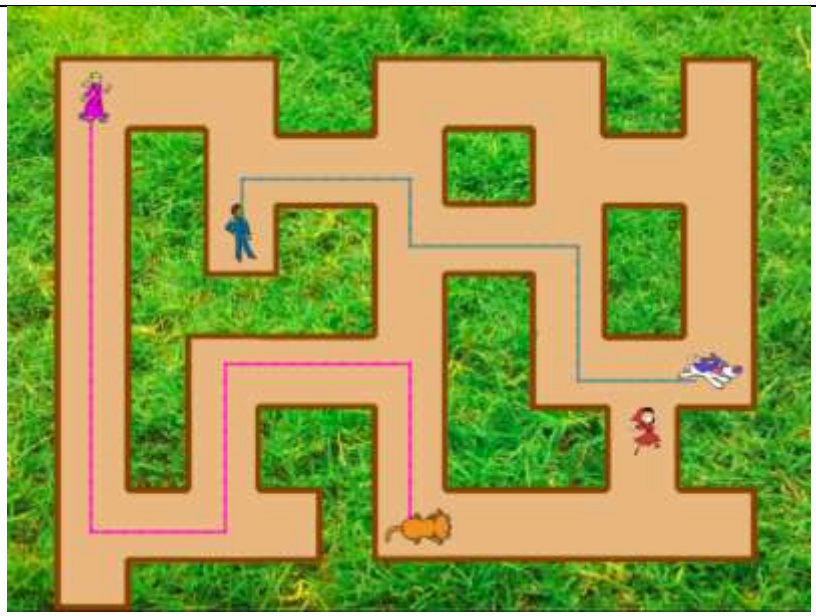
Per aiutarli a comprendere il concetto di ciclo continuo e come usarlo per rilevare il colore di sfondo, possiamo includere un codice per rilevare due situazioni: 1) l'oggetto è interamente su un colore, 2)

	l'oggetto tocca due colori contemporaneamente. Incarichiamoli di completare il codice per ogni caso. Modello di codice suggerito: 
Strumenti e risorse per gli insegnanti	• Attività Completa in Snap!: Base: https://snap.berkeley.edu/project?user=zapusek&project=chameleon_simple Completa: https://snap.berkeley.edu/project?user=zapusek&project=chameleon • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Risorse/materiali per gli studenti	• Modello in Snap!: https://snap.berkeley.edu/project?user=zapusek&project=chameleon_template • Metà attività in Snap!: https://snap.berkeley.edu/project?user=zapusek&project=chameleon_half_baked • Istruzioni per lo studente (C4G6_InstructionsForStudent.docx)

Scenario di apprendimento 7 - Aiutare il principe e principessa a trovare i loro animali

Titolo dello scenario di apprendimento	Aiutare il principe e principessa a trovare i loro animali
Precedente esperienza di programmazione	Aggiungere il testo per lo *sprite* Muovere gli oggetti con i tasti freccia utilizzando gli eventi Usare il condizionale per l'*oggetto che sta toccando* per definire lo stato dell'oggetto Usare gli eventi
Risultati di apprendimento	Obiettivi generali di apprendimento: ● i condizionali per *l'oggetto che sta toccando* un determinato colore ● muoversi con le coordinate ● penna su, penna giù ● colore della penna Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente usa *if sentence* per lo stato dell'oggetto e tornare indietro se si tocca un determinato colore ● lo studente imposta le coordinate iniziali x e y per lo *sprite* ● lo studente usa la penna su e la penna giù per disegnare una linea/un percorso ● lo studente cambia il colore della penna a seconda della coppia che sta collegando ● lo studente si rende conto che all'inizio deve cancellare tutti i percorsi precedenti
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: gli studenti, tramite il loro personaggio, devono aiutare la principessa a trovare il suo gatto e il principe il suo cane. Vanno dalla principessa e le mostrano, disegnando una linea/tratto, la strada per arrivare al gatto. Allo stesso modo mostrano al principe la strada per giungere al suo cane. Il personaggio deve evitare l'incontro tra gli animali in modo che i loro percorsi non possano incrociarsi. **Compiti:** In un primo momento gli studenti devono scegliere lo sfondo appropriato (un labirinto). Aggiungono cinque *sprite* nel labirinto: il loro *sprite* (una ragazza), una principessa, un principe, un gatto e un cane.

	Successivamente programmano i movimenti della ragazza con le chiavi (usando gli eventi), prestando attenzione affinché lo *sprite* non cammini sull'erba. Dopo programmano il disegno con una penna e cambiano il colore della penna con gli eventi. Devono anche programmare l'evento di partenza, che cancella il percorso e la ragazza dà le istruzioni. **Obiettivo: Agli studenti verrà illustrato il disegno con movimenti chiave. Inoltre impareranno come usare i condizionali per impedire allo *sprite* di muoversi su tutto lo schermo.**
Durata delle attività	30 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento basato sulla programmazione del gioco, *problem solving*
Didattica	Lezione frontale Lavoro individuale
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) Inizialmente viene dato agli studenti: ● sfondo ● lo *sprite* ragazza ● codice di movimento per una direzione Lo *sprite* ragazza decide di aiutare la principessa a trovare il suo gatto e il principe il suo cane mostrando (disegnando) il percorso verso i loro animali. Per evitare confusione, i percorsi dovrebbero essere di colori diversi e non dovrebbero incrociarsi.



[Fase 1]

Chiediamo agli studenti di modificare lo sfondo della scena: un labirinto. Per implementare se si tocca il colore "*if touching color*", lo sfondo (erba) deve essere monocromatico o il percorso deve avere una cornice monocromatica, come nel nostro caso. Per evitare problemi nel trovare uno sfondo adeguato, diamo loro questo sfondo.

[Fase 2]

Gli studenti muovono lo *sprite* ragazza. Devono trovare altri quattro *sprite* e metterli nel labirinto. Per tutti gli sprite devono impostare la dimensione appropriata (che è più piccola della larghezza dei percorsi nel labirinto). Per ogni sprire utilizzano il seguente codice:

Si raccomanda per la ragazza una dimensione pari all'8%, mentre gli altri sprite possono essere più grandi.

[Fase 3]

Successivamente devono muovere la ragazza nelle quattro direzioni

usando i tasti. Partiamo dal presupposto che sanno già come farlo. Ad ogni modo, diamo loro il codice per una direzione in modo tale che li possa aiuta a sviluppare gli altri tre.

[Fase 4]

Nel passaggio successivo devono impedire il movimento della ragazza attraverso il prato. Lo fanno aggiungendo un blocco condizionale se si tocca il colore marrone. Se la ragazza tocca il colore marrone (fine del percorso), si sposta di 10 passi indietro. Non vediamo questi due passaggi ed è come se la ragazza fosse nella stessa posizione. Questo è un codice per spostarsi a destra, quindi 10 passi indietro significa cambiare x di -10.

Aggiungono questo codice sotto il codice precedente, ad es. per la freccia destra:

Lo stesso deve essere fatto per altre tre direzioni.

[Fase 5]

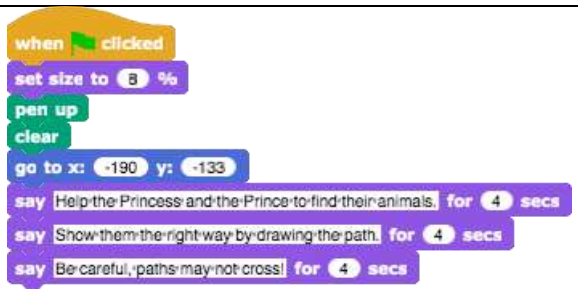
Quindi programmano il disegno. Lo fanno con i blocchi penna su "*pen up*" e penna giù "*pen down*" usando gli eventi *quando si preme il tasto*.

Quando il tasto "D" è premuto e la ragazza si muove, disegna una linea.
Quando il tasto "E" è premuto, il disegno si ferma.

Allo stesso modo si imposta il colore della penna premendo il tasto:

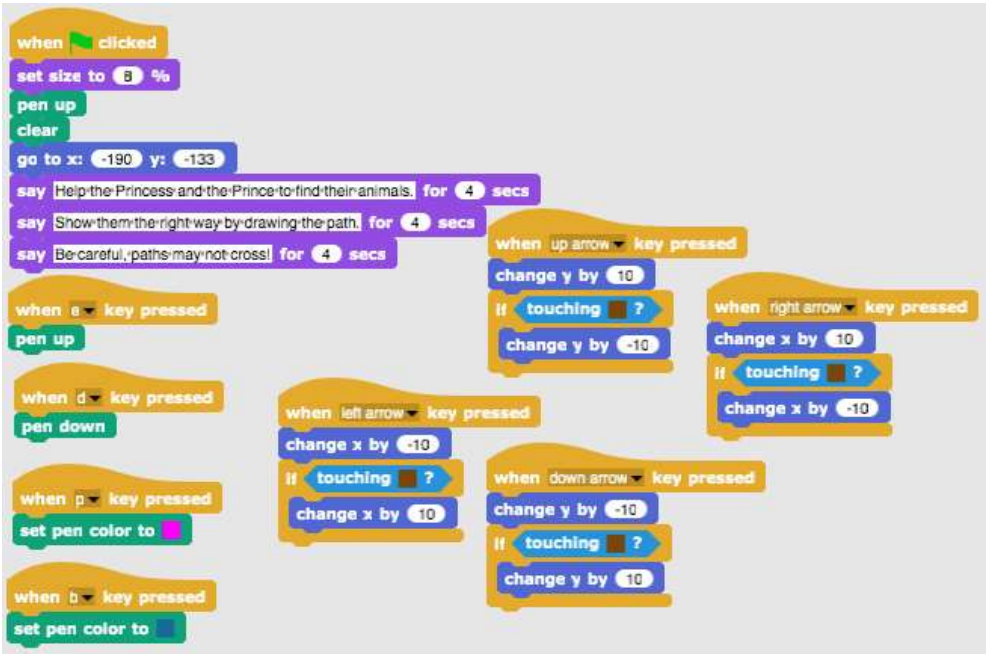
[Fase 6]

Alla fine si programma quando cliccare sull'evento bandiera verde, dove gli studenti aggiungono alcune istruzioni che la ragazza dice all'inizio.
Quando si gioca è bene fermare il gioco e riprenderlo di nuovo, gli studenti vedranno che è utile aggiungere i seguenti blocchi: *penna su* (nel caso in cui fosse rimasto basso dal gioco precedente), *pulisci* (cancella il percorso del gioco precedente) e *vai a x, y* (la ragazza inizia sempre da queste coordinate, che sono all'interno del percorso e non sull'erba).
Per determinare le coordinate iniziali per la ragazza, teniamo premuto il tasto del mouse sulla ragazza e la lasciamo dove vogliamo che inizi. Dopo clicchiamo su blocchi Movimento "*Motion*" dove possiamo trovare la *posizione x* e la *posizione y*. Cliccando sulla posizione x scopriamo la posizione x della ragazza, stessa cosa per la posizione y.



[Codice finale]

Ragazza



Es. Principessa

[Compiti aggiuntivi]

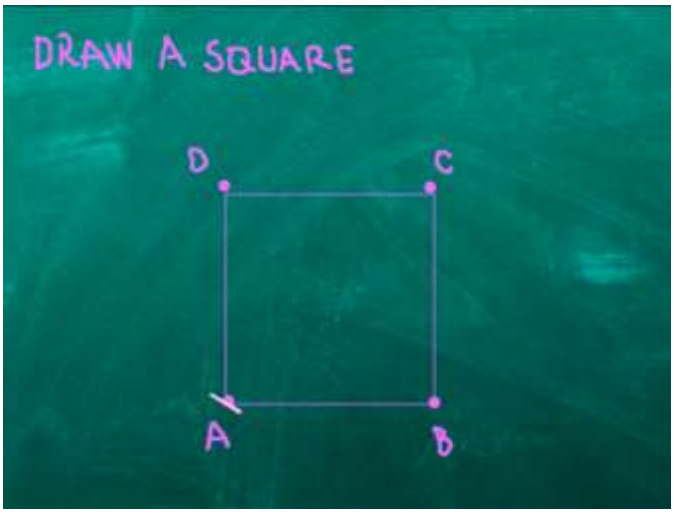
Gli studenti possono aggiungere attività in base ai loro desideri o possono seguire quelle descritte di seguito:

- Imposta le coordinate iniziali per il Principe e la Principessa e scrivi un codice per il loro movimento. Impostare la loro dimensione appropriata. Dovrebbero tracciare un percorso verso i loro animali.
- Aggiungi un altro sprite (animale) per la ragazza.

	• Ogni sprite deve disegnare con un colore diverso. • Sistemare le istruzioni iniziali. • Aggiungi le istruzioni per spostare uno *sprite* e disegnare facendo clic su di esso. Es. dice la Principessa: "Mi muovi premendo i tasti W, S, A e D. Traccio il percorso premendo il tasto 3. Smetto di disegnare premendo il tasto 4. Aiutami a trovare il mio gatto!"
Strumenti e risorse per gli insegnanti	• Attività completa in Snap!: https://snap.berkeley.edu/project?user=mateja&project=Helping%20Prince%20and%20Princess%20to%20find%20their%20animals • Attività in Snap! con compiti aggiuntivi (possibile soluzione): https://snap.berkeley.edu/project?user=mateja&project=Helping%20Prince%20and%20Princess%20to%20find%20their%20animals%20%2B%20Add.%20Task • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Risorse/materiali per gli studenti	• Metà attività in Snap!: https://snap.berkeley.edu/project?user=mateja&project=Helping%20Prince%20and%20Princess%20to%20find%20their%20animals%20-%20Part • Istruzioni per lo studente (C4G7_InstructionsForStudent.docx)

Scenario di apprendimento 8 - Disegnare con il gesso

Titolo dello scenario di apprendimento	Disegnare con il gesso
Precedente esperienza di programmazione	Aggiungere il testo allo *sprite* Disegnare con la penna (penna su, penna giù, impostazione dei colori) Muoversi con i passi Usare i cicli (*loops*) Usare gli eventi
Risultati di apprendimento	Obiettivi generali di apprendimento: ● ripetizione del ciclo ● girare di 90 gradi ● punto di direzione ● cambiare lo sfondo Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente sa ripetere il ciclo quando gli stessi blocchi si ripetono 2/4 volte ● lo studente sa girare di 90 gradi quando si disegnano forme diverse (quadrato, triangolo, lettera "T") ● lo studente comprende il significato del punto di direzione 90 ● lo studente sa come modificare lo sfondo con la combinazione di un evento quando il tasto è premuto
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: il giocatore realizza tre sfondi diversi e deve collegare i punti in tre forme diverse: un quadrato, un rettangolo e una lettera "T". **Compiti**: Gli studenti scelgono lo sfondo e iniziano a disegnare un quadrato. La loro posizione iniziale è il punto “A”. Quando disegnano un quadrato, ripetono alcuni passaggi 4 volte, quindi invece di scrivere lo stesso codice 4 volte, possono usare una ripetizione in un ciclo (*loop*) 4 volte. Poi disegnano un rettangolo, anche usando una ripetizione ciclica, questa volta ripetendola 2 volte. Nel loro ultimo compito devono collegare i punti a forma di lettera "T", devono scoprire il numero dei passi. Possono usare il *loop repeat* dove è possibile. **Obiettivo: agli studenti verrà mostrato come disegnare forme diverse con un codice. Impareranno ad usare la ripetizione ciclica “*loop repeat*”**

	per abbreviare il codice e cambiare uno sfondo.
Durata delle attività	60 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento basato sulla programmazione del gioco, *problem solving*
Didattica	Lezione frontale Lavoro individuale / Lavoro in coppia
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) Inizialmente viene dato agli studenti: ● tre sfondi con tutti i punti che devono connettere ● lo *sprite* del gesso Il gesso vuole disegnare un quadrato, un rettangolo e collegare punti a forma di lettera "T" ma non sa come muoversi e come girare. Scrivi un codice e mostra al gesso come farlo! [Fase 1]  Gli studenti iniziano con questo sfondo. Scrivono un codice per disegnare un quadrato. A partire dal punto "A", spostano X passi verso il punto "B", ruotano di 90 gradi a sinistra, spostano X passi verso il punto "C", ruotano di 90 gradi a sinistra, spostano X passi verso il punto "D ", ruota di 90 gradi a sinistra, spostano X passi verso il punto" A "(e ruota di 90 gradi a sinistra).

Usare la rotazione di 90 gradi "*turn 90 degrees*" è il modo più semplice, poiché possiamo sempre usare la rotazione di 90 gradi (dipende solo se vogliamo girare a sinistra o a destra). Un'altra opzione è usare il punto di direzione "*point in direction"* 0, 90, 180, -90, ma è un po' più complicata perché dobbiamo separare 4 possibilità e possiamo non usare il *loop repeat*.

Il blocco aspetta 1 secondo *"Wait 1 secs"* viene aggiunto solo per vedere il disegno / tutti i passi. Senza questo blocco l'intero codice avviene in un secondo. Gli studenti dovrebbero provarlo senza questo blocco per capirne il significato.

Chiediamo agli studenti come accorcerebbero il codice, se possibile. C'è qualche parte senza ripetizioni? La risposta è SI. Invece di scrivere lo stesso codice 4 volte, in programmazione usiamo la ripetizione ciclica "*loop repeat"*.

Se vogliamo davvero vedere cosa disegniamo, dobbiamo mettere un blocco penna prima del ciclo di ripetizione "*loop repeat"*.

Se vogliamo che il gesso non ruoti durante la rotazione, clicchiamo su non ruotare "*don't rotate*" nel blocco di direzione.

[Fase 2]

Per l'attivazione del codice, gli studenti utilizzano il blocco eventi, ad es., quando si preme il tasto "S". Possono anche impostare il colore della penna e, come già sanno dalle attività precedenti, i seguenti blocchi: *penna su* (nel caso in cui fosse rimasto basso dal ciclo precedente), *pulisci* (cancella il disegno del ciclo precedente) e *vai a x, y* (il gesso inizia sempre da queste coordinate).

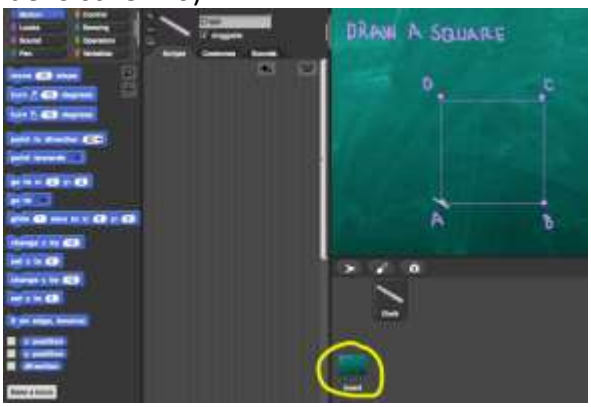
A volte succede che fermiamo il programma durante la riproduzione e uno *sprite* viene ruotato in "una strana direzione". Questo è un problema quando si riavvia un gioco, se uno sprite viene ruotato in modo errato, andrà per esempio verso il basso e non a destra al primo passaggio. Al fine di evitare questo problema, aggiungiamo un punto di blocco in direzione 90.

[Fase 3]

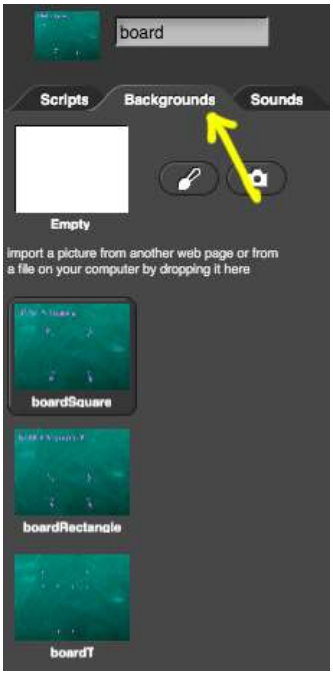
Dopo aver disegnato un quadrato, vogliamo disegnare un rettangolo.

Questo significa che dobbiamo cambiare lo sfondo. Lo faremo con due passaggi:

a) clicchiamo sullo sfondo (denominata scheda "*board*", sul lato destro dello schermo).

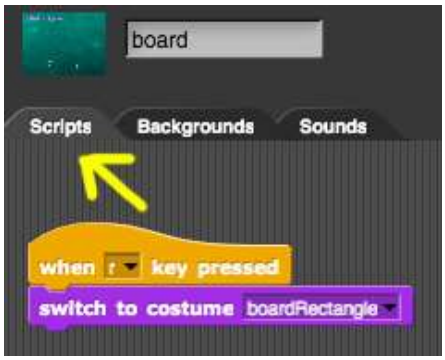


Cliccando su sfondo "*Backgrounds*" possiamo vedere tutti e tre gli sfondi necessari (*boardSquare, boardRectangle, boardT*), già preparati per questa attività.

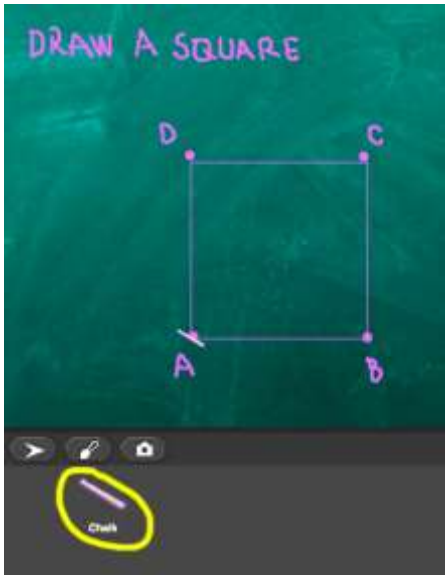


Per scrivere un codice gli studenti devono fare clic su "*Scripts*". Per programmare la modifica dello sfondo, scelgono un blocco eventi quando si preme il tasto "R" e, successivamente, passano

al “*costume boardRectangle”*.

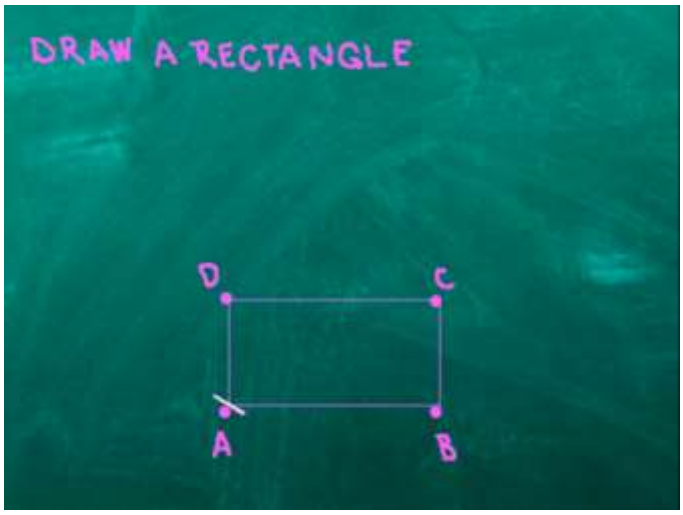


b) clicchiamo di nuovo sul gesso.



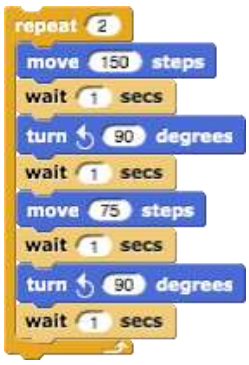
Sotto il codice della [Fase 2] gli studenti aggiungono un blocco, in cui diranno ad un giocatore cosa fare per cambiare lo sfondo, ovvero premere il tasto "R"

[Fase 4]



Dopo aver premuto il tasto “R”, lo sfondo passa a questo. Come prima, hanno bisogno di collegare i punti e disegnare un rettangolo. Gli studenti possono copiare i precedenti blocchi di codice e correggerli così che il programma disegnerà un rettangolo.

