O3 – Instructional Support Content

O3/A2 - USER GUIDE ON THE CODING4GIRLS SERIOUS GAME APPROACH

Dati del documento

Risultato: O3/A2 – Guida utente sull'approccio ai serious game di CODING4GIRLS

Opera di Ingegno N. - Titolo: O3 – Contenuto di supporto didattico

Leader dell'Opera di Ingegno: South-West University "Neofit Rilski" (Bulgaria)

Contributi

South-West University "Neofit Rilski", Bulgaria

Daniela Tuparova, Boyana Garkova, Ivanichka Nestorova, Rositsa Georgieva

UTH – Grecia

Hariklia Tsalapata, Olivier Heidmann, Kostas Katsimentes, Christina, Taka Roxani, Sotiri Evangelou, Nadia Vlachoutsou

University of Rijeka – Croazia

Nataša Hoić-Božić, Martina Holenko Dlab, Ivona Franković, Marina Ivašić Kos

EU-Track – Italia

Michela Tramonti, Alden Meirzhanovich Dochshanov and Luigi Tramonti

Virtual Campus - Portogallo

Carlos V. Carvalho, Rita Durão.

University of Ljubljana - Slovenia

Joze Rugelj, Mateja Bevcic, Spela Cerar, Matej Zapushek

GOI - Turchia

Ahu Şimşek, K.Fatih Mutlu, Abdurrahman Saygın

Dichiarazione di limitazione di responsabilità

Questo progetto è stato finanziato dal Programma Erasmus + dell'Unione Europea.
Le informazioni e le opinioni contenute in questa pubblicazione sono quelle degli autori e non riflettono necessariamente l'opinione ufficiale dell'Unione Europea. Né le istituzioni e gli organi dell'Unione Europea né qualsiasi persona che agisce per loro conto può essere ritenuta responsabile per l'uso che può essere fatto delle informazioni in essa contenute.
Tutti i diritti sono riservati. La riproduzione è autorizzata, salvo a fini commerciali, a condizione che la fonte sia riconosciuta.

Copyright © Coding4Girls, 2018-2020

Creative Commons - Attribution-NoDerivatives 4.0

International Public license (CC BY-ND 4.0)