Cambiano la ripetizione del *loop*. Ora, questo ciclo si ripeterà 2 volte.



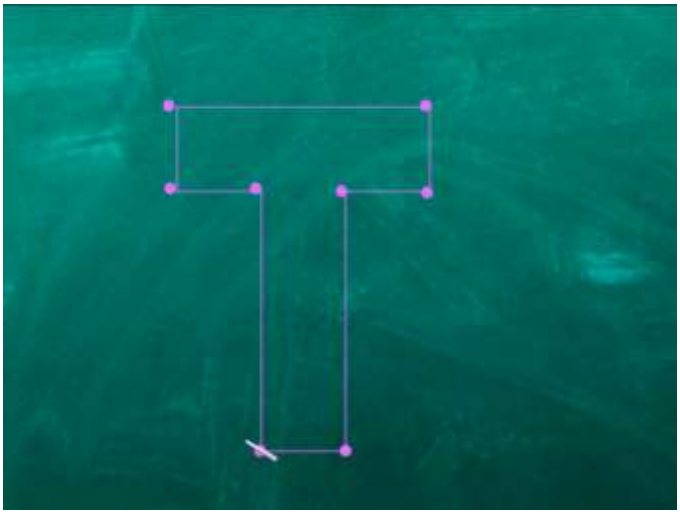
[Fase 5]

Dopo aver disegnato il rettangolo, gli studenti uniranno i punti per formare la lettera “T”. Ciò significa che devono cambiare lo sfondo, così in questa fase possono ripetere la [Fase 3], cambiano semplicemente la lettera (“T”) ed il costume (boardT):

a) cliccano sullo sfondo (denominato *board*, sul lato destro dello “*screen*”), e scrivono un codice per cambiare lo sfondo. Faranno ciò quando viene premuto il tasto “T” e poi passeranno alla scheda *costume boardT*.

b) cliccano di nuovo sul gessetto e sotto il codice di [Fase 4] aggiungono un blocco, in cui dicono ad un giocatore cosa fare per cambiare lo sfondo, ovvero premere il tasto “T”.

[Fase 6]



Dopo aver premuto il tasto “T”, lo sfondo cambia come questo. Come prima, hanno bisogno di collegare punti e disegnare una lettera "T". Gli studenti possono copiare i blocchi del codice precedente e correggerli. Gli studenti dovranno cambiare le coordinate iniziali, che non sono le stesse di prima. Sanno già come determinare le giuste coordinate dall'attività precedente.

Quindi scrivono un codice per disegnare una lettera "T". Devono scoprire il numero dei passi. Una possibile soluzione è:

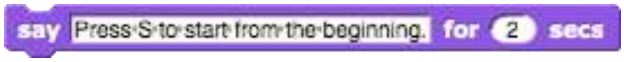
[Fase 7]

Dato che abbiamo cambiato lo sfondo, non possiamo tornare al primo sfondo per disegnare un quadrato. Quindi gli studenti dovranno aggiungere un ultimo codice. Ripetono [Fase 3/5].

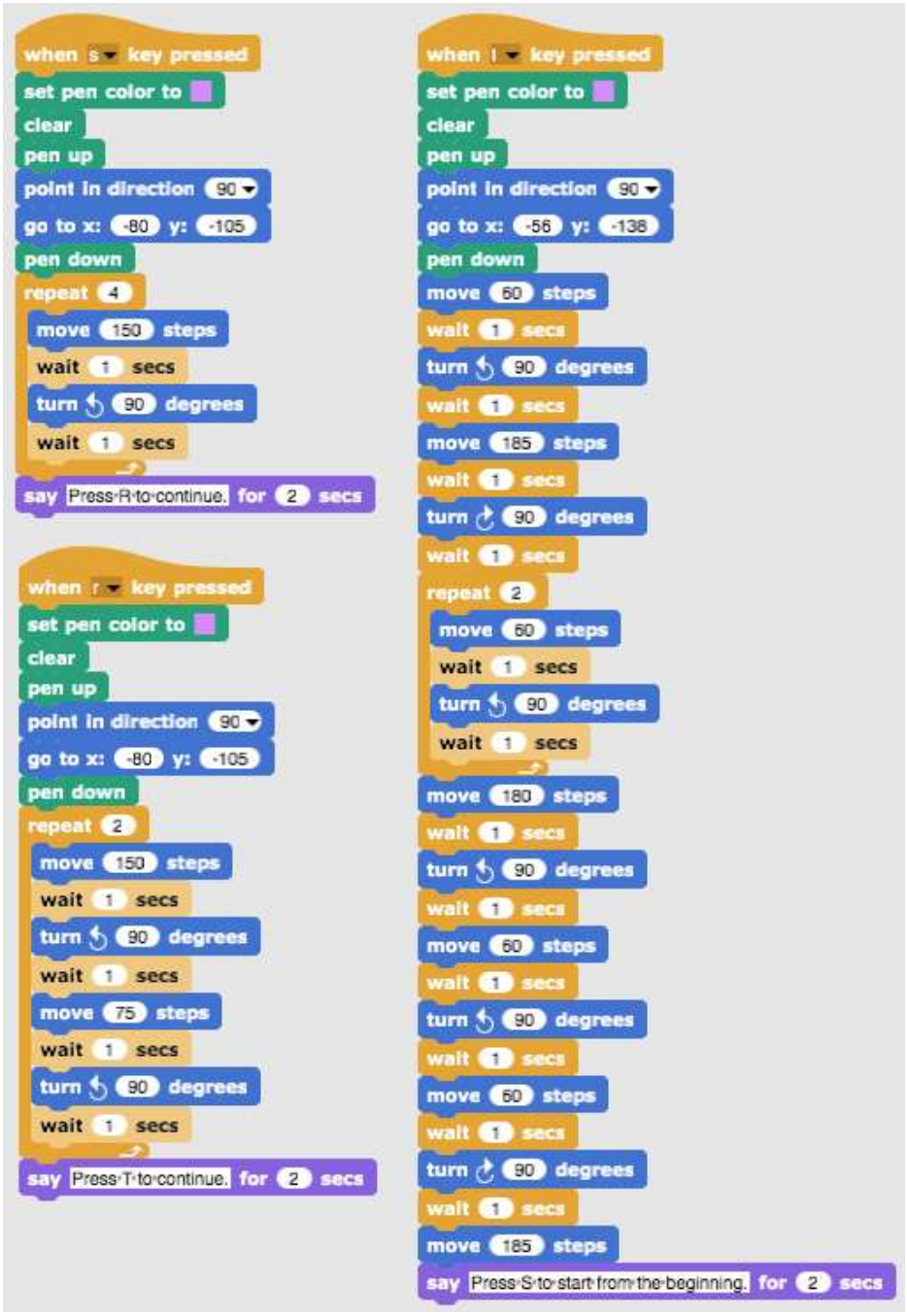
a) Cliccano sullo sfondo (denominato "*board*", sul lato destro dello screen), dove scrivono un codice per cambiare lo sfondo. Lo faranno quando si preme il tasto S, quindi passa alla scheda *"costume boardSquare"*.

b) Cliccano nuovamente sul gessetto e sotto il codice della [Fase

6] aggiungono un blocco in cui dicono a un giocatore cosa fare per cambiare lo sfondo, ovvero premere il tasto “S”.



[Codice Finale]



[Compiti aggiuntivi]

Gli studenti possono aggiungere attività aggiuntive in base ai loro

	desideri o possono seguire le seguenti: • Aggiungi un nuovo sfondo e disegna alcuni punti. • Scrivi un codice che collega i punti. Puoi disegnare uno sfondo o puoi usarne uno già dato.
Strumenti e risorse per gli insegnanti	• Attività completa in Snap!: https://snap.berkeley.edu/project?user=mateja&project=Drawing%20with%20a%20chalk • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Risorse/materiali per gli studenti	• Metà attività in Snap!: https://snap.berkeley.edu/project?user=mateja&project=Drawing%20with%20a%20chalk%20-%20Part • Istruzioni per lo studente (C4G8_InstructionsForStudent.docx)

Scenario di apprendimento 9 - Raccogliere la spazzatura e pulire il parco

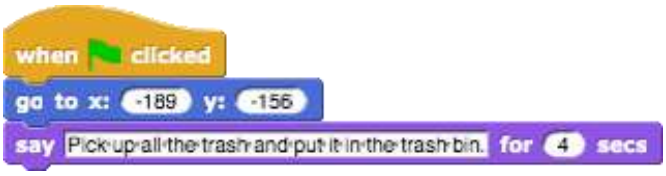
Learning Scenario Title	Raccogliere la spazzatura e pulire il parco
Precedente esperienza di programmazione	Impostare le coordinate iniziali Impostare la dimensione dello *sprite* Aggiungere il testo dello *sprite* Muovere gli oggetti con i tasti delle frecce usando gli eventi Usare il condizionale oggetto che sta toccando per definire lo stato dell'oggetto
Risultati di apprendimento	Obiettivi generali di apprendimento: ● variabili ● mostrare e nascondere gli *sprite* ● duplicare gli *sprite* ● duplicare i blocchi di codice ● condizionali Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente usa le variabili per contare i rifiuti raccolti ● lo studente sa nascondere lo *sprite* quando è toccato e lo mostra all'inizio ● lo studente sa come duplicare uno *sprite* (da una bottiglia a ad es., 4 bottiglie) ● lo studente sa come duplicare un blocco di codice (da una *sprite* bottiglia ad uno *sprite* carta) ● lo studente sa come usare i condizionali per verificare se uno *sprite* viene visualizzato e se viene raccolta tutta la spazzatura
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: il parco è pieno di spazzatura e la ragazza decide di pulirlo. Quando raccoglie tutta la spazzatura, la getta nel cestino. **Compiti**: Gli studenti iniziano con l'impostazione delle coordinate iniziali per la ragazza il gioco termina quando la ragazza raccoglie tutta la spazzatura e la getta nel bidone. Per fare ciò, gli studenti dovranno usare le variabili per contare i punti (1 bidone raccolto = 1 punto). Quando la ragazza tocca la spazzatura, la raccoglie, la spazzatura si nasconde ed il conteggio dei punti aumenta di 1. Quando raccoglie tutta la spazzatura, si reca al cestino. Se non

	raccoglie tutta la spazzatura e si reca prima alla pattumiera, questa dice di tornare quando raccoglie tutta la spazzatura. **Obiettivo: gli studenti impareranno come usare le variabili e come duplicare un blocco di codice o persino un intero *sprite*.**
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento basato sulla programmazione del gioco, *problem solving*
Didattica	Lezione frontale Lavoro individuale
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) Inizialmente viene dato agli studenti: ● uno sfondo ● lo *sprite* ragazza (con il codice del movimento), *sprite* bottiglia, *sprite* carta spazzatura *sprite* La ragazza vuole fare una passeggiata e godersi la giornata nel parco. Quando arriva lì, vede il parco pieno di spazzatura. Decide di raccogliere la spazzatura. Quando lo fa, può finalmente sdraiarsi e godersi la giornata di sole in un parco pulito. 

[Fase 1]

E' dato lo sfondo e anche lo *sprite* ragazza avente un codice per il movimento con i tasti e il condizionale per toccare la linea marrone.

Gli studenti devono impostare le coordinate iniziali per la ragazza con i blocchi *vai a x* e *y*. E' importante che le coordinate siano sul percorso. Gli studenti già conoscono come impostare le coordinate dalle precedenti attività. Aggiungono anche alcune istruzioni come ad es.:



[Fase 2]

Per contare il numero dei rifiuti raccolti dalla ragazza, utilizzeremo le variabili.

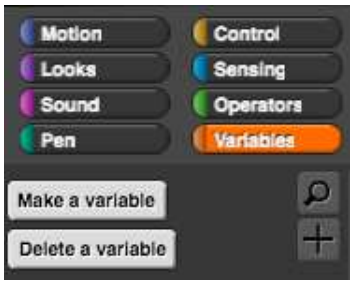
Cosa è una variabile?

Una variabile è come una scatola in cui archiviamo alcune informazioni.

Nel nostro caso, possiamo vedere la nostra variabile come una

scatola, denominata punti. Quando la ragazza raccoglie la spazzatura, l'immondizia viene accumulata attrverso una variabile "punti". Questa variabile conta quanti rifiuti ha scelto la ragazza.

Come facciamo una variabile?.



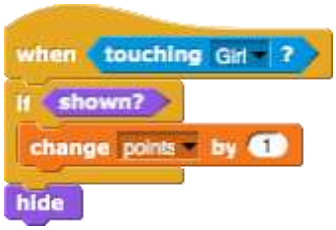
Selezioniamo un blocco arancione Variabili "*Variables"*, poi clicchiamo sul pulsante - Crea una variabile - "*Make a variable"*, scriviamo il nome della variabile "*Variable name"* e cliccchiamo OK. Quindi appare il blocco "punti".

Se la casella è selezionata, la variabile con il suo valore sarà visibile sullo schermo:

All'inizio del gioco, il valore della variabile deve essere 0, poiché non è stata raccolta alcuna spazzatura. Sotto il codice della [Fase 1] lo studente aggiunge un blocco - imposta ___ a 0 - "*set __ to 0".* Cliccando sul menu a discesa, si sceglie la variabile appropriata, ovvero punti.

[Fase 3]

Gli studenti scrivono un codice per una bottiglia. L'idea è che lo *sprite* scompaia (si nasconda) quando tocca la ragazza. Quindi il codice inizia quando lo *sprite* tocca la ragazza. Poi dobbiamo pensare

	in quale caso lei raccoglie la spazzatura. Se diciamo che il cestino si nasconde quando viene riempito dalla spazzatura, possiamo raccogliere solo se è ancora lì = viene mostrato. Se lo *sprite* (bottiglia) è ancora lì, lo prendiamo "e lo mettiamo nella variabile “scatola. Prima avevamo 0 elementi nella variabile *punti*, ora abbiamo 1. Possiamo vedere che raccogliendo il cestino cambiamo il numero della variabile (*punti*) di 1, ovvero aumentiamo di 1. Quando la spazzatura viene raccolta, la nascondiamo.  Ora possiamo verificare se il nostro codice sia corretto. Clicchiamo sulla bandiera verde e prendiamo la bottiglia. La bottiglia deve scomparire e il numero di punti deve essere 1. Quindi vogliamo giocare di nuovo e clicchiamo sulla bandiera verde. Che succede? Dov'è la bottiglia adesso? La bottiglia è nascosta, l'abbiamo nascosta prima. Quindi, all'inizio del gioco, dobbiamo programmare che la bottiglia sia mostrata. Lo facciamo selezionando il blocco mostra “*show*”.  [Fase 4] Ora gli studenti vogliono avere più bottiglie nel loro gioco in modo da poter duplicare facilmente il loro *sprite*. Cliccano con il tasto destro sullo *sprite* e scelgono duplicare.

Ora cliccano semplicemente con il mouse sulla nuova bottiglia e la trascinano da qualche parte all'interno del labirinto.

Possono ripetere questo passaggio e duplicare nuovamente la bottiglia.

[Fase 5]

Ora gli studenti vogliono avere lo stesso codice per lo *sprite* carta. Possono duplicare il codice della bottiglia cliccando con il tasto destro del mouse sul blocco del codice:

E lo rilasciano nello *sprite* di carta facendo clic con il mouse sullo *sprite* di carta .

Ripetono questo passaggio per duplicare il blocco del codice *quando si clicca sulla bandiera verde – mostra*.

Possono anche ripetere [Fase 4] e duplicare l'intero *sprite* carta per avere per avere più rifiuti di carta nel labirinto.

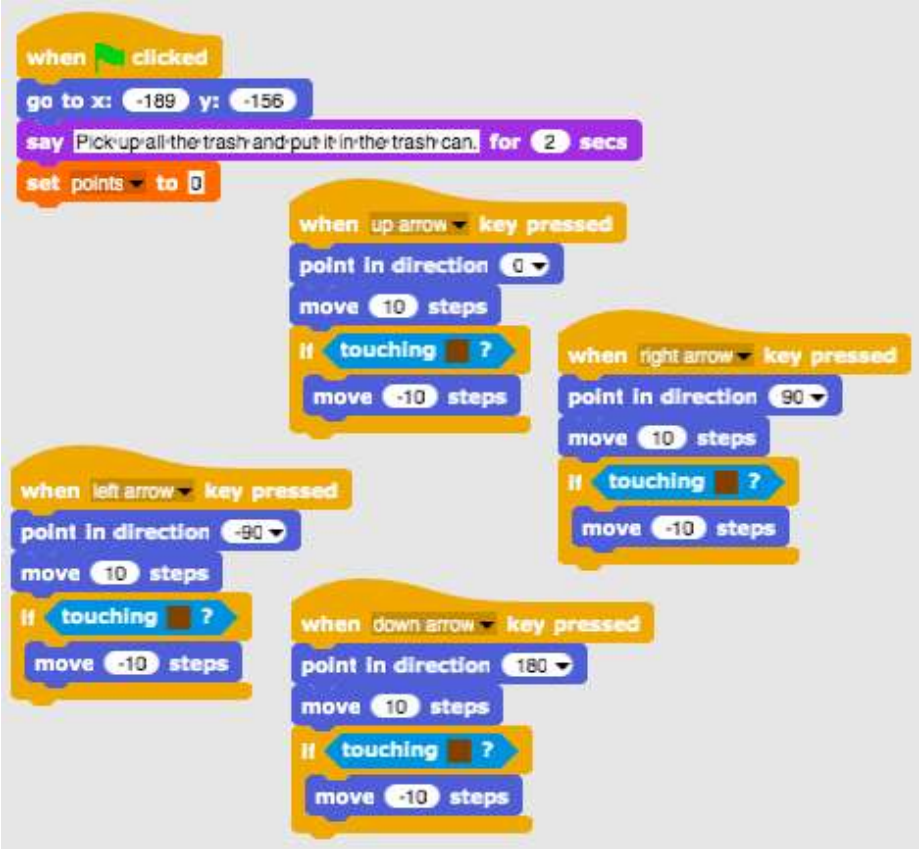
[Fase 6]

L'ultima cosa che gli studenti devono fare è scrivere un codice per il cestino. Lo *sprite* cestino è già stato dato, possono spostarlo ovunque all'interno del labirinto. Anche questo codice sarà attivato quando la ragazza lo tocca.

Il cestino dovrà verificare se tutti i rifiuti vengono raccolti. Grazie alla variabile punti, questo sarà facile da fare. Diciamo che abbiamo 8 *sprite* di spazzatura nel gioco, quindi gli studenti devono verificare se il numero di punti è uguale a 8. Se lo è, significa che tutta la spazzatura è stata raccolta, altrimenti no. Per programmare questo useranno l'istruzione "*if*" e aggiungeranno del testo per dire al giocatore se ha raccolto o meno tutta la spazzatura.

[Codice finale]

Ragazza



Bottiglie / Carte

Cestino

	[Compiti aggiuntivi] Gli studenti possono aggiungere attività in base ai loro desideri o possono seguire quelle descritte di seguito: • Aggiungere un altro tipo di rifiuto (ad es. Rifiuti organici). • Il cestino dice ad es. "Hai raccolto X bottiglie, carte Y e angurie Z". • Se un giocatore raccoglie tutta la spazzatura, la pattumiera dice: “Congratulazioni! Hai raccolto tutta la spazzatura! ” • Se un giocatore non raccoglie tutta la spazzatura, la pattumiera gli dice quale spazzatura non è stata raccolta, ad es. “Non hai raccolto tutte le bottiglie. Non hai raccolto tutte le angurie. " e "Torna quando raccogli tutta la spazzatura".
Strumenti e risorse per gli insegnanti	• Attività complete in Snap!: https://snap.berkeley.edu/project?user=mateja&project=Picking%20up%20trash%20and%20cleaning%20the%20park • Attività in Snap! con compiti aggiuntivi (possibile soluzione): https://snap.berkeley.edu/project?user=mateja&project=Picking%20up%20trash%20and%20cleaning%20the%20park%20%2B%20Add.%20Task • Lajovic, S. (2011). *Scratch. Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Risorse/materiali per gli studenti	• Metà attività in Snap!: https://snap.berkeley.edu/project?user=mateja&project=Picking%20up%20trash%20and%20cleaning%20the%20park%20-%20Part • Istruzioni per lo studente (C4G9_InstructionsForStudent.docx)

Scenario di apprendimento 10 – Nutrire i gatti

Titolo dello scenario di apprendimento	Nutrire i gatti
Precedente esperienza di programmazione	• condizionali (blocchi Se, Se-altrimenti) • testo stampato (blocco dire)
Risultati di apprendimento	Obiettivi generali di apprendimento: • impostazione e aumento del valore della variabile • assegnazione di un valore variabile all'interno / all'esterno del *loop* • cicli (ripetere n. volte) • numeri casuali • concatenazione di stringhe • operatori: logico, aritmetico • input Obiettivi specifici di apprendimento orientati al pensiero algoritmico: • lo studente riconosce la situazione per l'utilizzo del ciclo (ripetizione n. volte) • lo studente distingue tra l'assegnazione del valore in ogni iterazione del ciclo e una volta prima del ciclo. • lo studente usa il blocco ingresso "*input*" per ottenere il numero da un giocatore. • lo studente sa come usare l'operatore aritmetico per generare la risposta giusta. • lo studente usa le istruzioni *Se-altrimenti* per verificare la correttezza dell'input del giocatore e dare una risposta appropriata. • lo studente sa come usare una variabile per conteggiare le risposte corrette.
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: si programma un gioco in cui il giocatore dovrà eseguire dieci calcoli di moltiplicazione e contare le risposte corrette. **Compiti**: si programma l'attività in cui Marta, il guardiano del rifugio, chiederà ripetutamente al giocatore il numero di gatti che può nutrire in una determinata stanza. Il numero dipende da quello delle ciotole e

	dalla loro dimensione. Questi due numeri devono essere assegnati in modo casuale per ogni stanza. Dobbiamo anche avere un contatore che calcolerà le risposte giuste. Il custode del primo rifugio deve spiegare l'incarico al giocatore e poi inizia il gioco. Il gioco termina quando lei chiede il numero di gatti 10 volte. Ogni volta lei deve dare una risposta in merito al fatto se il numero inserito è corretto o meno. Dopo l'attività deve riepilogare il punteggio ottenuto dal giocatore, ossia dice quante volte il giocatore ha risposto correttamente e quante volte ha sbagliato. **Obiettivo**: **agli studenti verrà illustrato il concetto di assegnazione di più valori casuali variabili all'interno di un ciclo e di come sia diverso da quando lo facciamo al di fuori. Impareranno anche come ottenere, testare e contare gli input corretti del giocatore.**
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento collaborativo, *problem solving*
Didattica	Lezione frontale Lavoro individuale / Lavoro in coppia / Lavoro di gruppo
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) La custode del rifugio sta cercando di dare da mangiare ai suoi gatti in dieci stanze diverse. In ogni stanza c'è un numero casuale di ciotole (da 2 a 10), che hanno dimensioni diverse (da 1 a 5) ma all'interno di ogni stanza tutte le ciotole hanno le stesse dimensioni. La dimensione della ciotola indica quanti gatti vi possono mangiare, ad esempio se la dimensione della ciotola è 3 significa che 3 gatti possono mangiare. Aiuta a trovare il numero di gatti che può nutrire in ogni stanza. [Fase 1] Innanzitutto, chiediamo agli studenti di disegnare uno sfondo

interessante per il gioco. Se vogliamo risparmiare tempo, possiamo fornirglielo.

[Fase 2]

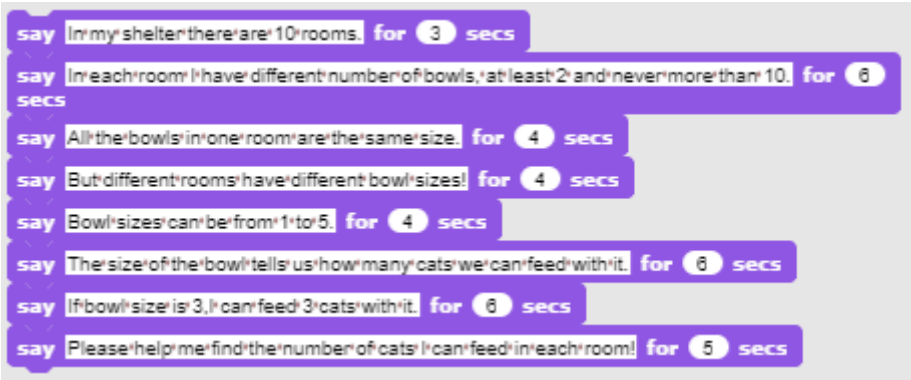
Dobbiamo selezionare un nuovo costume per lo *sprite* predefinito che rappresenterà il guardiano del rifugio per i gatti.

[Fase 3]

Per memorizzare i valori necessari abbiamo bisogno di tre variabili: 1) per memorizzare il numero di risposte corrette, 2) per assegnare il valore casuale per il numero di ciotole all'interno di ogni stanza (2-10) e 3) per assegnare il valore casuale per la capacità della ciotola (1-5). Il contatore di risposte corretto dovrà essere impostato su 0 e gli altri due non dovranno essere impostati prima del ciclo perché assegneremo loro nuovi valori casuali in ogni iterazione del ciclo. Vogliamo anche contare le stanze, ma non abbiamo bisogno di variabili speciali per contarle. Il suo numero verrà posizionato inizialmente su un valore 1 e, quindi, aumentato di 1 per ogni iterazione fino al raggiungimento del valore 10. Questo verrà replicato nel conteggio delle stanze.

[Fase 4]

Successivamente dobbiamo programmare le istruzioni per il giocatore. Lo facciamo usando la sezione - Aspetto "*Looks/say[string]*" e attendiamo il blocco [n] secondi.



[Fase 5]

Avviamo una discussione con gli studenti in merito a quali sono le azioni che accadranno in ogni stanza. Si tratta di comandi che dovranno essere inseriti all'interno del blocco del ciclo per essere eseguiti in ogni iterazione del ciclo.

Per prima cosa dovremo assegnare casualmente il valore (1-10) per il numero di ciotole e la dimensione della ciotola in quella stanza (1-5). Successivamente dovremo chiedere a un giocatore quanti gatti possiamo nutrire in quella stanza. La sua risposta dovrà essere verificata, fornendo una risposta adeguata e ricordando se fosse corretta (contatore delle risposte corrette). Alla fine di ogni iterazione dovremo anche aumentare il numero della stanza di 1.

[Fase 6]

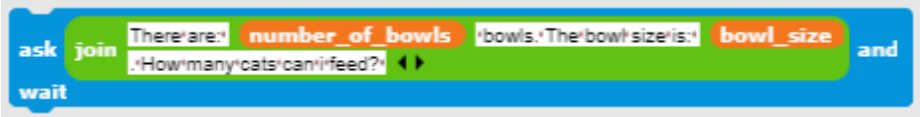
Per assegnare casualmente i valori per il numero di ciotole e le loro dimensioni useremo Variabili / imposta il valore [opzioni] "*Variables/set [options] value*" con Operatori / scegli casuali [n] su [m] "*Operators/pick random [n] to [m]*".

[Fase 7]

Vogliamo chiedere al giocatore il numero di gatti che possiamo nutrire inserendo la richiesta in Sensing/ask [string] e il blocco attendere, perché altrimenti verrà visualizzato per alcuni secondi e aggiornato con una nuova striga di testo. In questo modo i giocatori possono facilmente dimenticare quante ciotole / dimensioni ci sono nella stanza attuale. Per creare una stringa che sarà costruita da una combinazione di testo e riferimenti a variabili, utilizziamo il blocco "*Operators/join [string1][string2]*". Dovremo espandere questo blocco in modo che si adatti all'intera frase.

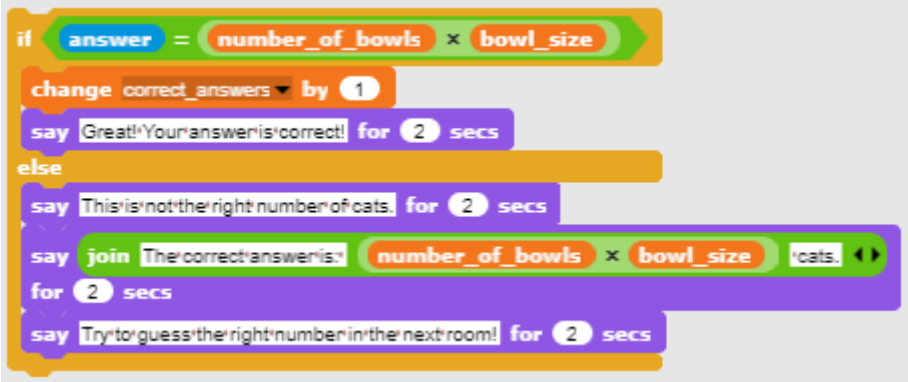
[Fase 8]

Dobbiamo mettere questa lunga stringa dentro "Sense/Ask [string]" e il blocco attendere per ottenere la risposta dal giocatore.



[Fase 9]

Quando il giocatore risponde, dobbiamo verificare la correttezza. Ci sono solo due possibili situazioni: il giocatore ha sbagliato o meno, quindi useremo il blocco "*If-Else*". La risposta giusta è il valore ottenuto moltiplicando il numero di ciotole con la dimensione della ciotola. Dobbiamo verificare se la risposta dei giocatori è uguale a quel numero. Se la risposta è corretta, aumentiamo il contatore delle risposte corrette di 1 e forniamo la risposta. In caso contrario, restituiamo solo il riscontro negativo. Non dobbiamo contare le risposte errate perché possiamo calcolarle dal contatore delle risposte corrette.



[Fase 11]

Ora dobbiamo selezionare un ciclo "*for loop*". E' meglio scegliere "*for loop*" perché la variabile utilizzata per iterare, replica il conteggio delle stanze.

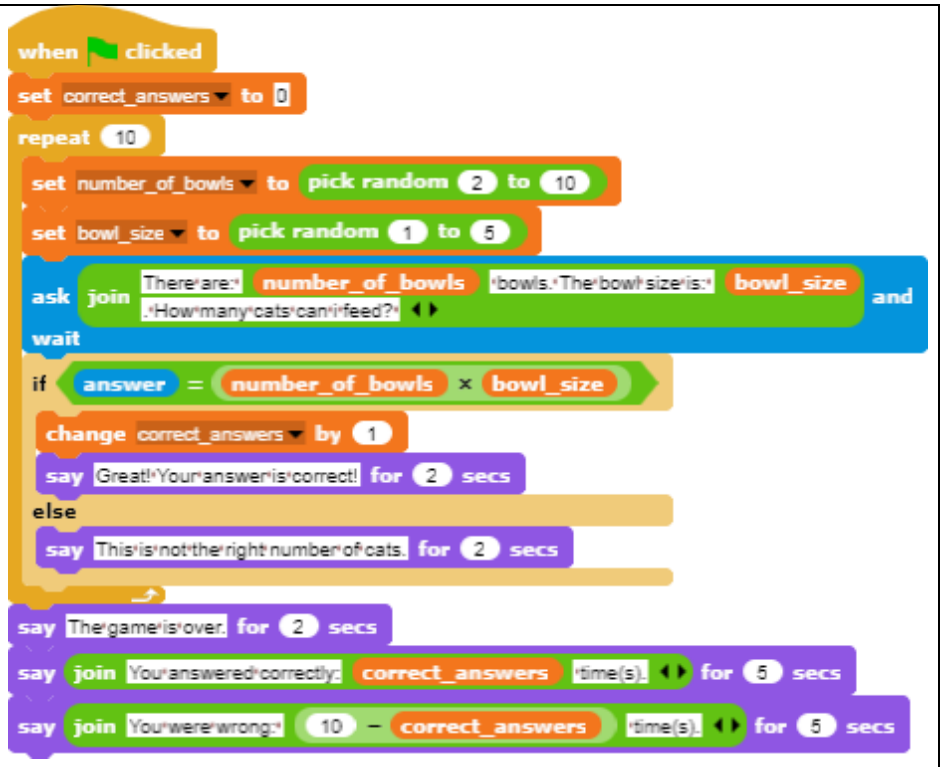
[Fase 12]

Quando il ciclo si interrompe, il gioco è finito. Forniamo le informazioni sul rendimento del giocatore. Il numero di risposte corrette è memorizzato nel contatore delle risposte corrette. Di conseguenza, è possibile calcolare il numero di risposte errate.

[Codice finale]

[Versione base dell'attività]

Per risparmiare tempo possiamo usare la versione base dello scenario. In una versione base sono inclusi tutti i concetti essenziali, altre funzionalità sopra descritte possono essere utilizzate come aggiornamenti successivi.

	
Strumenti e risorse per gli insegnanti	• Attività complete in Snap!: https://snap.berkeley.edu/project?user=zapusek&project=cat_feeding_2 • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Risorse/materiali per gli studenti	• Modello in Snap!: https://snap.berkeley.edu/project?user=zapusek&project=cat_feeding_template • Istruzioni per lo studente (C4G10_InstructionsForStudent.docx)

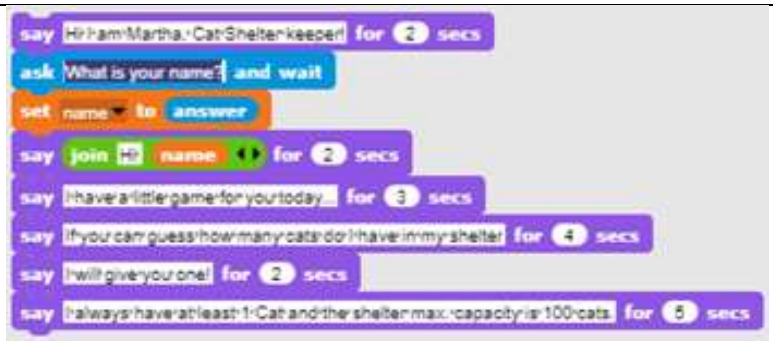
Scenario di apprendimento 11 - Indovinare il numero di gatti in un rifugio

Titolo dello scenario di apprendimento	Indovinare il numero di gatti in un rifugio
Previous programming experience	• condizionali (blocco Se) • testo stampato (blocco dire)
Risultati di apprendimento	Obiettivi generali di apprendimento: • valori casuali • assegnazione di variabili • usare gli *input* • il ciclio "ripetere fino" • operatori di confronto • il contatore Obiettivi specifici di apprendimento orientati al pensiero algoritmico: • lo studente assegna valori casuali alle variabili • lo studente usa i blocchi di ingresso per ottenere il numero da un giocatore • lo studente usa il ciclio "ripeti fino a" per chiedere ripetutamente al giocatore di inserire il numero ed eseguire un test del valore • lo studente esegue test del valore con le istruzioni "*Se*" e gli operatori di confronto e fornisce appropriate risposte • lo studente imposta le condizioni del ciclo per la ripetizione per verificare se il gioco è finito • lo studente si rende conto che non deve testare se il gioco è finito, perché è implicitamente coperto dalle condizioni • lo studente imposta un contatore per contare le congetture del giocatore e usa il valore finale per distinguere tra i risultati possibili
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: si programma un gioco semplice in cui all'inizio un numero casuale da 1 a 100 viene assegnato in modo casuale ad una variabile. Il giocatore proverà ad indovinarlo digitando i numeri. Riceverà sempre una risposta (se il numero di input sarà maggiore,

	minore o uguale al valore casuale). **Compiti**: Programma Marta la custode del rifugio dei gatti. La custode del rifugio vuole che si indovini il numero esatto di gatti che ha nel suo rifugio. Chiede al giocatore il suo nome e quindi spiega il compito. Successivamente Marta deve salutare il giocatore con il suo nome e quindi chiedere il numero esatto di gatti che ha nel suo rifugio. Quando il giocatore inserisce la sua ipotesi, Marta deve dire: 1) se il numero di input è inferiore al numero effettivo, dice: "il numero di gatti è più alto", 2) se il numero di input è superiore al numero effettivo, dice: "il numero di gatti è inferiore", 3) se il numero inserito è corretto, dice: "Eccellente, hai indovinato il numero giusto". Programma un contatore che conterà ogni tentativo del giocatore. Quando il giocatore indovina il numero giusto devi controllare se il numero di tentativi è inferiore a 5. Se il giocatore indovina il numero di gatti in meno di 5 tentativi, ottiene il gatto, altrimenti viene chiesto di giocare di nuovo. **Obiettivo: Agli studenti verrà illustrato come ripetere il ciclo e come impostare la condizione per monitorare implicitamente il requisito che interrompe il gioco. Impareranno anche ad usare la variabile in situazioni diverse: impostare un valore casuale, un contatore o ottenere l'input dei giocatori.**
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento collaborativo, *problem solving*
Didattica	Lezione frontale Lavoro individuale/ lavoro in coppia / lavoro di gruppo

Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) La custode del rifugio per gatti, Marta, vuole che si indovini il numero esatto di gatti che ha nel suo rifugio. Il numero può essere compreso tra 1 e 100. Quando il giocatore digita il numero, lei risponde se è inferiore, maggiore o uguale al numero corretto di gatti. Se il giocatore indovina il numero di gatti in meno di cinque tentativi, ottiene il gatto, altrimenti viene chiesto di giocare di nuovo. [Fase 1] Il primo compito è creare uno sfondo interessante per il gioco. Gli studenti possono disegnarlo da soli oppure utilizzare immagini aventi una licenza gratuita da Internet. Per risparmiare tempo, possiamo preparare in anticipo lo sfondo.  [Fase 2] Dobbiamo selezionare un nuovo costume per lo *sprite* di tartaruga predefinito che rappresenterà il guardiano del rifugio per i gatti. 

	[Fase 3] Avviamo una discussione con gli studenti in merito al fatto che questo gioco può essere interessante per giocarci più di una volta, se il numero di gatti è impostato casualmente. Per avere quel numero casuale disponibile per il confronto dei numeri di ipotesi, dobbiamo anche memorizzarlo in una variabile. La variabile è ora (supponiamo che non conoscano ancora il concetto di elenchi) l'unico modo per ricordare un determinato valore in Snap!. Questo deve avvenire all'avvio del programma (Evento / Quando si clicca sulla bandiera verde).  [Fase 4] Il custode del rifugio chiede al giocatore il suo nome per salutarlo. Questo viene fatto con “Sense/ask[string]” e il blocco aspetta. La risposta del giocatore è memorizzata nella variabile denominata risposta “*answer*”. Per salutare, dobbiamo unire la stringa memorizzata nella risposta variabile con un saluto. Questo viene fatto con il blocco “Operators/join[string1][string2]”. Per visualizzare il testo, usiamo “Looks/say [string]” per un blocco di n. secondi. Usiamo anche questi blocchi per scrivere le istruzioni del gioco. Possiamo anche sottolineare che è importante fare attenzione alla durata della visualizzazione del testo.



[Fase 5]

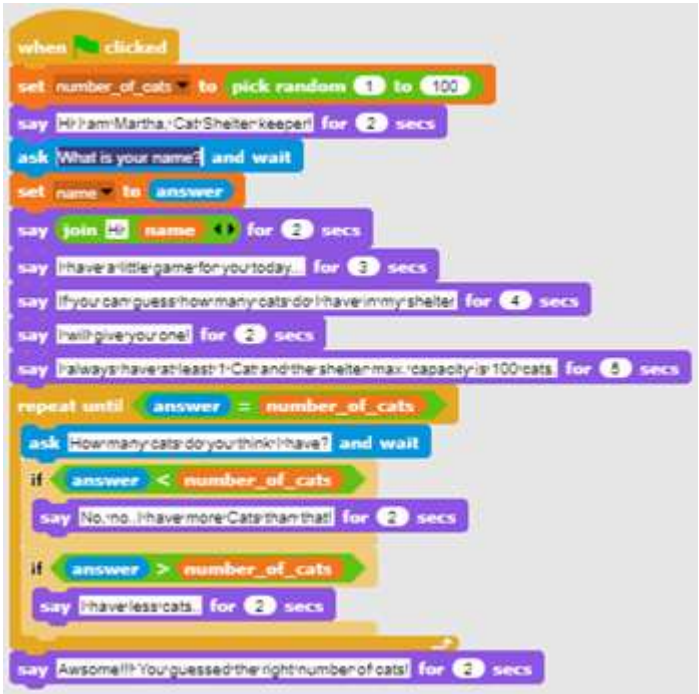
Condividiamo con gli studenti il fatto che non è possibile prevedere quante volte i giocatori dovranno indovinare il numero giusto. Chi è molto fortunato indovina al primo tentativo, ma forse ce ne vorranno 5, o anche di più, non possiamo dirlo! Questo è il motivo per cui dobbiamo scegliere il ciclo giusto per l'attività data. Il custode deve chiedere più volte il numero e fornire il riscontro fino a quando il giocatore non indovina il numero giusto. L'unico ciclo in cui possiamo implementare l'esecuzione desiderata viene ripetuto "*until[condition] loop*". La condizione è relativamente facile da vedere, dobbiamo ripeterla fino a quando la risposta del giocatore, che è memorizzata nella variabile risposta è uguale al valore memorizzato nella variabile "*cat_number*".

[Fase 6]

Successivamente, dobbiamo chiedere agli studenti quali sono i comandi che verranno inseriti all'interno del ciclo "*loop*". Qual è l'attività o i comandi che verranno ripetuti fino a quando il giocatore non indovina il numero giusto? Innanzitutto, dobbiamo chiedere al giocatore di inserire il numero, quindi dobbiamo dare una risposta in base al valore di quel numero.

[Fase 7]

L'ultima cosa da spiegare o discutere con gli studenti è quando finirà questo ciclo e cosa implica. Quando le risposte del giocatore saranno uguali al numero dei gatti, entrambe le condizioni all'interno del ciclo "*loop*" potrebbero essere false, quindi il "*loop*" passerà alla successiva iterazione, controllando le condizioni del ciclo. Quando la condizione sarà vera il ciclo terminerà e verranno eseguiti i comandi successivi. Per parafrasare, quando il ciclo termina sappiamo che quel giocatore ha indovinato il numero giusto. Quindi, ora, possiamo rispondere di conseguenza.

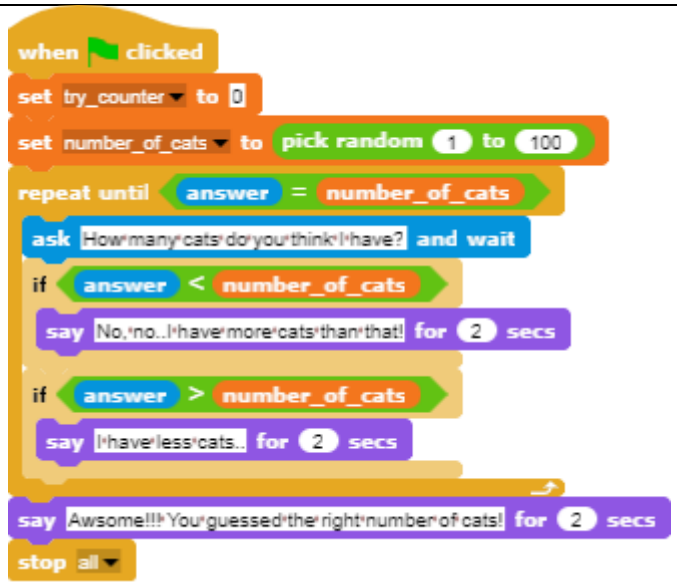


	[Fase 9] Dobbiamo creare una nuova variabile che avrà il ruolo di un contatore con un valore iniziale di 0. Condividiamo con gli studenti l'importanza della variabile iniziale e la differenza tra l'impostazione del valore ed il suo aumento. Quando impostiamo il valore di una variabile, il valore precedente viene perso. Questo non va bene per un contatore. Se vogliamo aumentare il valore della variabile di un certo numero, aggiungiamo a quel numero il valore precedente della variabile. Questo è esattamente ciò che vogliamo in questa situazione. Ogni volta che il giocatore inserisce un nuovo numero, vogliamo aumentarlo di 1. [Fase 10] Dopo la risposta giusta, dobbiamo controllare il valore della variabile contatore per decidere se il giocatore otterrà il gatto o meno. Poiché Snap! ha solo un operatore logico inferiore (<) e non un operatore inferiore o uguale, la condizione per decidere se il giocatore prende il gatto è "*cat_counter* < 6". Questo è anche un buon esempio per l'uso del blocco condizione "*If-Else*" perché di differenziano i due casi.

[Codice finale]

[Versione base dell'attività]

Per risparmiare tempo possiamo usare la versione base dello scenario. In una versione base sono inclusi tutti i concetti essenziali, altre funzionalità sopra descritte possono essere utilizzate come aggiornamenti successivi.

	
Strumenti e risorse per gli insegnanti	• L'intera attività in Snap!: https://snap.berkeley.edu/project?user=zapusek&project=cats_in_a_shelter • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Risorse/materiali per gli studenti	• Modello in Snap!: https://snap.berkeley.edu/project?user=zapusek&project=cats_in_a_shelter_template • Istruzioni per lo studente (C4G11_InstructionsForStudent.docx)

SCENARI AVANZATI DI APPRENDIMENTO

Scenario di apprendimento 12 - Catturare cibi sani

Titolo dello scenario di apprendimento	Catturare cibi sani
Precedente esperienza di programmazione	Aggiungere il testo allo *sprite* Mostrare e nascondere lo *sprite* Usare il punto di direzione Usare l'assegnazione causale Usare le variabili per il conteggio dei punti Usare la ripetizione del ciclo Usare il ciclio (ripetere per esempre) Usare i condizionali
Risultati di apprendimento	Obiettivi generali di apprendimento: ● variabili ● condizionali ● cicli ● punto di direzione ● causalità Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente utilizza la variabile per impedire l'avvio del gioco prima che la ragazza finisca di parlare (facoltativo) ● lo studente utilizza l'istruzione "*if*" per verificare (con l'aiuto di una variabile) se il cibo può iniziare a muoversi ● lo studente usa la ripetizione ciclica per il movimento del cibo fino a quando i punti sono inferiori a 5 ● lo studente usa il punto nella direzione 180 (in basso) per gli sprite che si spostano in basso ● lo studente usa la causalità per selezionare il numero di passi ● lo studente usa la causalità per spostarsi in posizione casuale ● lo studente usa la causalità per passare alla posizione x (*random*), y (*fixed*) (opzionale)
Obiettivo, Compiti e	**Breve descrizione:** La ragazza sta prendendo cibo. Deve stare

Breve Descrizione delle Attività	attenta: solo gli alimenti sani portano punti! **Compiti**: Gli studenti devono programmare due diversi *sprites*, una ragazza che dà istruzioni, dice cosa fare per iniziare il gioco e conta i punti; il cibo che casualmente cade dalla parte superiore dello schermo. Inoltre, gli studenti possono aggiungere una variabile e un'istruzione "*if*" per impedire il movimento del cibo prima che la ragazza smetta di parlare. **Obiettivo: Gli studenti impareranno come muoversi casualmente per X passi scegliendo una posizione e anche come utilizzare variabili e condizionali per prevenire altri eventi.**
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento basato sulla programmazione del gioco, *problem solving*
Didattica	Lavoro individuale / Lavoro in coppia
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) La ragazza sta prendendo cibo. Ogni cibo sano porta 1 punto, mentre ogni malsano perde 1 punto. Il gioco inizia con alcune istruzioni, fornite da una ragazza. Successivamente lei scompare e appare il cibo. Quando il giocatore raccoglie 5 punti, il cibo scompare e ricompare la ragazza.