INDICE

1. INTRODUZIONE 7

2. CONCETTI DI BASE 9

2.1. Serious game 9

2.2. Apprendimneto basato sul Gioco 11

2.3. Metofidologia del Design thinking 15

2.4. Teorie di apprendimento e apprendimento basato sul gioco 18

3. RAGAZZE, ICT E SERIOUS GAME 22

3.1. Posizione delle donne nel settore ICT– stato dell'arte 22

3.2. Attitudini delle ragazze ai serious game 23

4. INSEGNAMENTO DELLA PROGRAMMAZIONE ATTRAVERSO I GIOCHI 26

4.1. Gli approcci a supporto dell'insegnamento della programmazione attraverso i giochi 26

4.2. Gli ambienti basati sul gioco per l'insegnamento della programmazione 30

4.2.1. Apprendimento tramite il game-design 30

Scratch 30

Snap! 31

Alice 32

Tynker 33

4.2.2. L'apprendimento della programmazione in ambiente basato sul gioco 34

Code Combat 34

Human Resource Machine 35

LightBot 35

May's Journey 37

No Bug's Snack Bar 38

Robot ON! 39

Gioco educativo Pacman 40

CMX 42

5. LA PIATTAFORMA DI GIOCO CODING4GIRLS PER GLI INSEGNANTI E STUDENTI 45

5.1. La piattaforma di Coding4Girls e l'approccio design thinking 45

5.2 La Piattaforma degli Insegnanti 46

5.3. L'ambiente di Gioco degli Studenti 52

6. ESEMPI DI LEZIONI (SCHEDE DI APPRENDIMENTO) 58

6.1. Gli scenari di apprendimento; 58

6.2. Esempi con utilizzo dell'ambiente di gioco sviluppato nell'ambito del progetto Coding4Girls 60

APPENDICE 1. SCHEDE DI APPRENDIMENTO 73

Scenario di apprendimento 1 – Introduzione all'interfaccia di Snap! 73

Scenario di apprendimento 2 – E' tempo di dare vita al tuo *sprite* 78

Scenario di apprendimento 3 - Muoversi sulla scena 83

Scenario di apprendimento 4 - Cambiare costume e roteare 90

Scenario di apprendimento 5 - Suoni della fattoria 97

Scenario di apprendimento 6 – Le vacanze estive di un Camaleonte 105

Scenario di apprendimento 7 - Aiutare il principe e principessa a trovare i loro animali 116

Scenario di apprendimento 8 - Disegnare con il gesso 124

Scenario di apprendimento 9 - Raccogliere la spazzatura e pulire il parco 136

Scenario di apprendimento 10 – Nutrire i gatti 147

Scenario di apprendimento 11 - Indovinare il numero di gatti in un rifugio 156

SCENARI AVANZATI DI APPRENDIMENTO 165

Scenario di apprendimento 12 - Catturare cibi sani 165

Scenario di apprendimento 13 - *Storytelling* 175
Scenario di apprendimento 14 - Disegnare 187
Scenario di apprendimento 15 - Catturare il topo 198
Scenario di apprendimento 16 - Acquistare cibo per un picnic 208
Scenario di apprendimento 17 - Operazioni 218
Scenario di apprendimento 18 - Riciclare 224
Scenario di apprendimento 19.1 - Suonare un piano 232
Scenario di apprendimento 19.2 - Suonare un piano 238
Scenario di apprendimento 20 - Test 248
Scenario di apprendimento 21 - Gioco PACMAN semplificato 254
APPENDICE 2. CODICI DEGLI SCENARI DI APPRENDIMENTO 262
SCENARI DI APPRENDIMENTO DI BASE 262
Scenari in INGLESE 262
Scenari in SLOVENIO 264
Scenari in ITALIANO 265
Scenari in CROATO 265
Scenari in BULGARO 268
Scenari in GRECO 270
Scenari in PORTOGHESE 272
Scenari in TURCO 274
SCENARI AVANZATI DI APPRENDIMENTO 276
Scenari in INGLESE 276
Scenari in SLOVENIO 278
Scenari in ITALIANO 279
Scenari in CROATO 281
Scenari in BULGARO 282

Scenari in GRECO 284

Scenari in PORTOGHESE 285

Scenari in TURCO 287

BIBLIOGRAFIA 289

1. INTRODUZIONE

Il progetto Coding4Girls si rivolge ai settori dell'istruzione e della formazione in modo che siano aperti, innovativi e integrati nell'era digitale, concentrandosi sulle competenze di programmazione che sono molto richieste in una società guidata dalla tecnologia. Inoltre, la capacità di progettare algoritmi, codificare e programmare sono diventate abilità personali rilevanti in quanto connesse con il ragionamento logico e le capacità di *problem solving*. Le competenze informatiche sono altamente desiderabili in quanto fanno parte delle capacità richieste dal mercato del lavoro nei settori guidati dall'innovazione i quali guideranno una crescita economica sostenibile nei prossimi anni. Ciò porterà ad una maggiore occupazione e, di conseguenza, alla coesione sociale.

Tuttavia, attualmente vi è una carenza di personale specializzato nel settore dell'informatica e le aziende hanno difficoltà a trovare dipendenti con le competenze ICT richieste. CODING4GIRLS intende affrontare questa lacuna supportando le competenze informatiche negli ultimi anni della scuola di base e nei primi anni della scuola secondaria. Sembra che sia il momento in cui molti studenti perdono interesse per l'informatica. Inoltre, vi è un divario di genere ampiamente riconosciuto in questo settore con le ragazze che non sono interessate a perseguire una carriera informatica.