[Fase 1]

Questa attività è intesa come lavoro individuale o lavoro in coppia. Un insegnante fornisce alcuni indizi, spiega alcune parti più difficili e aiuta quando necessario.

Inizialmente viene dato agli studenti:

- Sfondo
- Girl sprite

Gli studenti scelgono lo sfondo e aggiungono uno sprite principale, ad es. una ragazza. La ragazza dà alcune istruzioni all'inizio e poi si nasconde. Come abbiamo visto dalle attività precedenti, è bene scrivere mostra "*show*" quando si clicca sulla bandiera. Il codice è, ad esempio:

Torneremo su questo *sprite* più tardi. Scriviamo un codice per un frutto ora.

[Fase 2]

Gli studenti aggiungono un nuovo *sprite*, un cibo sano, ad es., una mela.

Prima, programmano un movimento dello *sprite* che va dall'alto verso il basso, quindi selezionano i seguenti blocchi:

Se non vogliono che la mela si capovolga, possono scegliere la terza opzione che non ruota nel blocco di direzione.

Per rendere il gioco più interessante, il numero di passi può essere scelto casualmente, quindi la velocità non sarà sempre la stessa. Ad es.:

Il passo successivo è pensare a cosa succede quando la mela arriva in fondo allo schermo?

In questo caso gli studenti possono usare un blocco "*touching edge*" in combinaztione con l'istruzione "*if*". Se la mela tocca il bordo, verrà spostata in una posizione casuale. I blocchi per il movimento ci offrono il prossimo blocco:

Questo comando sceglierà casualmente le coordinate x e y e la mela potrebbe apparire ovunque sullo schermo (guarda i punti rossi sull'immagine).

Se vogliamo che la mela appaia sempre nella parte superiore dello schermo, il valore y può essere fissato e solo il valore x verrà scelto a caso. Con il seguente codice la mela apparirà

sempre nella parte superiore dello schermo (guarda i punti rossi sull'immagine).

[Fase 3]

Gli studenti possono ora creare una variabile, punti, che useranno per il conteggio. I punti devono essere impostati sullo 0 inizialmente (sullo *sprite* della ragazza).

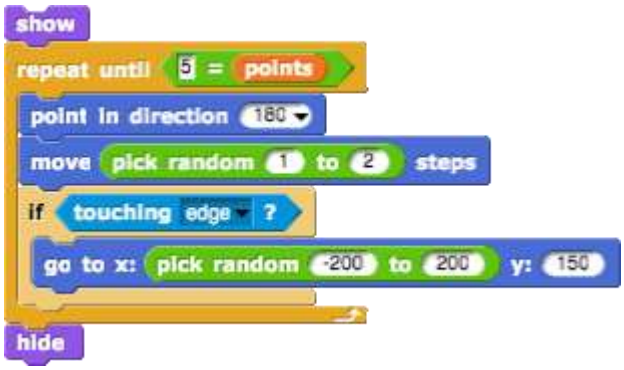
[Fase 4]

Se vogliamo che la mela si muova ripetutamente, abbiamo bisogno di un ciclo "*loop*". Gli studenti possono utilizzare una ripetizione ciclica ed impostare una condizione. Ad esempio, vogliono che il gioco finisca quando raggiungono i 5 punti. Quindi la condizione sarà punti = 5 e il ciclo si ripeterà fino a quando la condizione è falsa. Quando la condizione è vera, e il giocatore raggiunge 5 punti, il ciclo si fermerà.

[Fase 5]

Non vogliamo che la mela venga mostrata all'inizio, ma dopo che la ragazza ha dato le sue istruzioni. Gli studenti possono programmare la mela per mostrala quando viene premuto il tasto. Ovviamente, hanno dovuto aggiungere il blocco mostra "*show*" prima della ripetizione del ciclo e nasconderlo dopo.

L'intero codice per ora è:



[Fase 6]

Cosa succede quando si clicca sulla mela (o *mouse-entered*)?

La mela deve nascondersi, contare i punti, cambiare posizione e mostrarsi di nuovo. I punti saranno cambiati di 1 e per la posizione gli studenti possono usare lo stesso codice di prima.

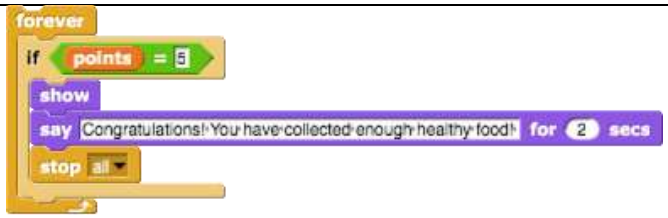
[Fase 7]

Torniamo alla ragazza.

La ragazza deve ora riapparire e dire, ad es. *Congratulazioni!*

Avremo bisogno di un ciclo per sempre "*loop forever*", che controllerà se abbiamo raggiunto 5 punti. Se lo facessimo, la ragazza mostrerà e dirà qualcosa. Dopodiché aggiungeremo un blocco per interrompere tutto "*stop all*".

Lascia che gli studenti capiscano cosa significa questo stop (senza sosta, la ragazza dirà per sempre *Congratulazioni*).



[Fase 8]

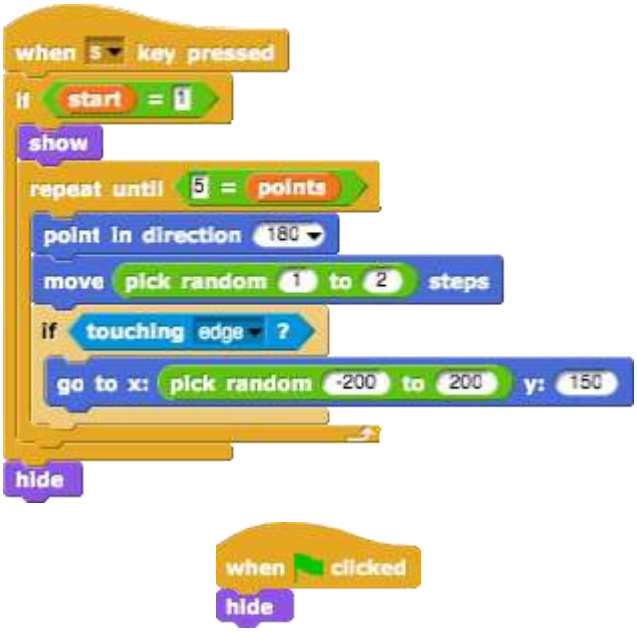
Quando si gioca di nuovo, gli studenti conoscono già tutte le istruzioni (dalla [Fase 1]) e vorranno sicuramente saltarle. Possono premere prima la "S" in modo che il gioco inizi, ma la ragazza continuerà a parlare.

Per evitarlo, possiamo creare un'altra variabile (denominata start), che deve essere impostata su 0 all'inizio. Quindi, dopo le istruzioni della ragazza, la variabile inizia cambierà in 1.

Ora dobbiamo programmare la mela per iniziare solo se l'inizio della variabile è uguale a 1, cosa che gli studenti faranno con l'istruzione "*if statement*". Con questo, gli studenti non saranno in grado di eseguire un gioco prima che la ragazza smetta di parlare.

Un'altra cosa può succedere quando giochiamo di nuovo al gioco. Se interrompiamo il gioco quando abbiamo ad esempio 3 punti, la mela non scomparirà. In questo caso, quando si riavvia il gioco, la mela sarà visibile prima che la ragazza finisca con il dare le istruzioni. Dato che non lo vogliamo, aggiungiamo un codice che nasconde la mela all'inizio del gioco.

Il codice della mela è ora:



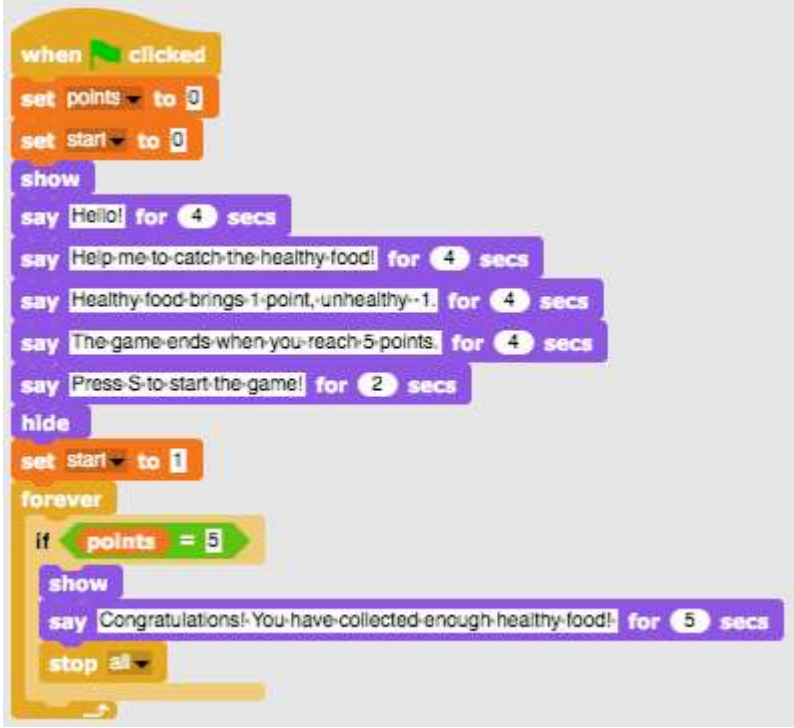
[Fase 9]

Gli studenti possono ora duplicare più volte lo *sprite* di mela e cambiare costume (se lo desiderano). Il codice sarà lo stesso.

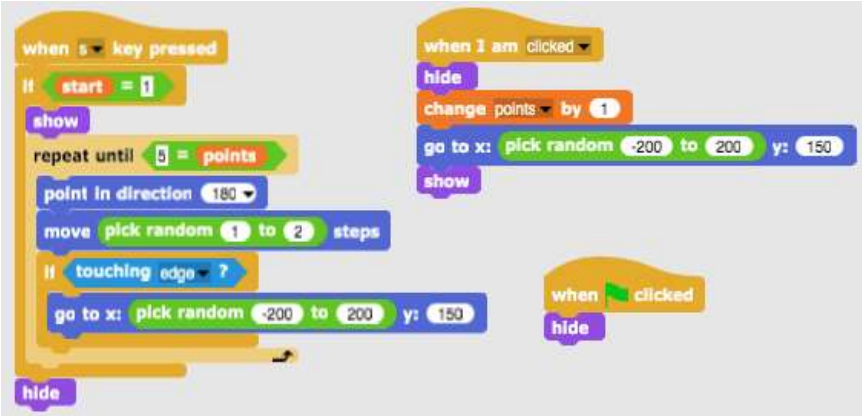
L'unica modifica è con il cibo malsano, dove perderanno 1 punto facendo clic su di esso.

[Codice finale]

Ragazza



Mela



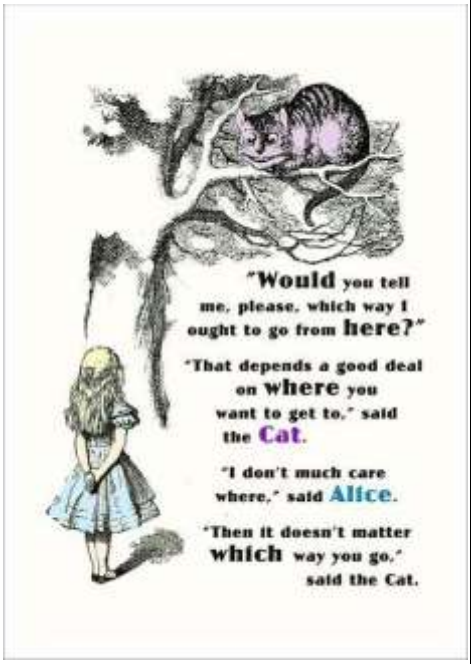
[Compiti aggiuntivi]

Gli studenti possono aggiungere attività in base ai loro desideri o possono seguire le seguenti:

- Cambia il gioco in modo che lo sprite della ciotola prenda cibo.
- Aggiungi un nuovo sprite (una ciotola). Disegnalo, trovalo

	online o usa le foto allegate della ciotola. • Imposta la posizione iniziale della ciotola (ad es. Nella parte inferiore dello schermo) e scrivi un codice per il movimento della ciotola (sinistra e destra, se si desidera anche su e giù). Gli sprite alimentari devono scomparire e riapparire in una posizione casuale toccando la ciotola (e non facendo clic con il mouse sul cibo come prima). • Modifica le regole: il gioco termina quando un giocatore segna 20 punti (vince) o quando raccoglie 3 cibi malsani (perde). • Aggiungi più sprite alimentari per rendere il gioco più interessante. • Cambia il costume della ciotola quando un giocatore segna ad es. 5, 10, 15 punti.
Strumenti e risorse per gli insegnanti	• L'intera attività in Snap!: https://snap.berkeley.edu/project?user=mateja&project=Catching%20healthy%20food • Attività in Snap! con compiti aggiuntivi (possibile soluzione): https://snap.berkeley.edu/project?user=mateja&project=Catching%20healthy%20food%20%2B%20Add.%20Task • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Risorse/materiali per gli studenti	• Attività non completa in Snap!: https://snap.berkeley.edu/project?user=mateja&project=C4G12_Catching%20healthy%20food%20-%20Part • Istruzioni per lo studente (C4G12_InstructionsForStudent.docx) • Immagine: bowl1.png, bowl2.png, bowl3.png, bowl4.png

Scenario di apprendimento 13 - *Storytelling*

Titolo dello scenario di apprendimento	Storytelling
Precedente esperienza di programmazione	Mostrare e nascondere lo *sprite* Usare i condizionali Usare la stringa “dire” (dalla sezione aspetto) Usare aspetta per n__secondi
Risultati di apprendimento	Obiettivi generali di apprendimento: ● spostare gli oggetti e modificare le dimensioni ● trasmissioni ● comporre la struttura del racconto ● modificare lo sfondo delle scene Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente pianifica i dialoghi e le attività degli *sprites* nella storia ● lo studente utilizza l'invio di trasmissioni per il dialogo tra *sprite* ● lo studente sa muovere e cambiare la dimensione degli *sprite* ● lo studente mostra e nasconde gli *sprite* ● lo studente utilizza il codice refactor per lo *sprite* ed lo estende
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: Il coniglio racconta la storia di Alice nel Paese delle Meraviglie. Inizia il racconto con molte frasi sullo sfondo con l'etichetta *Alice nel Paese delle Meraviglie*. La storia di Alice inizia nella foresta. Alice cammina e meravigliandosi afferma "Dove sono?" Per realizzare il movimento di Alice che si sta allontanando gradualmente la sua dimensione si riduce. Alice arriva ad un incrocio e vede il gatto del Cheshire sull’albero. Inizia una conversazione tra di loro. La conversazione è rappresentata nella figura.  **Compiti**: Gli studenti devono sperimentare un breve esempio della storia

	dell'incontro tra Alice e il gatto basato sulla sincronizzazione del dialogo attraverso un blocco di attesa. Poi rivedono una seconda versione della storia usando i messaggi *broadcast*. I comandi di messaggistica sono stati inseriti. Gli studenti completano il codice dei personaggi in base al testo della foto. Il compito è complesso: occorre cambiare l'arredamento del palcoscenico attraverso la trasmissione e spostare Alice nei boschi prima che incontri il gatto. **Obiettivo: Gli studenti impareranno come pianificare lo *storytelling*, come utilizzare i messaggi broadcast per la sincronizzazione delle attività degli *sprite* e dei cambiamenti di scena.**
Durata delle attività	90 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento basato sulla programmazione del gioco, *problem solving*
Didattica	Lavoro individuale / Lavoro in coppia / Discussione
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) 1. L'insegnante e gli studenti condividono la storia di Alice nel Paese delle Meraviglie e viene mostrata l'immagine di Alice che incontra il gatto del Cheshire. La storia di Alice può essere ricreata usando il coding. Gli studenti hanno il compito di avviare il progetto e guardare i codici degli sprite. https://snap.berkeley.edu/project?user=ddureva&project=Alice_1 Discussione: Chi inizia a parlare per primo? Quando viene coinvolta Alice e quando - il gatto? Perché non c'è sincronizzazione nel dialogo dei personaggi? La risposta sta nel calcolo impreciso dei tempi in cui ciascuno dei personaggi "parla" e "la mancanza di un tempo di attesa affinchè un personaggio finisca di rispondere".

I codici vengono commentati e la tabella è completata:

Sprite	Attività	Avvio dall'inizio	Fine	Durata
Coniglio	Dice: Ciao! Hai sentito parlare di Alice e delle sue avventure nel Paese delle Meraviglie? Ora vediamo una delle sue storie.	0	14	14
Alice	Dice: Mi diresti, per favore, per quale strada dovrei andare da qui?	9	21	12
Gatto	Dice: Dipende molto da DOVE vuoi andare.	10	20	10

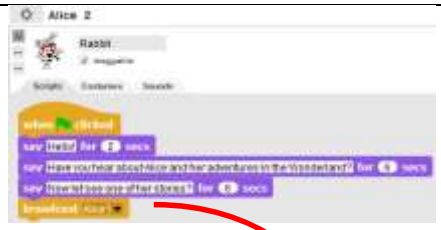
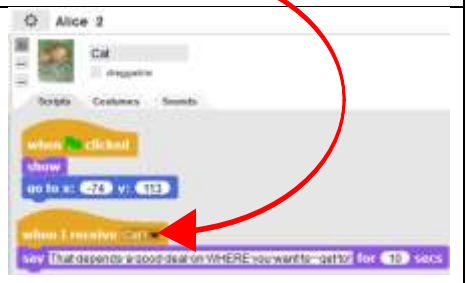
La conclusione è che la sincronizzazione con il blocco *aspetta n… secondi* può portare errori nel comportamento dei personaggi nella narrazione.

2. L'insegnante ha il compito di avviare e rivedere il codice del progetto

https://snap.berkeley.edu/project?user=ddureva&project=Alice_2

Quali sono i comandi sconosciuti finora?

Vengono confrontati i codici di Alice_1 e Alice_2.

Alice_1	Alice_2
	
	
	

3. Vengono introdotti i blocchi per la trasmissione:

Un possibile confronto con gli studenti può avvenire sul fatto che i messaggi di trasmissione vengono inviati a tutti i personaggi, ma possono essere ricevuti solo da alcuni dei personaggi. La trasmissione (broadcast) ed il blocco “aspetta” richiede che tutti i personaggi che hanno ricevuto il messaggio eseguano le loro azioni, e le azioni dello *sprite* che ha inviato il messaggio continuano.

L'insegnante dimostra come assegnare un nome a un messaggio di trasmessione “*broadcast message*” e come usarlo nell’evento “Quando io ricevo …”.

1. 

2. 

3. Inserimento di un nome. OK

Usalo in un evento:

1. 

2. Il messaggio che dovrebbe essere ricevuto dallo *sprite* è selezionato dall'elenco.

3. Il gruppo discute su come completare la storia nella foto. Come nominare i messaggi: ad es. il messaggio dal gatto ad Alice può essere Alice2 e da Alice al gatto - gatto1.

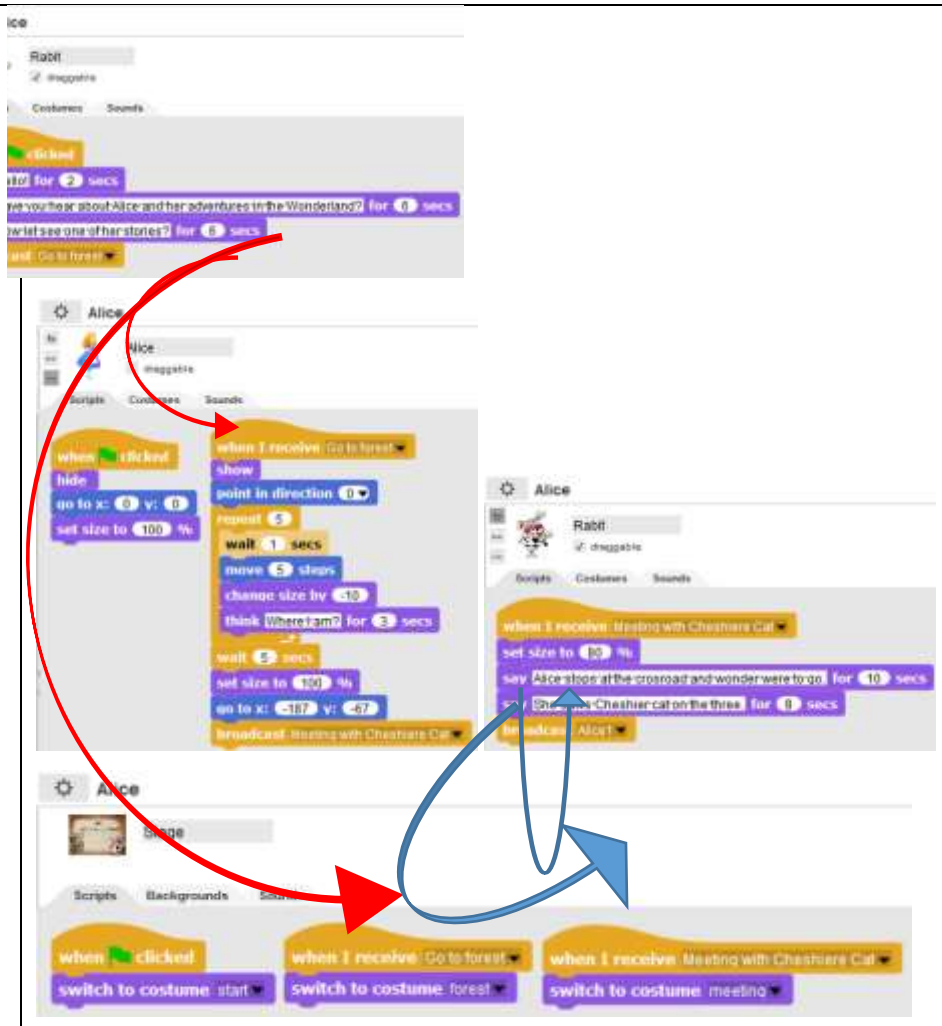
4. Gli studenti completano la storia in coppia.

5. L'insegnante commenta che la narrazione spesso richiede un cambiamento nei costumi di scena. *"Rendiamo la storia di Alice più completa partendo dalla storia del Coniglio su uno sfondo introduttivo, spostandosi successivamente nella foresta dove Alice sta camminando e chiedendosi "Dove sono?". Le sue dimensioni diminuiscono gradualmente man mano che si allontana. Quindi si ritrova ad un bivio e vede il gatto Cheshire. La conversazione tra i due inizia".*

6. L'insegnante mostra il progetto.
   https://snap.berkeley.edu/project?user=ddureva&project=Alice

I cambiamenti di scena e le azioni dei personaggi sono commentati. *"Quando cambia una scena? Quando appare Alice e quali sono le sue azioni? Quando appare il gatto e quali sono le sue azioni?".*

Vengono discusse le scene del progetto Alice_2. Ci sono 3 scene, una già utilizzata. Quale scena usare per iniziare? Cosa si dovrebbe fare per evitare che gli sprite di Alice e il Gatto vengano visualizzati all'inizio? Come cambiare l'arredamento del palcoscenico? Una trasmissione può essere usata per cambiare la scena impostata dal coniglio dopo le sue parole introduttive. Alice appare quando la scena viene cambiata con il messaggio *Vai alla foresta.*

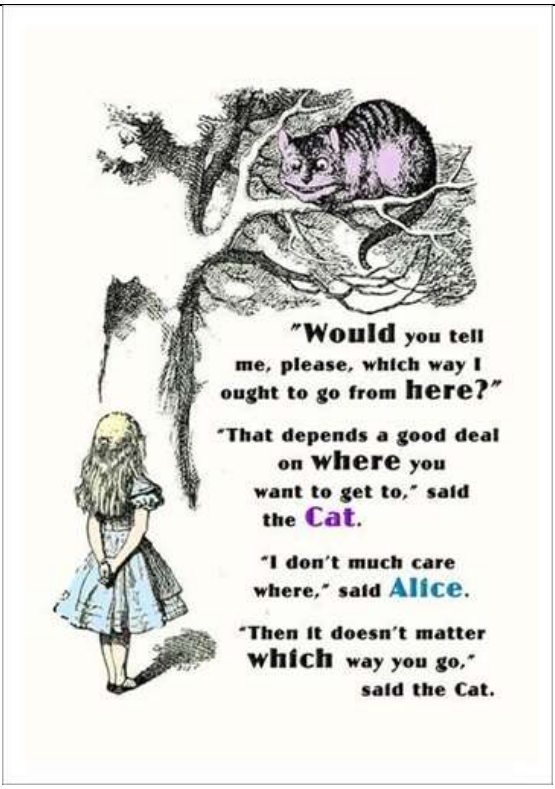


Quando Alice è sul sentiero nel bosco, cammina e "si meraviglia", ma per un maggiore realismo, le sue dimensioni diminuiscono del -10%. Questo viene ripetuto 5 volte usando il ciclo di ripetizione.

Quando raggiunge l'incrocio, la scena cambia con il messaggio "Incontro con il gatto del Cheshire" *"Meeting the Cheshire Cat"*. Questo messaggio viene ricevuto contemporaneamente dal Coniglio che riduce la sua dimenzione dell'80% e continua a raccontare la storia con le sue dimensioni ridotte. In questa fase lo *sprite* di gatto non viene mostrato perché è presente come parte della scena. Appare sul messaggio Cat1.

L'insegnante può spiegare che il gatto è stato tagliato dalla scena usando un editor grafico esterno.

Dopo il rilascio del messaggio del Coniglio, la storia di *Alice 1* continua

	come è stato fatto nel progetto *Alice 2*. 3. L'insegnante commenta che per raccontare una storia bisogna prima inventare una trama. Una tabella aggiuntiva può essere utilizzata per descrivere lo scenario della storia. (Appendice 1). A discrezione dell'insegnante, può essere data la sceneggiatura completa o parzialmente completata ed in quest'ultimo caso gli studenti possono completarla guidati dall'illustrazione. 4. Gli studenti hanno il compito di descrivere gli scenari esaminati e di completare la storia del progetto Alice_2 in coppia.
Strumenti e risorse per gli insegnanti	L'intera attività in Snap!: https://snap.berkeley.edu/project?user=ddureva&project=Alice
Risorse/materiali per gli studenti	 • https://snap.berkeley.edu/project?user=ddureva&project=Alice_1 • https://snap.berkeley.edu/project?user=ddureva&project=Alice_2 • Istruzioni per lo studente (C4G13_InstructionsForStudent.docx)

Appendice 1. Trame/Scenari

Nome	Sfondo	Azione	Note
1. Inizio		La storia inizia con la scena (quando si clicca sulla bandiera verde)	In questo contesto, il Coniglio introduce la storia.
2. Foresta		Lo scenario appare quando il coniglio completa la sua introduzione (è stato inviato un messaggio *Vai alla foresta*)	Su questo sfondo, Alice appare posizionata al centro della scena. Comincia a muoversi, chiedendosi "*Dove sono?*". Lo *sprite* riduce gradualmente le sue dimensioni 5 volte del 10%. Quando raggiunge la fine del percorso (ad un incrocio), la scena cambia in "*Meeting*". (Alice invia un messaggio Incontro con il gatto Cheshire "*broadcast Meeting with Cheshire Cat*").
3. Riunione		Appare quando viene ricevuto il messaggio di Alice "Incontro con il gatto Cheshire".	Qui Alice e il gatto fanno parte dello sfondo. Per usare lo *sprite* di Alice, prima del messaggio, posizionalo in modo da coprire la sua immagine sulla scena. Lo *sprite* del gatto appare in una fase successiva. Mentre la scena cambia, il Coniglio continua a raccontare la storia. Successivamente si svolge una

			conversazione tra Alice e il gatto del Cheshire.

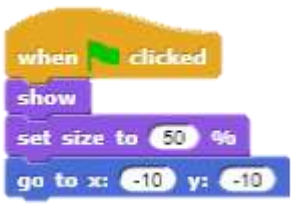
Sprite

Sprite	Azioni	Sfondo della scena
	All'inizio: Dice: Ciao! (per 2 secondi) Dice: Hai sentito parlare di Alice e delle sue avventure nel Paese delle Meraviglie? (per 6 secondi) Dice: Ora vediamo una delle sue storie! (per 6 secondi) Invia il messaggio *Vai alla foresta*.	
	All'inizio: Si nasconde dalla scena; è al centro e al 100% della sua dimensione, pronta per essere visualizzata sullo nuovo sfondo.	
	All'inizio: Si nasconde dalla scena; posizionato su x: -74, y: 113 (Le posizioni sono predeterminate dopo che lo *sprite* gatto è stato impostato sulla fase Riunione "*Meeting*".)	
	Riceve il messaggio *Vai alla foresta*: Lo *sprite* si mostra sulla scena. Ripeti 5 volte: attesa di 1 secondo; muovi 5 passi; riduci la dimensione (cambia a -10); si meraviglia: *Dove sono?* Preparazione per la prossima scena: in attesa di 5 sec.; ripristinare le dimensioni dello *sprite* (modifica del 100%) e posizionarlo su x: -187, y: -67 Invia il messaggio: *Incontra il gatto Cheshire*.	
	Nessuna azione. Diventa appena visibile dalla scena precedente.	

	Riceve il messaggio: *Incontra il gatto Cheshire*. Riduci la dimensione all'80% Dice: *"Alice si ferma all'incrocio e si meraviglia su dove andare."* (per 10 secondi). Dice, *"Ha visto il gatto Cheshire sull'albero"* (per 8 secondi) Invia un messaggio *Alice1*	
	Riceve il messaggio *Alice1*. Si sposta in primo piano. Lei dice: *"Ciao!"* (per 2 secondi) Dice: *"Vorresti dirmi per piacere, da che parte dovrei andare da qui!"* (per 10 secondi). Invia un messaggio di trasmissione "*broadcast"* al gatto Ca*t1*.	
	Riceve il messaggio *Cat1*. Lo *sprite* si mostra sulla scena. Dice: *"Dipende molto da DOVE vuoi arrivare!"* (per 10 secondi). Invia un messaggio *Alice2*.	
	Riceve il messaggio *Alice 2*. Dice: …… Invia un messaggio *Cat2*.	
	Riceve il messaggio *Cat2*. Dice: …… Invia un messaggio *Rabbit1*.	
	Riceve il messaggio *Rabbit1*. Dice: "*Qual è la morale della storia?*" (per 8 secondi) Dice: "*Per sapere da che parte andare, bisogna prima determinare il proprio obiettivo*."	

Scenario di apprendimento 14 - Disegnare

Learning Scenario Title	Disegnare
Precedente esperienza di programmazione	Aggiungere uno *sprite* Usare il punto di direzione Usare le variabili per il conteggio dei punti Usare il ciclo di ripetizione Usare i condizionali
Risultati di apprendimento	Obiettivi generali di apprendimento: ● variabili ● condizionali ● cicli ● punto di direzione ● operatori Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente usa la penna per disegnare ● lo studente usa i cicli per disegnare ● lo studente cambia i valori di una variabile quando disegna ● lo studente usa il punto di direzione per disegnare oggetti sulla scena ● lo studente usa la trasmissione per controllare lo *sprite* ● lo studente usa i condizionali per cambiare la scena ● lo studente usa gli operatori > per cambiare la scena
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: Il clima è molto cambiato, l'aria è fortemente inquinata a causa dell'industria. Gli alberi devono essere piantati per migliorare la qualità dell'aria! **Compiti**: Per migliorare la qualità dell'aria, gli studenti devono programmare lo sprite per disegnare due tipi di alberi diversi: pino e quercia e i pulsanti che simboleggiano questi tipi di alberi. Quando si clicca su un pulsante, viene disegnato un tipo di albero specifico. **Obiettivo: Lo studente imparerà a disegnare in Snap!, cambiare il colore e lo spessore della penna e ad usare variabili e condizioni che causano un nuovo evento.**
Durata delle	45 minuti

attività	
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento basato sulla programmazione del gioco, *problem solving*
Didattica	Lavoro individuale / Lavoro in coppia
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) All'inizio del gioco, vengono mostrati sulla scena un'industria che causa cambiamenti climatici e una variabile che mostra la qualità dell'aria. Per migliorare la qualità dell'aria devono essere piantati gli alberi. Due differenti tipi di alberi devo essere piantati: pino e quercia. Quando viene disegnato un pino, l'aria migliora di 3 unità mentre disegnando una quercia l'aria migliora di 2 unità. Quando la qualità dell'aria raggiunge le 10 unità, lo sfondo della scena cambia in un prato. [Fase 1] Occorre scaricare l'archivio con materiali e scompattalo. Da tali materiali gli studenti impostano l'immagine di sfondo dell'industria inquinante l'aria. Inoltre, si aggiunge un nuovo *sprite* matita "*pencil a*" dagli *sprite* offerti. Poiché lo *sprite* è troppo grande, dovrebbe essere ridotto al 50%. È inoltre necessario specificare la posizione iniziale della matita (coordinate), ad es., X = -10, y = -10.  [Fase 2] Gli studenti aggiungono un nuovo *sprite* - pino. La sua dimensione

dovrebbe essere ridotta del 20%. Un messaggio "pino" dovrebbe essere trasmesso quando si clicca sullo *sprite*.

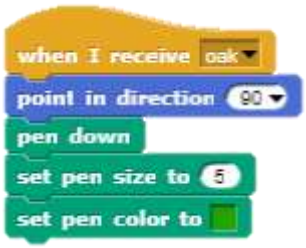
Deseleziona il pulsante "trascinabile" così lo *sprite* non può essere spostato.

Ripeti il processo e aggiungi la *sprite* quercia "*tree1*" dagli *sprite* offerti. Un messaggio "quercia" dovrebbe essere trasmesso quando si clicca su questo *sprite*. Deseleziona il pulsante "trascinabile" così lo *sprite* non può essere spostato.

[Fase 3]

Lo sprite della matita dovrebbe ricevere i messaggi "quercia" e "pino" e disegnare l'albero appropriato in risposta al messaggio.

Per prima cosa, segna lo *sprite* matita e aggiungi il codice che consentirà di disegnare la quercia usando la penna quando lo *sprite* riceve il messaggio "quercia". Un punto in direzione dovrebbe essere impostato su 90 per mantenere la chioma rotonda, la penna dovrebbe essere abbassata e il suo spessore e colore dovrebbero essere impostati, ad es. impostare la dimensione della penna su 5 e scegliere una sfumatura di verde.



Per disegnare la chioma della quercia, sposta lo *sprite* di 1 passo e gira 3 gradi a sinistra dopo ogni passo.

Questo movimento dovrebbe ripetersi 120 volte.

Una volta che la chioma è completata, dovrebbe essere disegnato il tronco. Lo sprite matita deve essere spostato al centro del cerchio disegnato, di -3 passi e il colore della penna diventa marrone.

Per disegnare il tronco, lo *sprite* dovrebbe essere mosso di 10 passi, ruotato a destra di 90 gradi, dopo 1 secondo di 20 passi e di nuovo ruotato a destra di 90 gradi.

Questa parte è ripetuta due volte.

Al termine del disegno, è necessario sollevare la penna in modo che non tracci la linea quando si sposta lo *sprite* della matita.

[Fase 4]

Allo stesso modo, è necessario aggiungere il codice allo *sprite* a matita per disegnare i pini. Il pino dovrebbe essere disegnato quando *sprite* riceve il messaggio "pino".

Un punto in direzione dovrebbe essere impostato su 200 per disegnare la chioma a forma di triangolo, la penna dovrebbe essere abbassata e il suo spessore e colore dovrebbero essere impostati.

Per disegnare la chioma del pino, occorre muovere lo *sprite* di 82 passi, ruotare a sinistra di 110 gradi poi muovere di 60 passi, ruotar a sinistra di 110 gradi ed infine muovere di 82 passi.

Per rendere visibile il processo di disegno, è necessario consentire alla penna di attendere 0,5 secondi dopo ogni giro a sinistra.

Dopo la chioma dovrebbe essere disegnato il tronco. Occorre sollevare la penna e muovere di -80 passi per raggiungere il fondo della chioma.

Affinché il tronco sia nella posizione corretta, occorre girare la matita a sinistra di 100 gradi, porre il punto in direzione 90 e muovere di -40 passi.

Per iniziare a disegnare, abbassa la matita e imposta il colore sul marrone.

Occorre spostare 15 passi poi ruotare di 90 gradi a destra. Successivamente, la matita dovrebbe attendere 0,5 secondi per vedere il processo di disegno.

Questo movimento dovrebbe essere ripetuto 4 volte.

Infine, è necessario sollevare la penna in modo che lo *sprite* non lasci traccia durante il movimento successivo.

[Fase 5]

Occorre aggiungere un nuovo *sprite* segno - X che sarà usato per cancellare tutti gli alberi disegnati. Le sue dimensioni dovrebbero essere impostate sul 50%. Quando si clicca sullo *sprite*, tutto viene cancellato. Inoltre, deselezionando il pulsante "trascinabile", lo *sprite* non può essere spostato.

[Fase 6]

Si creare una nuova variabile "aria pulita" per mostrare l'attuale qualità dell'aria. Occorre impostare il valore iniziale su -10 e mostrare la variabile sulla scena.

Ogni volta che viene disegnato un pino l'aria migliora di 3 unità, quindi aggiungi il blocco allo *sprite* di pino che cambierà il valore della variabile "aria pulita" di 3 ogni volta che il pino viene cliccato.

Ogni volta che una quercia viene disegnata l'aria migliora di 2 unità, quindi aggiungi il blocco allo *sprite* quercia che cambierà il valore della variabile "aria pulita" di 2 ogni volta che la quercia viene cliccata.

[Fase 7]

Quando la variabile "aria pulita" arriva a 10, la scena dovrebbe cambiare a prato. Pertanto, dai materiali scaricati aggiungi una nuova "erba" di sfondo per la scena (lo sfondo proviene dai materiali scaricati).

Aggiungi allo sprite della matita il cappello "Quando" dalla sezione "Controllo".

Poi aggiungi l'operatore >.

Definisci che lo *sprite* trasmette il messaggio "erba" quando la variabile "aria pulita" è maggiore di 10.

Aggiungi il codice alla scena per cambiare il costume a "erba" quando ha ricevuto il messaggio "erba".

[Fase 8]

Occorre consentire alla scena di tornare al costume "industria" quando tutti gli alberi vengono eliminati o quando il gioco viene riavviato. Per fare ciò, si aggiunge il blocco per inviare un messaggio "industria" al blocco "Quando clicco" nello *sprite* X.

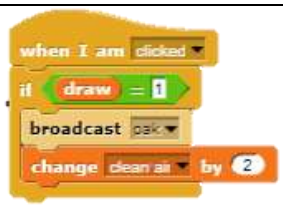
Si aggiunge il codice alla scena in modo tale da permettere il cambio del costume a “industria” quando il messaggio è ricevuto.

Per mostrare solo l'immagine dell'industria all'inizio del gioco, aggiungi per la scena un interruttore “*switch*” al costume “industria” e il blocco “pulisci” "Quando si clicca la bandiera verde". Il blocco “pulisci” vuol dire che, al riavvio del gioco, si cancella tutto ciò che era stato disegnato.

[Fase 9]

Per assicurarsi che ogni albero sia disegnato completamente, aggiungi una nuova variabile "disegna". Quando si disegna la quercia, il valore della variabile, prima che la penna inizi a disegnare, deve essere impostato su 0. Dopo che la penna termina il disegno, il valore della variabile deve essere impostato su 1.

Per lo *sprite* quercia occorre aggiungere una condizione al blocco "Quando sono cliccato" al fine di abilitare la trasmissione del messaggio "quercia" solo se il valore della variabile "disegno" è uguale a 1.



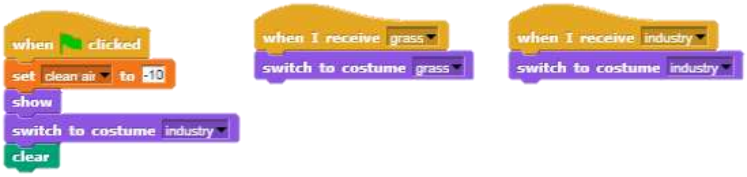
Allo stesso modo occorre aggiungere la medesima condizione per lo *sprite* pino per abilitare la trasmissione del messaggio "pino" solo se il valore della variabile "disegno" è uguale a 1. A causa di queste condizioni, il disegno di un albero non può iniziare prima della fine del disegno di un altro albero.

[Codice finale]

Pino

Quercia

X

	Matita  Scena 
Strumenti e risorse per gli insegnanti	Progetto in Snap!: https://snap.berkeley.edu/project?user=ifrankovic&project=Improve%20the%20Climate (9.1.2020)
Risorse/materiali per gli studenti	• Modello in Snap!: https://snap.berkeley.edu/ (9.1.2020) • Istruzioni per lo studente (C4G14_InstructionsForStudent.docx) • Immagini: grass.png, industry.png

Scenario di apprendimento 15 - Catturare il topo

Titolo dello scenario di apprendimento	Catturare il topo
Precedente esperienza di programmazione	• lo studente è in grado di aggiungere uno sfondo • lo studente è in grado di aggiungere un nuovo *sprite* • lo studente è in grado di aggiungere un nuovo suono • lo studente sa come far dire qualcosa allo *sprite* • lo studente sa come cambiare il costume dello sprite per fare un'animazione • lo studente è in grado di impostare un movimento di un oggetto con i tasti freccia utilizzando gli eventi e tiene conto delle restrizioni • lo studente è in grado di differenziare due stati e sa come esprimerli con espressioni logiche • lo studente sa come usare i condizionali
Risultati di apprendimento	Obiettivi generali di apprendimento: • il ciclo per sempre • numeri casuali • contatore; • timer. Obiettivi specifici di apprendimento orientati al pensiero algoritmico: • lo studente sa utilizzare il il ciclo per sempre per muovere gli *sprite* • lo studente usa i numeri casuali per determinare la posizione degli *sprite*, per muovere e ruotare gli *sprite* con passi casuali • lo studente imposta il contatore per conteggiare i topi catturati e usa il valore finale per riepilogare il risultato ottenuto • lo studente usa il *timer* per determinare la fine del gioco
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: si programma un gioco in cui il giocatore (il gatto) dovrà catturare il topo. **Compito**: si programma l'attività in cui il gatto catturerà il topo. Il gatto verrà spostato da un giocatore con i tasti freccia e il topo si sposterà in modo casuale. Quando il gatto toccherà il topo, questo si nasconderà e apparirà in una posizione casuale. Dobbiamo anche avere un contatore che conterà il numero di volte in cui il gatto ha catturato il topo. Dobbiamo anche avere bisogno di un timer per finire il gioco. Dopo l'attività, si deve riassumere il successo del giocatore, una ragazza deve dire quante volte il giocatore ha catturato il topo.

	Scopo: **Agli studenti verrà illustrato il concetto di assegnazione di più valori casuali variabili. Impareranno come usare il blocco Operatori / scegli casuali da [x]a[y] "*Operators/pick random[x]to[y]*".**
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento collaborative, *problem solving*, apprendimento basato sulla programmazione del gioco
Didattica	Lezione frontale Lavoro di coppia / lavoro di gruppo
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) **Motivazione-Introduzione** Motiviamo gli studenti mostrando il gioco. Discutiamo con loro su come avrebbero iniziato a programmare questo gioco. Insieme agli studenti, determiniamo la sequenza dei passaggi, ad esempio: 1. scegli lo sfondo ed aggiungi gli *sprite* 2. programma il gatto che si sposta con i tasti freccia 3. programma il topo che si muove casualmente 4. programma il topo affinché si nasconda (e appaia in un luogo casuale) quando il gatto lo tocca 5. programma il contatore; 6. aggiungi il *timer* e determina la fine del gioco; 7. aggiungi una ragazza e programmala in modo tale che riassuma il risultato ottenuto dal giocatore 8. programma la ragazza a salta quando tocca il topo 9. aggiungi il suono del gatto e del topo Gli studenti possono essere aiutati con questi passaggi o stabilire le proprie regole di gioco.

Introduciamo l'operatore per l'assegnazione del valore casuale.

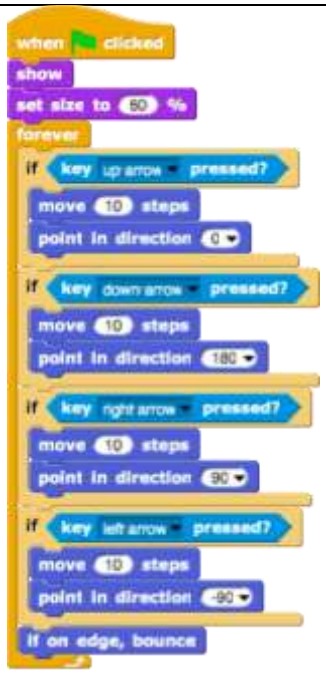
Gli studenti programmano i seguenti compiti in coppia/gruppo con il supporto dell'insegnante.

[Fase 1]

Il primo passo è determinare lo sfondo del gioco. Gli studenti cercano immagini online gratis. Successivamente, aggiungono nuovi *sprite*: il gatto e il topo.

[Fase 2]

Gli studenti programmano il gatto per muoversi con i tasti freccia. Qui devono determinare cosa succede se il gatto è sul bordo.



[Fase 3]

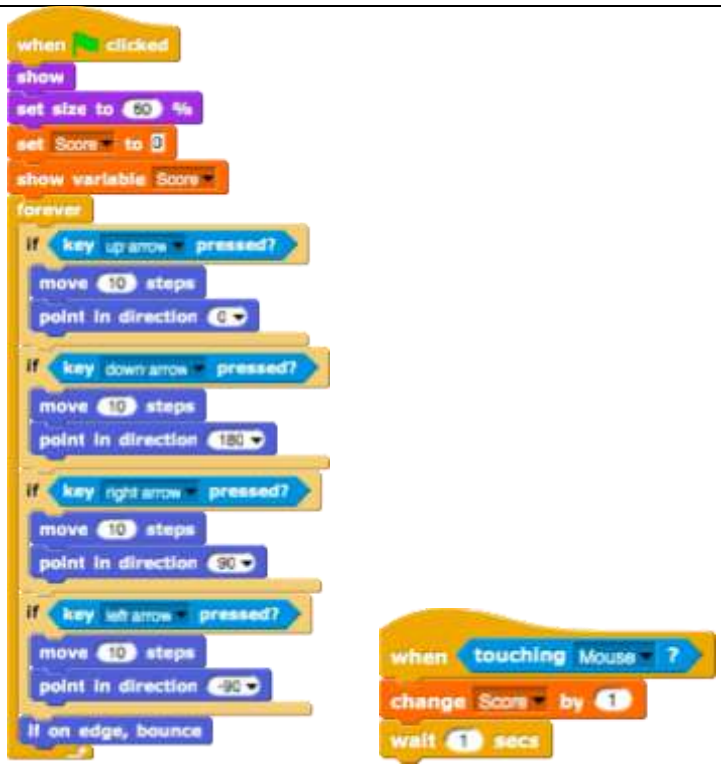
Gli studenti devono programmare il topo per spostarsi in modo casuale. In questo caso, l'idea è che il topo, in un ciclo infinito, compia un numero casuale di passaggi e giri. Gli studenti lo fanno con i blocchi di movimento/spostamento [x] *"Motion/move[x]steps"* e blocco di movimento/ giro [x] *"Motion/turn[x]degrees"* in cui inseriscono l'operatore di selezione casuale *[x]in[y]*.