CODING4GIRLS si rivolge all'istruzione, alla formazione e ai giovani promuovendo le pari opportunità tra ragazze e ragazzi nelle carriere altamente gratificanti dell'informatica. Il progetto combatte le percezioni errate e gli atteggiamenti di studenti, insegnanti e genitori sulle carriere informatiche, ne elogia i benefici per ragazze e ragazzi, dimostrando i suoi collegamenti con la vita reale ed evidenziando il fatto che tali carriere sono gratificanti indipendentemente dal sesso o dal contesto. Affronta anche i risultati insufficienti nelle competenze informatiche, che fanno parte del più ampio gruppo di competenze scientifiche e tecnologiche (STEM).

Il progetto aiuta a costruire il necessario contesto culturale affinché gli studenti della scuola possano comprendere che il settore ICT consentirà loro di poter avere successo nei vari impegni futuri in ambito accademico (nell'istruzione terziaria o altra formazione continua in informatica) o professionale (in attività professionali legate all'informatica). CODING4GIRLS affronta anche lo sviluppo delle competenze degli insegnanti ed il profilo della professione di insegnante consentendo agli educatori di sviluppare in modo efficace le

abilità informatiche attese per i loro studenti. Consente inoltre agli educatori di condurre iniziative che promuovono prospettive di parità di genere nel perseguimento di percorsi accademici o professionali nel campo della scienza.

Questa guida utente per gli insegnanti fa parte degli output intellettuali sviluppati nell'ambito del progetto. Gli insegnanti troveranno informazioni su concetti principali basati sulla metodologia Coding4Girls - apprendimento basato sul gioco, *serious games*, pensiero progettuale; posizione delle ragazze nell'ICT e la loro percezione del *serious game*; caratteristiche dell'ambiente di gioco Coding4Girls con due componenti: parte insegnante e parte studente; revisione degli approcci di apprendimento basati sul gioco per l'insegnamento della programmazione ed esempi di utili ambienti di programmazione.

La guida si occupa anche delle schede relative alle lezioni di apprendimento e del loro adattamento secondo la metodologia Coding4Girls e l'ambiente di gioco sviluppato.

2. CONCETTI DI BASE

2.1. Serious game

Il gioco è un concetto ampio, il che rende difficile sottolineare le caratteristiche più importanti che fanno di una certa attività un gioco. Per Thorton la più importante caratteristica del gioco è l'interattività. Johnston vede il gioco come un'attività che ha regole a cui tutti i partecipanti devono obbedire, uno o più obiettivi, interazione e immagini dinamiche (Johnston & de Felix, 1993). Malone (Malone T. , 1981) nella sua teoria rileva come fantasia, curiosità, sfida e controllo siano le componenti più importanti di ogni gioco. La descrizione più completa è fatta da Garris et al. (Garris , Ahlers, & Driskell, 2002) quando sostiene che le component sono la competizione, sfida, interazioni sociali, conversazione e fantasia. La teoria del gioco basato sull'apprendimento di Prensky è quella più influente. Sostiene che il gioco consiste in questi elementi: regole, obiettivi e compiti, risultati e feedback, situazioni di conflitto (competizione, sfida, opposizione), interazione e quadro di fantasia (Prensky , 2001). Gli autori del libro "Serious games" definiscono il gioco come un'attività volontaria (una forma di libertà) separata dalla vita reale (mondo immaginario che può avere o meno relazione con la vita reale), che assorbe la piena attenzione del giocatore e si basa su regole costruite che tutti i giocatori devono seguire ((Michael & Chen, 2006).

Una delle caratteristiche più esposte del gioco è l'interattività e si riferisce all'interazione con il computer, l'avversario (controllato dal computer o da una risorsa umana) o con altri giocatori della squadra. I giochi per computer sono divisi per il numero di giocatori impegnati simultaneamente in un gioco ossia giocatore singolo, più giocatori e numero massiccio di giocatori. In base al genere noi conosciamo videogioco, azione, guerra, giochi di ruolo, strategici, da tavolo, sportivi, di simulazione e di avventura. I grafici sulla popolarità dei giochi per computer mostrano che i più giocati sono quelli di ruolo multiplayer di massa, giochi di strategia e azione.