[Fase 4]

Il prossimo passo è programmare il topo per nasconderlo quando tocca il gatto. L'idea è che il topo si nasconda e appaia in una posizione casuale.

In questo caso, il gioco non termina alla prima presa del topo. Gli studenti possono aggiungere le proprie regole qui. In ogni caso, devono usare

	l'operatore "*pick random[x]to[y]*".  [Fase 5] Nel caso in cui vogliamo sapere il numero di volte in cui il topo è stato catturato, dobbiamo aggiungere un contatore. Gli studenti creano una nuova variabile: segnarla e aggiungerla al codice del gatto. Il punteggio all'inizio del gioco deve essere sempre zero. Gli studenti lo fanno con il blocco Variabili/impostare [variabile] su [x] "*Variables/set[variable]to[x]*". Se vogliamo che il punteggio venga mostrato al giocatore del gioco, gli studenti devono aggiungere il blocco [mostra variabile]. Quindi gli studenti aggiungono un nuovo blocco di controllo/quando "*Controllo/quando"* per verificare se il gatto tocca il topo. Se il gatto tocca il topo, il risultato viene aumentato di 1 Variabili/cambia [punteggio] di [x] "*Variables/change[score]by[x]"*.



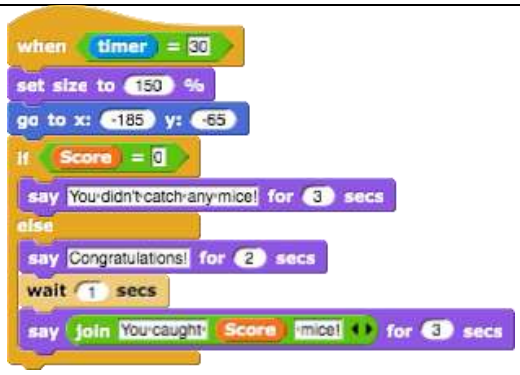
[Fase 6]

Gli studenti determinano quando finisce il gioco. Lo fanno aggiungendo il *timer*. Dopo un po' di tempo (ad es. 30 secondi) il topo e il gatto scompaiono, la variabile punteggio viene nascosta e il gioco termina.

Gli studenti devono aggiungere questi blocchi allo *script* del gatto e del topo.

[Fase 7]

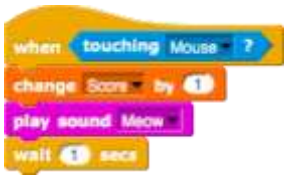
Gli studenti devono programmare la ragazza per riassumere il successo del giocatore. Se il giocatore non cattura alcun topo, la ragazza dice: "*Non hai catturato alcun topo!*". Altrimenti dice: "*Congratulazioni! Hai catturato X topi!*"



[Compiti aggiuntivi]

Gli studenti possono aggiungere qualsiasi elemento al loro gioco. Ad esempio, la ragazza che salta ogni volta che tocca un topo.

Gli studenti possono aggiungere suoni. Ad esempio, aggiungono il suono del gatto. Il suono viene riprodotto quando viene catturato il topo.



Riflessione e valutazione

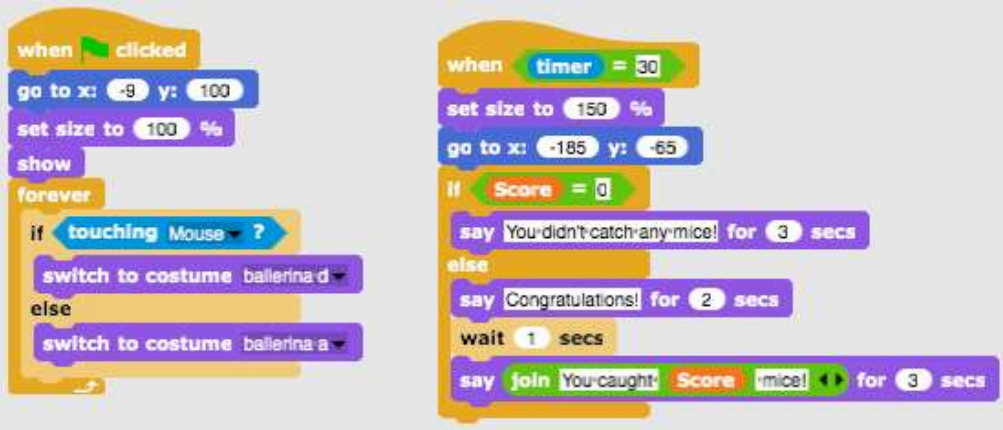
Gli student sistemano il codice:

- il topo si muove da 20 a 60 passi per sempre
- il topo va al luogo x = 100 quando tocca il gatto
- il topo si gira di 90 gradi per sempre

[Codice finale]

Il topo

Il gatto

	La ragazza  *Lo sfondo* 
Strumenti e risorse per gli insegnanti	• L'intera attività in in Snap!: https://snap.berkeley.edu/project?user=tadeja&project=Catch%20the%20mouse • Sito web per le immagini: https://pixabay.com/ • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Risorse/materiali per gli studenti	• Modello in Snap!: https://snap.berkeley.edu/project?user=tadeja&project=Catch%20the%20mouse_0 • Sito web per le immagini: https://pixabay.com/ • Istruzioni per lo studente (C4G15_InstructionsForStudent.docx)

Scenario di apprendimento 16 - Acquistare cibo per un picnic

Titolo dello scenario di apprendimento	Acquistare cibo per un picnic
Precedente esperienza di programmazione	Aggiungere il testo allo *sprite* Mostrare e nascondere gli *sprite* Usare gli operatori Usare le variabili Usare la concatenazione di stringhe Usare i condizionali
Risultati di apprendimento	Obiettivi generali di apprendimento: ● variabili ● condizionali ● operatori Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente usa le variabili per impostare il prezzo dei vari *sprite* ● lo studente cambia il valore delle variabili poiché il budget cambia quando il giocatore acquista cibo ● lo studente usa l'istruzione "*if*" per verificare la disponibilità del denaro ● lo studente usa gli operatori per unire testo - valore delle variabili – testo ● lo studente usa gli operatori per comparare i prezzi e la disponibilità di denaro ● lo studente usa gli operatori (sottrazione) per modificare il valore delle variabili
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: La ragazza va a fare un picnic e ha bisogno di aiuto per comprare del cibo. Ha 15 euro e non può spendere di più. Quando acquista qualcosa, il valore del budget cambia. Se il suo budget è troppo basso, non può comprare gli alimenti che desidera. **Compiti**: Gli studenti devono programmare tre diversi *sprite* - una ragazza, cibo (che si può duplicare con lievi modifiche) e pulsante di fine. La ragazza dà le istruzioni, dice quanto denaro ha il giocatore e alla fine (cliccando sul pulsante fine) dice quanti prodotti sani e non sani ha comprato. Il cibo indica il suo prezzo quando è cliccato con il

	mouse. Se il giocatore ha abbastanza denaro, il valore del *budget* cambia. Se non si ha denaro sufficiente il cibo non può essere acquistato. **Obiettivo: gli studenti impareranno come lavorare con le variabili - impostare differenti valori iniziali, usare i condizionali per comparare il valore delle variabili, cambiare il valore delle variabili, usare le variabili per conteggiare il cibo non sano. Inoltre, ripeteranno l'aggiunta di testo, l'unione di testi e l'istruzione "*if*".**
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento basato sulla programmazione del gioco, *problem solving*
Didattica	Lavoro individuale / lavoro in coppia
Teaching summary	(Motivazione-introduzione, attuazione, riflessione e valutazione) La ragazza è in una drogheria a comprare cibo per un picnic. Ha 15 euro. Può vedere il prezzo del cibo quando il puntatore del mouse lo posiziona e può acquistarlo facendo clic sul cibo selezionato. Può comprare cibo solo fino a quando ha abbastanza soldi. Il giocatore acquista facendo clic sul pulsante *Fine*, e la ragazza dice quanti prodotti sani e non salutari ha acquistato. 

[Fase 1]

Questa attività è intesa come lavoro individuale o lavoro in coppia. L'insegnante fornisce alcuni indizi, spiega alcune parti più difficili e aiuta quando è necessario.

Gli studenti scelgono lo sfondo e aggiungono uno *sprite* principale, ad es. una ragazza. La ragazza dà alcune istruzioni all'inizio, ad es.:

[Fase 2]

In questo gioco abbiamo bisogno di alcune variabili:

- *budget*, per impostare il denaro disponibile,
- *finish*, per terminare il gioco,
- *healthy_food*, per calcolare quanti alimenti salutari ha acquistato il giocatore,
- *unhealthy_food*, per calcolare quanti alimenti non salutari ha acquistato il giocatore,
- una variabile per ogni cibo, ad es., *acqua_prezzo,* per impostare il prezzo di ogni alimento.

All'inizio, la variabile *budget* è impostata ad es., 15 euro. Altre tre variabili sono impostate su 0. Questo codice può essere aggiunto prima del codice della ragazza nella [Fase 1].

[Fase 3]

Gli studenti aggiungono uno *sprite* (cibo) e scelgono il suo costume.

Il codice alimentare (acqua) richiede tre eventi di controllo:

a) *quando la bandiera verde è cliccata*: mostra e imposta il presso del cibo.
Lascia che il prezzo della variabile sia ragionevolmente determinato (ovviamente, non 0, ma maggiore di 1).

b) Il blocco *When mouse-entered:* per dire al giocatore quanto costa il prodotto.
Gli studenti possono usare il blocco *"Looks – thinking"*
con la stringa unione per il testo - valore della variabile - testo, ad es.:

c) Il blocco *When clicked*: gli studenti devono fare una piccola riflessone.
1) in quale caso il giocatore può acquistare il prodotto oppure no?
2) cosa accade con il *budget* se compra il cibo?
3) come contiamo i prodotti acquistati?
4) cosa succede con il cibo sullo scaffale?

Il giocatore può acquistare il prodotto se ha abbastanza denaro. Così gli studenti devono comparare due variabili: il *budget* ed il prezzo del bene. Se l'alimento costa più di quello che ha, non può comprarlo. Gli studenti possono aggiungere del testo per dire al giocatore che non può acquistare questo prodotto.

Se il giocatore ha 15 euro e compra l'alimento per 4 euro egli ha ora a disposizione 15 – 4 = 11 euro. In tal modo il valore del *budget* è ora: precedente valore del *budget* – *acqua_prezzo*.
Gli studenti possono aggiungere il testo qui.

Il conteggio del numero di prodotti acquistati verrà ottenuto modificando la variabile "*healthy_food*" di 1.

Quando il cibo è cliccato, scomparirà.

Una possibile soluzione é:

[Fase 4]

Per avere più cibo sugli scaffali, gli studenti possono duplicare lo *sprite* del cibo. Diciamo che il secondo cibo sarà una torta. Il codice dalla [Fase 3] necessita quindi di alcune modifiche. Gli studenti devono:

- cambiare il costume
- realizzare una nuova variabile: *torta_prezzo*
- impostare il valore della *torta_prezzo*
- cambiare il codice di ogni blocco *acqua_prezzo* con *torta_prezzo*
- cambiare la risposta sull'acquisto della torta

- sostituisci *cambia salutare_cibo a 1* in *cambia non salutare_cibo a 1*

Ad es., quando è cliccato il codice per la torta potrebbe essere:

[Fase 5]

Quando il giocatore termina i suoi acquisti, fa clic sul pulsante Fine. Per dire al programma che il giocatore ha fatto clic sul pulsante (termina di comprare cibo), inviamo il messaggio.

[Fase 6]

Alla fine torniamo allo *sprite* della ragazza.

Quando il giocatore finisce di fare acquisti, vogliamo che la ragazza gli dica quanti prodotti sani e non salutari ha comprato.

Quando il giocatore fa clic sul pulsante *Fine*, viene inviato un messaggio di *Fine*.

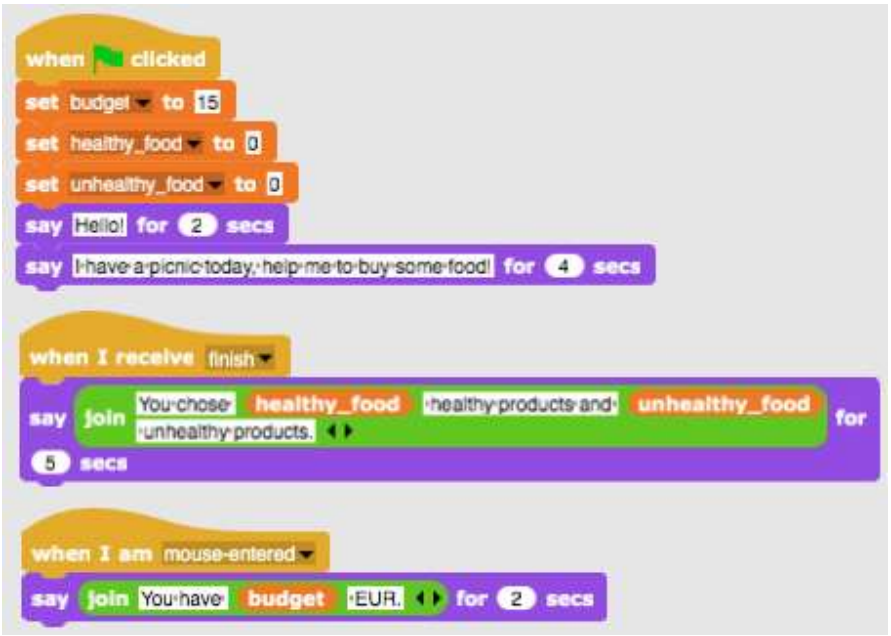
Quando la ragazza riceve il messaggio di *Fine*, dice, ad es. "Hai scelto X prodotti sani e Y prodotti non salutari".

[Fase 7]

In qualsiasi momento del gioco, il giocatore può controllare il proprio budget utilizzando il *mouse-enter* sulla ragazza. Ad esempio, può dire / pensare qualcosa del genere:

[Codice finale]

Ragazza



	Cibo  Pulsante di fine  [Compiti aggiuntivi] Gli studenti possono aggiungere attività in base ai loro desideri o possono seguire le seguenti: • Cambia il gioco in modo da poter acquistare ogni cibo 3 volte. • Dai più soldi al giocatore all'inizio. • Alla fine la ragazza dice anche quanti prodotti hai comprato. Per esempio. "Hai comprato 2x anguria, 1x uva, 2x patatine fritte".
Strumenti e risorse per gli insegnanti	• L'intera attività in Snap!: https://snap.berkeley.edu/project?user=mateja&project=Buying%20food%20for%20a%20picnic

	● Attività in Snap! con compiti aggiuntivi (soluzione possibile): https://snap.berkeley.edu/project?user=mateja&project=Buying%20food%20for%20a%20picnic%20%2B%20Add.%20Task ● Lajovic, S. (2011). *Scratch. Nauči se programirati in postani računalniški maček.* Ljubljana: Pasadena. ● Vorderman, C. (2017). *Računalniško programiranje za otroke.* Ljubljana: MK.
Risorse/materiali per gli studenti	Istruzioni per lo studente (C4G16_InstructionsForStudent.docx)

Scenario di apprendimento 17 - Operazioni

Titolo dello scenario di apprendimento	Operazioni
Precedente esperienza di programmazione	Usare le variabili per contare i punti e scegliere il costume della scena e dello *sprite* Usare i numberi casuali per scegliere le decorazioni sceniche e costumi per lo *sprite* Usare il ciclo continuo Usare i condizionali Usare le operazioni per la comparazione Usare il rilevamento *"sensing"* per il dialogo (chiedi ... e aspetta) Usare gli eventi di trasmissione
Risultati di apprendimento	Obiettivi generali di apprendimento: ● variabili ● condizionali ● ciclo ● blocchi di rilevamento ● eventi di trasmissione Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente usa variabili per il conteggio dei punti e per i costumi della scena e del mantenimento dello *sprite* ● lo studente usa le variabili per il conteggio dei punti ● lo studente imposta le variabili per il conteggio dei punti ● lo studente usa i condizionali e le operazioni logiche ● lo studente usa l'evento di trasmissione per cambiare *sprite* e calcolare il risultato finale
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: Controlliamo durante una partita se il giocatore ha imparato le operazioni aritmetiche in Snap!. Le regole sono le seguenti: dieci volte un'operazione aritmetica con il primo operando di 6 viene selezionata in modo casuale e il secondo operando viene selezionato in modo casuale per essere un numero compreso tra 1 e 3. Il giocatore deve inserire la risposta corretta. Vengono contate le risposte giuste e sbagliate. Alla fine del gioco viene riportato il risultato corretto.

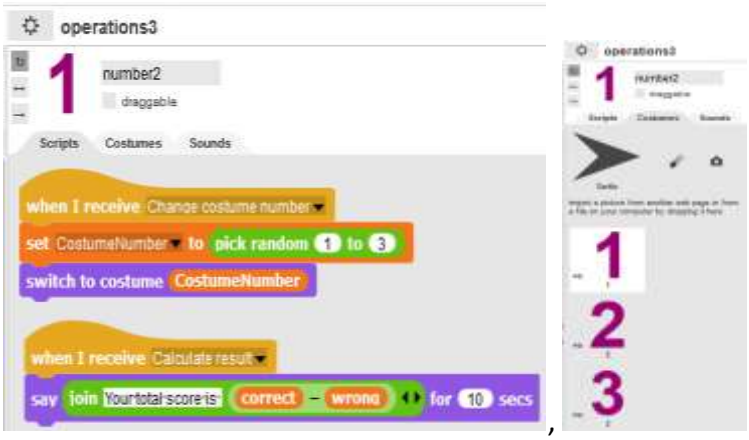
	Compito: gli studenti devono definire lo scenario della scena e il costume dello *sprite*; per pianificare le variabili richieste, determinare quali blocchi sono necessari. Alla fine devono creare i codici per lo stage e lo sprite. Compiti aggiuntivi potrebbero essere: ● Per assegnare lo *sprite*, a seconda del risultato, dire: "Buono per te!" o "Non conosci ancora bene le operazioni aritmetiche in Snap!" **Obiettivo: gli studenti miglioreranno le loro conoscenze acquisite in precedenza su variabili, numeri casuali, cicli, trasmissioni.**
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo (discussioni, esperimento con un gioco precedentemente preparato), apprendimento basato sulla programmazione del gioco, *problem-solving*
Didattica	Lavoro individuale / lavoro in coppia / lezione frontale con l'intera classe
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) 1. L'insegnante pone il problema relativo alla necessità di un gioco per determinare se le operazioni aritmetiche in Snap! sono stati padroneggiate e dimostra il progetto. https://snap.berkeley.edu/project?user=ddureva&project=operations3  2. L'insegnante discute su come formulare le condizioni dell'attività. L'attività viene formulata. Dieci volte in modo casuale, viene selezionata un'operazione aritmetica con il primo operando 6 e un secondo operando viene selezionato in

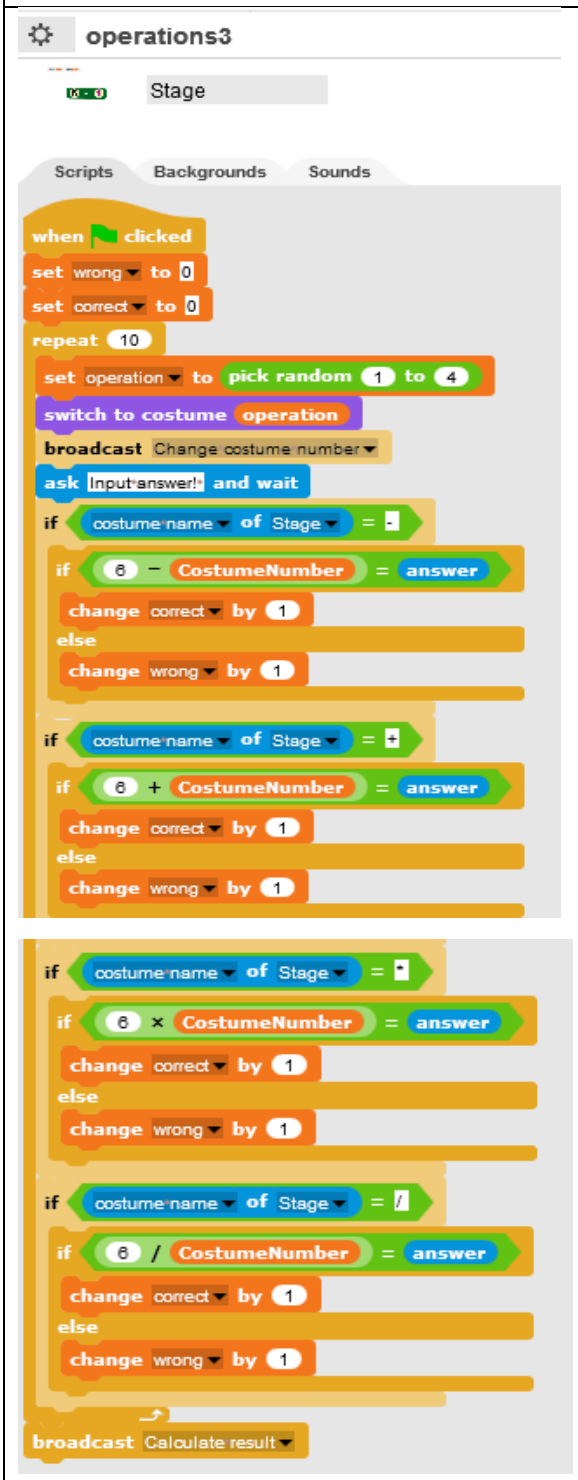
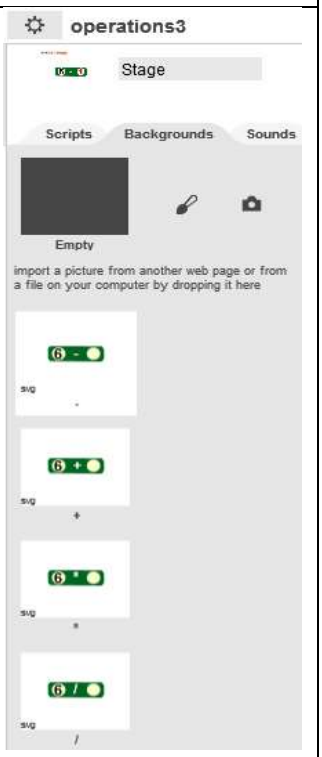
modo casuale, dai numeri da 1 a 3.
Il giocatore deve inserire la risposta corretta. Vengono contate le risposte giuste e sbagliate. Il risultato è riportato alla fine del gioco.

3. Le variabili sono commentate, così come il modo in cui sono definite, impostate e modificate.

4. Vengono rivisti i comandi dei numeri casuali, le operazioni aritmetiche e logiche, i comandi degli eventi di trasmissione.

5. Si discute se il codice di base è sulla scena o sullo *sprite*. Nell'esempio, il codice principale è la scena e il codice dello *sprite* ha degli *script* per cambiare il costume e calcolare il risultato finale.



Codice per la scena	Scenario della scena
	

Il codice della scena contiene le istruzioni per le variabili dela risposta giusta e sbagliata.

Per selezionare un'operazione vengono utilizzati i seguenti comandi:

La scelta del costume per lo sprite è fatta per trasmissione a "*Number Sprite*".

Il numero di costume selezionato viene memorizzato nella variabile "*Costume Number*" che è definito per tutti gli oggetti nel progetto e può quindi essere utilizzato nel codice della scena.

Una volta che lo scenario della scena/il costume dello sprite sono stati scelti a caso, viene posta una domanda al giocatore per inserire la risposta corretta per l'operazione con il seguente comando:

La risposta inserita viene confrontata con il risultato delle operazioni selezionate.

Viene utilizzato il seguente comando:

"*if (conditional)*"

"*else*"

Se viene selezionata l'operazione "-" viene eseguito un controllo se il risultato di 6 - "Numero costume Sprite" corrisponde alla risposta. Se corrisponde la variabile corretta aumenta altrimenti aumenta la variabile per il conteggio delle risposte errate.

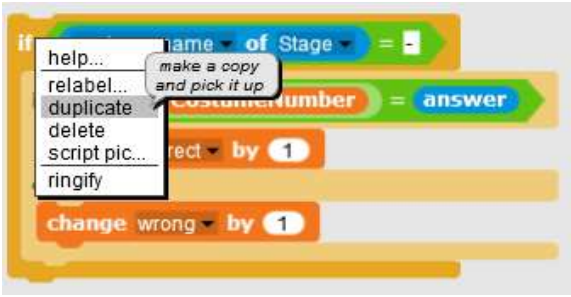
Per il resto dei comandi lo *script* è simile, la differenza è nell'operazione selezionata.

Per evitare ripetuti ordinamenti di codice per il resto delle operazioni, agli studenti può essere insegnato come copiare parte del codice e modificare

l'operazione aritmetica in  :

Copia del codice:

1. cliccare con il tasto destro del mouse sullo *script*
2. scegliere duplicato



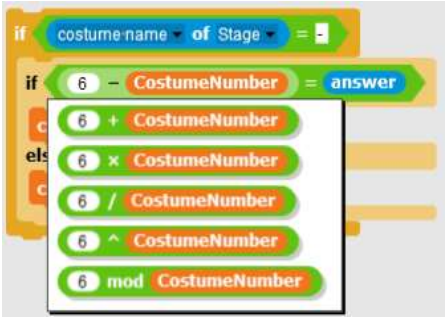
3. utilizzare il mouse per posizionare lo *script* duplicato nella posizione corrispondente.

   A discrezione dell'insegnante, gli studenti possono essere aiutati a capire come copiare parte del codice stesso.

Modificare l'operazione.

1. Cliccare con il tasto destro del mouse sul segno dell'operazione. Apparirà il menu contestuale.

2. Scegliere la nuova etichetta. Apparirà un elenco di operazioni.



	3. Scegliere l'operazione Nota: Se l'età degli studenti e la loro conoscenza delle operazioni aritmetiche lo consentono, l'attività può essere ampliata con operazioni, classificazione (^) e divisione per modulo (mod). Gli studenti lavorano in *team* creando i loro scenari e i costumi per lo *sprite*. Se ci sono vincoli temporali, è possibile utilizzare un progetto con metà supporto contenente la scena e lo *sprite*.
Strumenti e risorse per gli insegnanti	L'intera attività in Snap!: https://snap.berkeley.edu/project?user=ddureva&project=operations3 L'interra attività in Scratch: ● Дурева Д., М. Касева, Г. Тупаров, Компютърно моделиране, 4. клас, Просвета, 2018, София (Dureva, D., M. Kaseva, G. Tuparov, Kompyutarno modelirane, 4. klas, Prosveta, 2019, Sofia)
Risorse/materiali per gli studenti	● Attività non completa in Snap! https://snap.berkeley.edu/project?user=ddureva&project=operations_half ● Istruzioni per lo studente (C4G17_InstructionsForStudent.docx)

Scenario di apprendimento 18 - Riciclare

Titolo dello scenario di apprendimento	Riciclare
Precedente esperienza di programmazione	Mostrare e nascondere lo *sprite* Usare le variabili per il conteggio dei punti Usare il ciclo continuo Usare i condizionali Usare le operazioni per la comparazione Usare il rilevamento dei colori
Risultati di apprendimento	Obiettivi generali di apprendimento: ● variabili ● condizionali ● ciclo ● punto di direzione ● blocchi di rilevamento ● codice di refactor Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente usa *aspetta fino e operazioni logiche* alla fine del gioco ● lo studente usa *aspetta fino* e il blocco per cambiare la scena ● lo studente usa le variabili per contare i punti ● lo studente usa i condizionali e le operazioni logiche ● lo studente compara i codici degli *sprites* simili ● lo studente effettua il … codice di refactor ● lo studente sa posizionare gli *sprites* (in un'attività aggiuntiva utilizzare il posizionamento casuale)
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: qualcuno ha scaricato l'immondizia davanti alla scuola. Al giocatore viene chiesto di aiutare a separare la raccolta dei rifiuti ordinandola per il riciclaggio di carta e vetro. Quando l'immondizia viene posizionata nel contenitore corretto, l'immondizia viene nascosta. Se l'immondizia viene collocata nel contenitore errato, viene visualizzato il messaggio pertinente *" Questo non è un contenitore di carta" o "Questo non è un contenitore di vetro" e la spazzatura torna nella posizione originale*.

	Il gioco termina quando tutta la spazzatura viene messa nei contenitori giusti. **Compito**: gli studenti devono esplorare i codici della scena e degli *sprites*, confrontare i codici dei tipi di *sprites* carta e vetro, aggiungere nuovi *sprites* e *script* e cambiare la sceneggiatura sulla scena rispetto agli *sprites* appena aggiunti. Compiti aggiuntivi potrebbero essere: ● modificare la posizione degli *sprites* di rifiuti con scelta casuale delle coordinate degli *sprites* ● ridurre il numero di fasi ed estrarre il robot come *sprite* separato (il robot fa parte dello sfondo della scena). **Obiettivo: Gli studenti miglioreranno le loro conoscenze acquisite in precedenza ed estenderanno lo scenario di gioco con nuovi oggetti, codice e cambiando codice rispetto ai nuovi *sprite*. Impareranno il codice di refactor.**
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo (discussioni, con un gioco precedentemente preparato), apprendimento basato sulla programmazione del gioco, *problem solving*
Didattica	Lavoro individuale / lavoro in coppia / lavoro frontale con tutta la classe
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) 1. L'insegnante pone il problema della raccolta differenziata dei rifiuti e commenta i colori dei bidoni per i diversi tipi di immondizia: blu per la carta, verde per la plastica. 2. Si inizia con gli studenti che giocano e descrivono a parole: Quante scene guardano e quanti *sprite* (personaggi)?. Come inizia il gioco?. Quale *sprite* chiede il nome del giocatore?. Quante variabili vengono utilizzate e come vengono denominate? Cosa succede quando la carta viene posizionata in un contenitore di vetro e cosa quando viene posizionata in un contenitore di carta?

1. Aggiornamento dei comandi studiati

Vengono richiamati i comandi per avviare un dialogo con l'utente. Viene commentato il cambiamento delle scene - Scena 1 con il Robot, Scena 2 con la scuola e la spazzatura e scena 3 con il robot e la didascalia Bravo!. Vengono discussi i possibili comandi di cambio scena.

Si discute il fatto che il controllo del corretto posizionamento della spazzatura in un contenitore dovrebbe essere effettuato con un blocco condizionale e blocchi con condizioni tattili del gruppo di rilevamento "*Sensing group*".

Viene fornita una descrizione verbale: se un pezzo di carta da buttare tocca il cestino della carta, la spazzatura viene nascosta (posizionata

nel contenitore corretto) e i punti per i rifiuti di carta raccolti vengono aumentati di 1, Se un pezzo di immondizia di carta tocca il secchio di vetro dice "*Questo non è un contenitore di carta*".

Lo stesso succede con la spazzatura di vetro.

2. Esame dei codici di scene e personaggi.

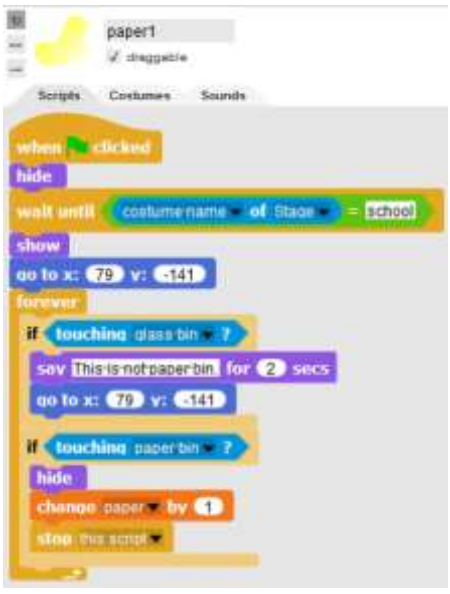
Dopo aver condiviso le possibilità di risolvere il problema, vengono discussi i codici per la scena e i personaggi.

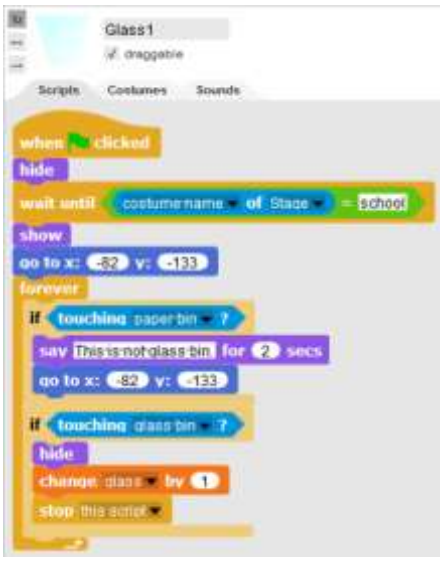
Il codice scena viene commentato ponendo enfasi su:

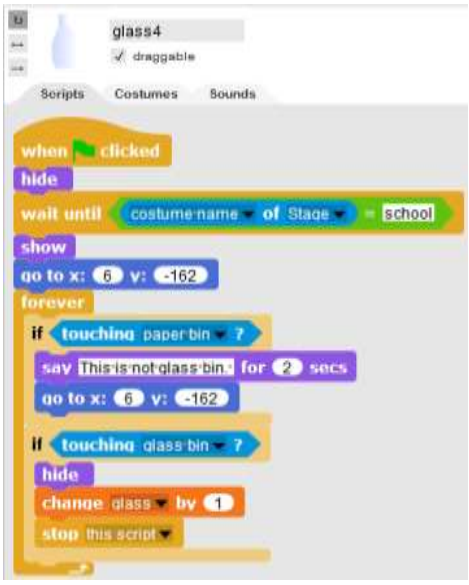
- l'impostare il valore iniziale della variabile del nome e utilizzo in una finestra di dialogo con l'utente
- il cambiare lo scenario della scena (costumi) e le condizioni per finire il gioco.

Quando si guardano i codici dei caratteri, è consigliabile mostrarli su una singola *slide* o dare la stampa di due codici ciascuno per i pezzi di

carta e vetro spazzatura. Viene effettuato un confronto tra gli elementi comuni e quelli diversi nei codici.







3. Impostare un'attività per completare il gioco con due nuovi *sprite*: immondizia di carta e immondizia di vetro, assegnando loro un codice e modificando la scena e i codici del contenitore dell'immondizia.

Viene discusso come creare i due nuovi *sprite*. Opzioni: duplica quelli esistenti e modificali in Snap!, creane dei nuovi in un editor grafico o cerca immagini distribuite liberamente su Internet e importale nel gioco.

	È inoltre necessario commentare le modifiche al codice scena relative al completamento del gioco. Dovrebbe anche essere discusso se sia possibile impostare i valori iniziali delle variabili non nel codice dei due contenitori, ma nel codice della scena e apportare una regolazione di conseguenza. A discrezione dell'insegnante, il compito può essere reso più difficile: ● la spazzatura dovrebbe essere sparsa in qualsiasi luogo adatto all'avvio del gioco. È bene notare qui che le coordinate in cui è possibile disperdere la spazzatura dovrebbero essere limitate in modo che si trovi in un luogo realistico. Ad esempio, delimitato dalle coordinate del rettangolo rosso.  ● introduci un nuovo Robot Sprite e si riduce il numero di elementi scenici sul palco. Scrivi il codice appropriato per il robot in modo che dialoghi con il giocatore anziché con uno sprite contenitore blu.
Strumenti e risorse per gli insegnanti	L'intera attività in Snap!: https://snap.berkeley.edu/project?user=ddureva&project=recycling L'intera attività in Scratch: ● Дурева Д., М. Касева, Г. Тупаров, Компютърно моделиране, 4. клас, Просвета, 2018, София (Dureva, D., M. Kaseva, G. Tuparov, Kompyutarno modelirane, 4. klas, Prosveta, 2019, Sofia)
Risorse/materiali per gli studenti	● Attività non completa in Snap! https://snap.berkeley.edu/project?user=ddureva&project=recycling ● Istruzioni per lo student (C4G18_InstructionsForStudent.docx)

Scenario di apprendimento 19.1 - Suonare un piano

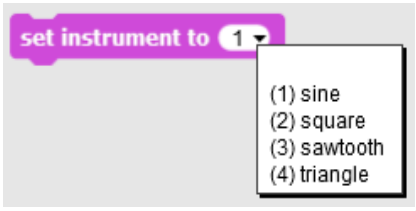
Titolo dello scenario di apprendimento	Suonare un piano
Precedente esperienza di programmazione	Usare le variabili per il conteggio dei punti Usare gli eventi *Quando sono cliccato* Usare il ciclo continuo Usare i condizionali Usare gli eventi di trasmissione per cambiare lo scenario, la scena e gestire le attività dello *sprite*
Risultati di apprendimento	Obiettivi generali di apprendimento: ● variabili ● condizionali ● ciclo ● eventi di trasmissione ● suoni ● programmare la musica Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente usa le variabili per il conteggio dei punti ● lo studente avvia le variabili per il conteggio dei punti ● lo studente usa le condizionali per stimare i punti realizzati ● lo studente usa gli eventi di trasmissione per cambiare lo scenario , la scena e gestire le attività di *sprite* ● lo studente usa i blocchi dal gruppo suoni "*Sound*" per comporre melodie ● lo studente identifica la necessità di ripetere il ciclo per ridurre il numero di blocchi nello *script* ● lo studente estende la funzionalità del gioco
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: Entriamo nel meraviglioso mondo della Regina Maria "*Queen Mary*". La regina invita il giocatore nel suo palazzo ad ascoltare un po 'di musica. Nella sala da ballo, il suo piccolo amico dinosauro Dino suona il piano. Nel gioco Dino suona alcuni toni musicali e i giocatori devono riconoscere di quale tono si tratta. Se indovinano correttamente

	ottengono un punto per la risposta giusta, altrimenti ottengono una riduzione del punto per la risposta sbagliata. Dopo aver identificato i toni, viene impostato un compito più complesso: Dino suona una melodia e il giocatore deve riconoscere quale sia la canzone. Il giocatore ottiene 5 punti per ogni melodia correttamente identificata. **Compito**: Gli studenti usano un file non completo di costumi di scena, scenografia e *sprite*. Devono pianificare le variabili necessarie, determinare quali blocchi necessitano; conoscere i blocchi del gruppo Suono "*Sound"* e il modo di suonare le note. Occorre creare *script* per riprodurre diversi brani "*tunes*". **Obiettivo: Gli studenti impareranno a conoscere il codice e l'esecuzione delle melodie e miglioreranno le loro conoscenze acquisite in precedenza su variabili, cicli, condizionali, trasmissioni e altri eventi.**
Durata delle attività	90 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo (discussioni, esperimento con il gioco preparato precedentemente), apprendimento basato sulla programmazione del gioco, *problem solving*
Didattica	Lavoro individuale / lavoro in coppia/ lavoro frontale con tutta la classe
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) 1. L'insegnante prepara l'attività per creare il gioco. Vengono discussi i mezzi con cui è possibile completare l'attività. Si è concluso che al momento non sono a conoscenza delle risorse di scrittura del codice disponibili per programmare una melodia. 2. L'insegnante dimostra parte del gioco componendo una melodia. https://snap.berkeley.edu/project?user=ddureva&project=Play_a_Piano_1

3. L'insegnante mostra il codice e spiega come utilizzare i comandi del gruppo audio "*Sound* ".

In Snap! è possibile utilizzare suoni dalla libreria integrata, nonché file dal computer o toni musicali riprodotti su vari strumenti.

Per selezionare uno strumento, utilizzare il comando:



,

Nota: Ci sono molti altri strumenti in Scratch.

Gli studenti testano il suono dei vari strumenti.

4. L'insegnante spiega come impostare le note musicali:

Si usa il commando 

In esso, il primo numero imposta la nota e il secondo numero descrive per quanto tempo deve essere suonata.

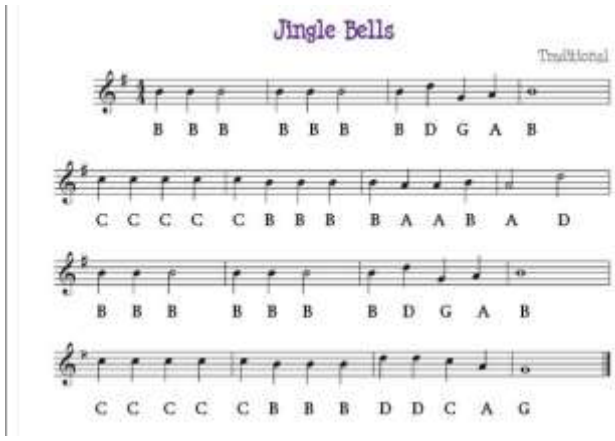
Quando si clicca sulla freccia accanto al primo numero, viene visualizzata una tastiera di pianoforte e da essa può essere selezionata una nota. Questa tastiera per pianoforte si estende su due ottave.

C	C#	D	Eb	E	F	F#	G	G#	A	Bb	B	C
60	61	62	63	64	65	66	67	68	69	70	71	72

La durata di ogni nota è impostata dai numeri 1 - nota intera, 0,5 - metà, 0,25 - un quarto. (Per gli studenti che non hanno studiato i numeri reali, le frazioni decimali possono essere presentate sotto forma di frazioni ordinarie: ½, ¼, 1/8, etc.)

A discrezione dell'insegnante, gli studenti possono sperimentare i comandi e stabilire autonomamente le dipendenze.

5. La sceneggiatura del brano di *Jingle Bells* viene discussa anche usando lo spartito.

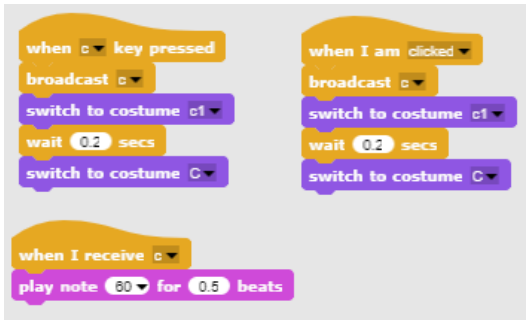


6. L'attività è impostata per ridurre il numero di righe nel codice che si ripetono. Viene discusso il comando da utilizzare (ciclo ripetuto). Gli studenti sono divisi in squadre che sono necessarie per creare il gioco, impostato all'inizio della lezione. Ogni squadra discute lo scenario di gioco e descrive il piano di gioco nel foglio descrittivo. È possibile aggiungere tabelle alla descrizione per dettagliare le azioni nelle fasi e negli sprite. Una condizione può essere aggiunta per far ballare il dinosauro mentre gioca. (Il dinosauro ha diversi costumi nel file preparato).

	7. L'insegnante può visualizzare alcune parti degli scenari dal file. https://snap.berkeley.edu/project?user=ddureva&project=PlayAPiano
Strumenti e risorse per gli insegnanti	L'intera attività in Snap!: https://snap.berkeley.edu/project?user=ddureva&project=Play_a_Piano_1 https://snap.berkeley.edu/project?user=ddureva&project=PlayAPiano
Risorse/materiali per gli studenti	● Attività non completa in Snap!: https://snap.berkeley.edu/project?user=ddureva&project=Play_a_Piano_Half_backed ● Istruzioni per lo studente (C4G19.1_InstructionsForStudent.docx)

Scenario di apprendimento 19.2 - Suonare un piano

Titolo dello scenario di apprendimento	Suonare un piano
Precedente esperienza di programmazione	Usare il ciclo continuo Usare le variabili Usare i condizionali
Risultati di apprendimento	Obiettivi generali di apprendimento: ● condizionali ● cicli Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente usa la ripetizione del ciclo per suonare musica ● lo studente usa il codice per far reagire gli *sprite* agli input ● lo studente aggiunge i suoni ad uno *sprite* ● lo studente usa il code per cambiare il costume dello *sprite*
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: Lo studente deve suonare una canzone su un piano in base alle note fornite. **Compiti**: Gli studenti dovrebbero programmare i tasti del piano - ogni tasto deve suonare un tono particolare. Sulla scena, devono essere mostrati due diversi pulsanti, uno per visualizzare le note e l'altro per suonare la melodia. **Obiettivo: Gli studenti impareranno a suonare e cambiare costume cliccando su uno *sprite*.**
Durata delle attività	45 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento basato sulla programmazione del gioco, *problem solving*

Didattica	Lavoro individuale / Lavoro in coppia
Riepilogo dell'insegnamento	(Motivazione-introduzione, attuazione, riflessione e valutazione) All'inizio, viene mostrato un piano sulla scena. Accanto al piano dovrebbero esserci due pulsanti. Fare clic sul primo pulsante per visualizzare le note e le parole del brano e fare clic sul secondo pulsante per riprodurre la melodia che deve essere ripetuta. Inoltre, accanto al piano dovrebbe essere presente il pulsante "X", che riavvierà il progetto. [Fase 1] Scarica l'archivio con i materiali. L'archivio contiene tutti gli sfondi e gli sprite necessari per questa attività. Viene fornito lo sfondo e anche gli sprite per i tasti C - G con un codice per suonare un tono premendo gli sprite e usando la tastiera. Per esempio. Per il codice chiave C è:  Gli studenti devono aggiungere uno sprite per i tasti A e B. Aggiungere un nuovo sprite - tasto A e importare il costume "A" dai materiali.  Fare lo stesso per il tasto B e posizionarli alla fine della tastiera.

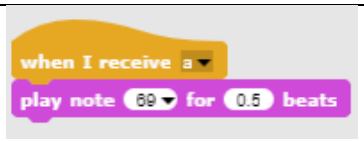
Duplica lo sprite "black_key" 2 volte per ottenere 5 tasti neri e nominali tasto nero 4 e tasto nero 5. Posiziona i nuovi tasti neri tra i tasti G e A e i tasti A e B.

Deselezionato il pulsante "trascinabile", quindi gli sprite dei tasti non possono essere spostati durante la riproduzione.

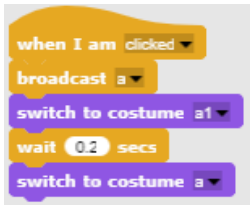
[Fase 2]

Abilita la riproduzione dei toni premendo gli sprite. Per il tasto "A", aggiungi il cappello "Quando faccio clic" e consenti di trasmettere il messaggio "a".