Per capire veramente il potere del gioco, consideriamo prima i giochi dei bambini. Questi rappresentano una delle attività più importanti che aiutano a sviluppare abilità funzionali per tutta la vita. Attraverso il gioco, il bambino migliora le sue capacità intellettuali scoprendo i primi concetti di base del mondo reale e creando connessioni valide tra loro. Jean Piaget vede il gioco come strumento per supportare i meccanismi pscichici ala base

dell’appendimento: assimilazione di nuove informazioni nella struttura cognitiva esistente e loro accomodamento al fine di consolidare quanto appena appreso. Freud enfatizza i benefici emozionali del gioco sostenendo che riduce l'ansia oggettiva e istintiva, aiutando il bambino a risolvere i problemi emotivi. Il costruttivismo sociale afferma che il gioco sia importante poiché supporta lo sviluppo delle abilità sociali.

Imparare giocando è una delle attività più comuni in tenera età, ma gli effetti positivi del giocare e i giochi sull’apprendimento sono spesso trascurati dall’educazione formale. Sono state condotte molte ricerche nel campo dell'apprendimento basato sul gioco le qualli hanno dimostrato che, se il materiale didattico è presentato in un formato di gioco per computer, ha effetti positivi sulla motivazione. Il gioco può aiutare gli studenti ad attirare l'attenzione sul materiale di apprendimento ed allo stesso tempo è una fonte di divertimento e di piacere. Avere queste due componenti nell’ambito di un processo di apprendimento permette di avere uno studente rilassato e altamente motivato ed è, quindi, maggiormente disposto a imparare. Altri vantaggi pedagogici sono l'apprendimento collaborativo (capacità espresse in giochi che prevedono più giocatori contemporaneamente), l'apprendimento sperimentale e attivo, l'apprendimento basato sui problemi e le attività basate sulla vita autentica. L'uso delle TIC è oggigiorno comune nell'educazione formale e ciò ci dà l'opportunità di sviluppare materiali di qualità al fine di promuovere l'apprendimento.

I serious games di solito si riferiscono a giochi utilizzati per la formazione, la pubblicità, la simulazione o l'istruzione. Consentono agli studenti di sperimentare situazioni impossibili nel mondo reale per diversi motivi, come sicurezza, costo o tempo. Si suppone che abbiano risultati di apprendimento definiti e si afferma che abbiano un impatto positivo sullo sviluppo di nuove conoscenze o abilità diverse da parte dei giocatori. Per progettare un buon serious game, dobbiamo essere in grado di progettare e produrre un buon software, ma i serious game sono molto più che software. Dobbiamo anche essere in grado di progettare e produrre una buona componente pedagogica. Il valore potenziale dei *serious game* per l'apprendimento sembra alto, ma permane la resistenza all'uso dei giochi in classe. Un modo ragionevole per convincere più insegnanti a provare i giochi è attraverso la dimostrazione del valore pedagogicoa, collegando gli elementi progettuali dei giochi esistenti con il processo di apprendimento e le teorie didattiche accettate.

Rieber, Smith e Noah (Rieber , Smith , & Noah, 1998) affermato già nel 1998 che ci sono due distinte applicazioni dei giochi nell'educazione: giocare ad un gioco e la progettazione di un gioco. Giocare ad un gioco è il tradizionale dove si forniscono giochi già pronti agli studenti. La progettazione di un gioco presume che l'atto di costruire un gioco sia esso stesso un percorso di apprendimento, indipendentemente dal fatto che il gioco risulti interessante o meno per altre persone. L'idea di "imparare progettando" si basa sul presupposto che la partecipazione attiva al processo di progettazione e sviluppo sia il modo migliore per imparare qualcosa. Questo approccio ha acquisito maggiore importanza a causa della proliferazione di strumenti di progettazione e creazione basati su computer.