Per produrre suono quando si preme un tasto, aggiungere il cappello "Quando ricevo c" e aggiungere una nota di riproduzione 60 per 0,5 battute.



Per evidenziare quale tasto viene premuto, il costume di quello *sprite* dovrebbe essere temporaneamente cambiato. Importa costume c1 nello sprite C. Nel blocco "Quando faccio clic", cambia il costume in c1 per 0,2 secondi, quindi torna al costume c.

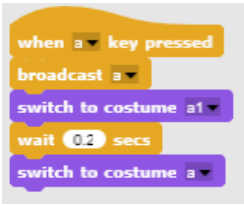


[Fase 3]

Ripetere il passaggio 2 per il tasto B. Definire quel tasto B per suonare la nota 71.

[Fase 4]

Per suonare il piano usando la tastiera, aggiungi un blocco "Quando premuto c" al tasto c sprite e copia il resto del codice dal blocco "Quando faccio clic".



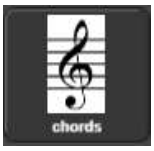
Notare che se si tiene premuto il tasto c sulla tastiera, il suono verrà ripetuto finché si tiene premuto il tasto. Ciò accade perché il messaggio "a" viene ripetutamente trasmesso. Per interrompere la trasmissione di un messaggio, alla fine del codice aggiungere un blocco "attendere fino" dalla sezione “controllo”.

Per terminare la trasmissione di un messaggio, utilizzare l'operatore "non" e aggiungere un blocco "tasto a premuto".

Fare lo stesso per il tasto B.

[Fase 5]

Crea un nuovo sprite e importa un'immagine di una chiave di violino come costume. Questo sarà un pulsante per visualizzare le parole e le note da suonare.



Per visualizzare le note, abilitare la trasmissione del messaggio "accordi" quando si fa clic sul pulsante.

Importa un nuovo "accordo" in costume per il palcoscenico.

Aggiungi un codice che consenta allo stage di cambiare il costume in

	"accordi" quando riceve un messaggio "accordi".  [Fase 6] Trova lo sprite con una nota come costume. Questo sarà un pulsante per riprodurre il brano che deve essere ripetuto.  Il codice è scritto per i primi due versetti della canzone e devi scrivere il codice per gli altri versi. È la stessa canzone visualizzata negli spartiti.

[Fase 7]

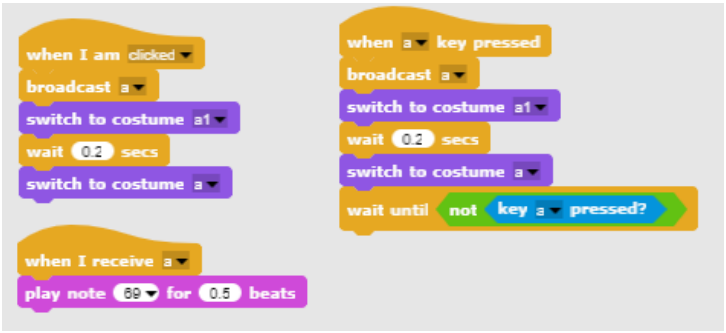
Crea un nuovo pulsante X che ripristinerà il progetto (senza le note). Crea un nuovo sprite - resetta, scegli il costume "X" e imposta la sua taglia al 50%. Abilita la trasmissione del messaggio "vuoto" quando

si preme il pulsante.

Ad un cappello "Quando ricevo" sul palco per cambiare il costume in "vuoto" dopo aver ricevuto il messaggio "vuoto".

[Codice finale]

Chiave A



Chiave di violino

	Nota 

when I am clicked
repeat 2
play note 60 for 0.5 beats
repeat 2
play note 67 for 0.5 beats
repeat 2
play note 69 for 0.5 beats
play note 67 for 0.5 beats
wait 0.1 secs
repeat 2
play note 65 for 0.5 beats
repeat 2
play note 64 for 0.5 beats
repeat 2
play note 62 for 0.5 beats
play note 60 for 0.5 beats
wait 0.1 secs
repeat 2
play note 67 for 0.5 beats
play note 67 for 0.5 beats
play note 65 for 0.5 beats
play note 65 for 0.5 beats
play note 64 for 0.5 beats
play note 64 for 0.5 beats
play note 62 for 0.5 beats
wait 0.1 secs
repeat 2
play note 60 for 0.5 beats
repeat 2
play note 67 for 0.5 beats
repeat 2
play note 69 for 0.5 beats
play note 67 for 0.5 beats
wait 0.1 secs
repeat 2
play note 65 for 0.5 beats
repeat 2
play note 64 for 0.5 beats
repeat 2
play note 62 for 0.5 beats
play note 60 for 0.5 beats

	X  Per lo stage  [Compiti aggiuntivi] Gli studenti possono aggiungere attività in base ai loro desideri o possono seguire le seguenti: • Duplica la nota sprite (e cambia la sua posizione sullo sfondo) e scrivi un programma per un altro brano. • Aggiungi uno sfondo con gli accordi per il nuovo brano.
Strumenti e risorse per gli insegnanti	L'intera attività in Snap!: https://snap.berkeley.edu/project?user=ifrankovic&project=Play%20a%20Piano

Risorse/materiali per gli studenti	Attività non complete in Snap!:
	https://snap.berkeley.edu/project?user=ifrankovic&project=Play%20Piano (27.1.2020)
	Immagini:
	• Immagini per gli Sprite:
	  • a.png, a1.png
	  • b.png, b1.png
	  • violin_key.png
	• per lo sfondo: notes.png

Scenario di apprendimento 20 - Test

Titolo dello scenario di apprendimento	Test
Precedente esperienza di programmazione	Mostrare e nascondere lo *sprite* Usare le variabili per il conteggio dei punti Usare il ciclo continuo Usare i condizionali Usare le operazioni per la comparazione Usare il riconoscimento dei colori Cambiare la scena
Risultati di apprendimento	Obiettivi generali di apprendimento: ● variabili ● condizionali ● ciclo ● blocchi di riconoscimento Obiettivi specifici di apprendimento orientati al pensiero algoritmico: ● lo studente usa i condizionali per stimare la risposta – corretta o sbagliata ● lo studente usa i blocchi per cambiare il costume della scena ● lo studente usa le variabili peril conteggio dei punti ● lo studente usa le operazioni logiche ● lo studente utilizzare un editor grafico esterno per preparare gli sfondi complessi delle fasi
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione**: Aiuta il tuo insegnante a testare la tua conoscenza di Snap! creando un gioco basato sulle missioni per testare i comandi utilizzati in Snap!. **Compito**: Gli studenti devono esplorare il gioco posto come esempio, dal gioco non completo "*half-backed*" devono trovare o progettare il proprio *sprite* che porrà domande o progettare lo sfondo dello stadio iniziale e gli sfondi dello stadio con domande appropriate, modificare ed estendere gli *script* in prova rispetto alle domande. **Obiettivo: Gli studenti miglioreranno le proprie conoscenze,**

	acquisite in precedenza, ed estenderanno lo scenario di gioco con nuovi sfondi e codici e sapranno modificare il codice rispetto alle nuove fasi.
Durata delle attività	90 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo (discussioni, esperimento con un gioco preparato in anticipo), apprendimento basato sulla programmazione del gioco, *problem solving*
Didattica	Lavoro individuale / Lavoro in coppia / lavoro frontale con l'intera classe
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) 1. L'insegnante condivide con gli studenti la necessità di creare un test di gioco per testare le conoscenze di programmazione. 2. Incarica gli studenti di giocare e descrivere a parole: quante decorazioni sceniche osservano e quanti *sprite* (personaggi)?. Come inizia il gioco? Quante variabili vengono utilizzate, come vengono denominate, a cosa servono? Cosa succede quando la risposta è giusta / sbagliata? Come vengono presentate le domande nel test? A discrezione dell'insegnante la scelta di svolgere l'attività con lavoro individuale o lavoro in coppia. 3. Si commenta l'algoritmo per porre e rispondere alle domande: questa attività è frontale: • passare a un costume della scena (contiene la domanda); • assegnando ad Abby (*sprite* ragazza) un costume per porre una domanda; • Abby dice – Risposta Sì o No; • il giocatore inserisce una risposta - Sì o No; • se la risposta è corretta, Abby dice "Corretto" e il numero di risposte corrette aumenta, altrimenti Abby dice "Ti sbagli" e

aumenta il numero di risposte sbagliate.

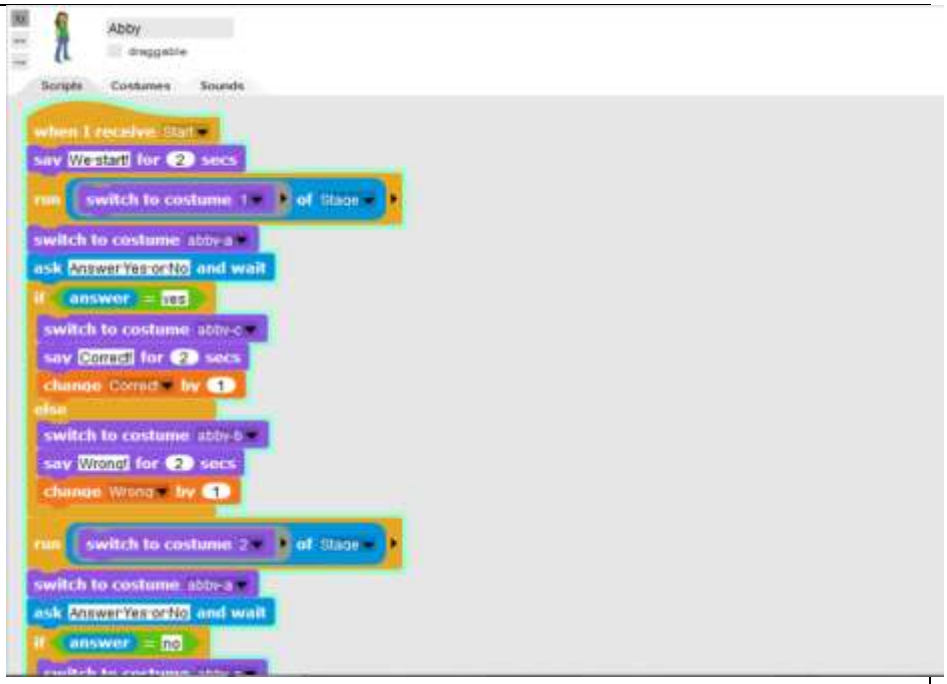
4. Si commenta cosa succede dopo aver risposto a tutte le domande. Questa attività è frontale:

• cambiare il costume/sfondo della scena

• Abbey indica i numeri delle risposte giuste e sbagliate e fornisce una stima

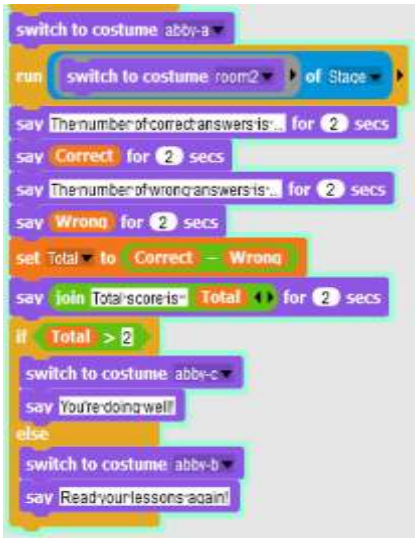
5. Si esaminano i codici nel gioco e si aggiornano le proprie conoscenze.

   Vengono commentati i comandi per avviare un dialogo con l'utente, per modificare la scena e il costume del personaggio e i comandi condizionali. Vengono esaminati i codici di ciascun personaggio. Si commenta la creazione di una variabile.



Si commentano le situazioni in cui la risposta corretta (YES) e non corretta (NO).

Il codice per la classificazione viene discusso in dettaglio e sul motivo per cui viene utilizzata la variabile totale.



Si discute il modo di progettare lo scenario della scena per le singole domande.

Perché in Snap! non è possibile scrivere testi in costumi e scenografie, è necessario utilizzare un editor grafico esterno.

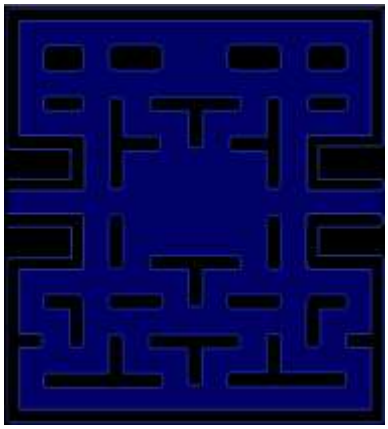
	Un'altra opzione è utilizzare MS Power point per creare la domanda ed esportare la casella di testo corrispondente in formato grafico. Può essere rivisto l'inserimento di un costume in Snap! 1. dividere il gruppo in gruppi di 2 o 3 studenti; 2. pubblicare l'argomento per le domande del test, ad es., utilizzo delle variabili, cicli, operazioni di movimento, rilevamento, aritmetica e logica; 3. progettare le scene con domande su un argomento da parte del rispettivo team. Se necessario, l'insegnante consiglia agli studenti il contenuto delle domande. Le domande vengono discusse e ogni squadra crea una scena per almeno due domande; 4. viene fornito agli studenti un file non completo di costumi per la scena e gli *sprite*. Se lo desiderano, possono anche creare un file tutto loro. Il lavoro viene svolto per analogia con il test del modello.
Strumenti e risorse per gli insegnanti	L'intera attività in Snap!: https://snap.berkeley.edu/project?user=ddureva&project=test2 L'intera attività in Scratch: ● Дурева Д., М. Касева, Г. Тупаров, Компютърно моделиране, 4. клас, Просвета, 2018, София (Dureva, D., M. Kaseva, G. Tuparov, Kompyutarno modelirane, 4. klas, Prosveta, 2019, Sofia)
Risorse/materiali per gli studenti	● Half-baked activity in Snap!: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_20_test_en_tmp ● Istruzioni per lo studente (C4G20_InstructionsForStudent.docx)

Scenario di apprendimento 21 - Gioco PACMAN semplificato

Titolo dello scenario di apprendimento	Gioco PACMAN semplificato
Esperienza di programmazione precedente	• condizionali, • codifica di più oggetti, • rilevamento del colore singolo, • cicli (per sempre, ripetere fino a), • movimento di oggetti basato su eventi, • numeri casuali
Risultati di apprendimento	Risultati di apprendimento generali: • clonazione di un oggetto, • definizione del comportamento di un clone, • trasmissione di messaggi, • letture di valori booleani in espressioni logiche, • definizione, differenziazione, controllo dinamico e risposta a due diversi stati di gioco. Risultati di apprendimento specifici orientati al pensiero algoritmico: • lo studente implementa il movimento degli oggetti con i tasti freccia utilizzando gli eventi e tiene conto delle restrizioni, • studente utilizza cloni per creare istanze dell'oggetto originale, • lo studente sa come codificare un comportamento di ciascun clone, • gli studenti conoscono il significato di invio di messaggi, • lo studente implementa l'invio di un messaggio dal clone al contatore di incrementi, • lo studente sa come rilevare il messaggio ricevuto dall'oggetto e fornisce una risposta appropriata
Obiettivo, Compiti e Breve Descrizione delle Attività	**Breve descrizione:** programma di gioco in cui il personaggio principale raccoglierà stelle posizionate casualmente e verrà inseguito da un fantasma.

	Compiti: gli studenti devono programmare il movimento del personaggio principale in modo che si muova all'interno di un labirinto. Devono applicare restrizioni di movimento in modo che il personaggio principale non possa muoversi attraverso i muri. Successivamente devono programmare un oggetto stella che si clonerà automaticamente all'avvio del gioco e quindi in una nuova posizione casuale ogni volta che un personaggio lo raccoglierà. Devono memorizzare il valore delle stelle raccolte e finire il gioco quando il giocatore raccoglie 20 stelle. Per rendere il gioco più interessante, possono programmare fantasmi malvagi che si muoveranno casualmente in tutto il labirinto. Se un giocatore tocca il fantasma, il gioco è finito. **Con questa attività gli studenti rivedranno le loro conoscenze sul movimento all'interno di un labirinto con l'uso del blocco di colori sensoriali che hanno appreso nelle attività precedenti. Verranno introdotti al concetto di clonazione dell'oggetto con restrizioni di posizione e come creare un personaggio non giocatore molto semplice con un suo movimento casuale.**
Durata delle attività	90 minuti
Strategie e metodi di apprendimento e insegnamento	Apprendimento attivo, apprendimento collaborativo, problem solving
Didattica	Insegnamento frontale Lavoro individuale / in coppia / di gruppo
Riepilogo dell'insegnamento	(Motivazione-introduzione, attuazione, riflessione e valutazione) Il giocatore sta raccogliendo stelle posizionate casualmente mentre viene inseguito da un fantasma rosso. Se un giocatore e un fantasma si scontrano, il gioco è finito. Se un giocatore raccoglie le 20 stelle, vince. [Fase 1]

Chiediamo agli studenti di progettare un labirinto in cui l'area in cui il giocatore è autorizzato a muoversi è di un colore (ad esempio blu) e pareti che fermano il movimento del giocatore che sono colorate in qualche altro colore (ad esempio nero). Per risparmiare tempo, possiamo preparare in anticipo l'immagine di sfondo del labirinto.



[Fase 2]

Devono disegnare il pacman e il fantasma rosso. Per una stella possiamo semplicemente disegnare un cerchio all'interno di Snap!:

[Fase 3]

Per far muovere Pacman, possiamo usare diverse possibilità. Il seguente esempio è uno di questi. In esso utilizziamo un sistema di eventi per rilevare quale tasto viene premuto, a sinistra, a destra, in alto o in basso. Dopo che si è verificato ciascuno di questi eventi, dobbiamo verificare se sta toccando il colore dell'area in cui è autorizzato a muoversi. In questo caso, si trasforma prima in quella direzione e si muove. Ma se tocca il colore delle pareti, deve tornare indietro, perché altrimenti rimarrebbe bloccato al muro a causa della prima condizione.

[Fase 4]

Il prossimo compito è programmare le stelle. Le stelle saranno tutte uguali ma ce ne saranno molte. In questo caso è meglio che creare più oggetti identici (nel nostro caso 20), creare un oggetto e quindi crearne i cloni. All'inizio del gioco il primo clone apparirà casualmente all'interno del labirinto, quindi quando il giocatore lo raccoglierà scomparirà e ne verrà creato uno nuovo in una diversa posizione casuale. Per creare il primo clone all'inizio del gioco abbiamo inserito questo codice su uno script della scena.

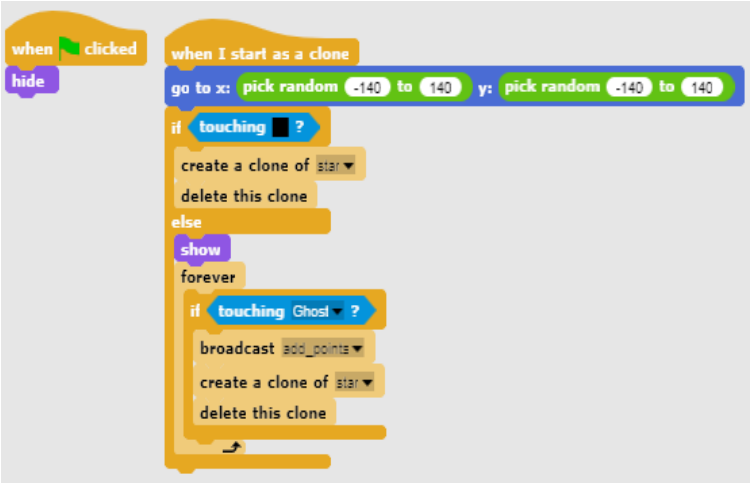
Per nascondere un oggetto originale e mostrare solo i cloni, dobbiamo farlo all'inizio del programma.

Per trovare posizioni casuali adatte dobbiamo osservare alcune restrizioni. Se una stella viene creata su un muro, un giocatore non può raggiungerla, il che significa che non possiamo posizionarla lì. La strategia per farlo è la seguente:

1. Dobbiamo trovare la posizione casuale x, y del clone stellare.

Entrambe le coordinate xey sono sullo stesso intervallo [-140, 140]. Quindi scegliamo un numero casuale da quell'intervallo per entrambi.

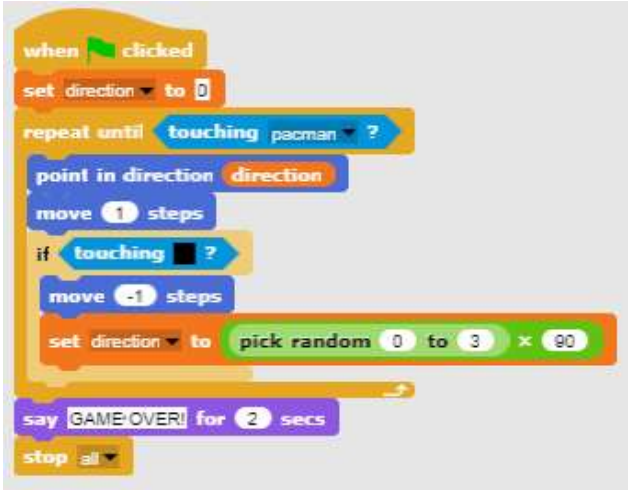
2. Quindi controlliamo se questo clone sta toccando il colore del muro. In questo caso la sua posizione non è legale.
3. Se la posizione è ok, dobbiamo mostrare il clone (ricorda, l'originale è nascosto e il clone sarebbe nascosto anche se non usassimo il blocco mostra) e in loop continuo verificare se si verifica la collisione con il giocatore.
4. Se la posizione non è corretta, creiamo un nuovo clone (sperando che per quello nuovo, vengano scelti numeri casuali in modo che venga inserito in una posizione legale) ed eliminiamo questo.
5. Per contare i cloni raccolti dobbiamo informare un contatore totale di stelle che deve essere definito al di fuori del clone, ad es. sul giocatore. Questo può essere fatto trasmettendo un messaggio in cui si è verificata la collisione. Quindi possiamo eliminarlo.



[Fase 5]

Quindi programmiamo un fantasma. Deve muoversi in modo casuale in tutto il labirinto e deve cambiare direzione quando si imbatte nel muro. Per rendere casuale il suo movimento, vogliamo che si muova in una direzione casuale dopo l'urto. In Snap! le direzioni sono espresse in gradi:

1. 0 gradi - UP
2. 180 gradi - DOWN
3. 90 gradi - RIGHT
4. 270 gradi - LEFT

	In altre parole, se scegliamo a caso il numero da 0 a 3 e lo moltiplichiamo per 90, otteniamo una direzione casuale! Deve muoversi fino a quando non si scontra con un pacman. Quindi il gioco è finito.  [Fase 6] Ora dobbiamo programmare quando il giocatore vincerà la partita. Sarà quando raccoglierà 20 stelle. Abbiamo un contatore di stelle all'interno della sceneggiatura di Pacman. All'inizio lo inizializziamo a 0, quindi aumentiamo il suo valore di 1 ogni volta che il clone invia un messaggio che il giocatore lo ha raccolto. Se il contatore arriva a 20, Pacman vince e dobbiamo fermare il gioco. 
Strumenti e risorse per gli insegnanti	• L'intera attività in Snap!: https://snap.berkeley.edu/project?user=zapusek&project=pacman_clone • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*.

	Ljubljana: MK.
Risorse/materiali per gli studenti	• Modello in Snap!: https://snap.berkeley.edu/project?user=zapusek&project=pacman_template • Istruzioni per lo studente (C4G21_InstructionsForStudent.docx)

BIBLIOGRAFIA

Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena.

Rugelj, J. (2019). Game design based learning of programming.

Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.