2.2. Apprendimneto basato sul Gioco

Ci sono molte ragioni richiamano l'attenzione dell'educatore sui giochi (Gee, 2007). Nell'istruzione formale sperimentiamo un passaggio dal modello didattico tradizionale, incentrato sull'istruzione, a un modello incentrato sul discente che enfatizza il ruolo attivo del discente. Abbiamo anche cambiato la visione degli obiettivi di apprendimento da livelli tassonomici più bassi, come il semplice richiamo di informazioni, a livelli più alti, come la ricerca e l'utilizzo di informazioni in nuove impostazioni (Rugelj, Serious computer games design for active learning in teacher education, 2016).

Prensky (Prensky , 2001), Gee (Gee, 2003), e Whitton (Whitton, 2009) definiscono il gioco basato sull'apprendimento come un processo di apprendimento con l'uso dei giochi digitali. I giochi possono fornire la motivazione ad apprendere, aumentando così la possibilità che vengano raggiunti i risultati di apprendimento attesi. L'apprendimento è definito come l'acquisizione o la modifica di conoscenze, abilità attraverso l'esperienza o la pratica, e quale modo migliore per apprendere se non attraverso un gioco (Pivec & Kearney, 2007), (Pivec, Koubek, & Dondi, 2004). Quasi tutti gli studi sull'apprendimento basato sul gioco mostrano che gli studenti sono altamente motivati quando i materiali di apprendimento sono presentati in un formato di gioco per computer e che questo ha effetti positivi sull'accelerazione di un processo di apprendimento (Zapušek & Rugelj, 2013). Gli studenti hanno bisogno di motivazione per concentrarsi su ciò che deve essere appreso, ma per qualsiasi apprendimento di qualità ciò non è sufficiente. Confrontando i risultati di apprendimento degli studenti che hanno imparato attraverso i giochi per computer e quelli

che hanno imparato con un altro tipo di materiale di apprendimento mostra che non vi è alcuna differenza significativa tra loro. E' importante la progettazione appropriata del gioco. I giochi possono essere molto attraenti per gli studenti, ma se intrattengono e non insegnano, l'uso dei giochi nell'istruzione non ha molto senso. Dobbiamo individuare quali siano gli elementi che rendono un gioco da computer un gioco educativo.

Gross (Gross, 2003) sostiene che i giochi digitali per scopi educativi devono avere obiettivi di apprendimento ben definiti e devono promuovere lo sviluppo di strategie e abilità importanti per aumentare le capacità cognitive e intellettuali degli studenti.

Secondo Malone (Malone T. W., 1981b) e Garris et al. (Garris , Ahlers, & Driskell, 2002), gli elementi che contribuiscono ai valori educativi dei giochi digitali sono stimoli sensuali (rappresentazioni visive e audio del materiale di apprendimento), fantasia (contesto presentato in un ambiente immaginario), sfida (situazione impegnativa o stimolante) e curiosità (desiderio di sapere o apprendere). Questi elementi devono essere incorporati in una piattaforma integrata, strutturare obiettivi e ruoli, un contesto di apprendimento significativo, una storia accattivante, feedback immediato, un alto livello di interattività, sfida e competizione, elementi casuali di sorpresa e ambienti ricchi per l'apprendimento.

Un gioco può essere esemplificativo per l'apprendimento poiché implica stimolazione mentale (e talvolta fisica) e sviluppa abilità pratiche - costringe il giocatore a decidere, a scegliere, a definire le priorità, a risolvere i problemi, ecc. La ricompensa immediata (e il feedback) è importante fattore motivazionale, può essere tradotto come entità di gioco (più forza vitale, accesso a nuovi livelli, ecc.) o come impulsi neurologici (felicità, sensazione di realizzazione, ecc.). I giochi possono essere ambienti sociali, a volte coinvolgendo grandi comunità distribuite. Implicano (Baptista & Vaz de Carvalho, 2010) abilità autoapprese (ai giocatori viene spesso richiesto di cercare informazioni per padroneggiare il gioco stesso), consentire il trasferimento dell'apprendimento da altre realtà e sono intrinsecamente esperienziali con il coinvolgimento di più sensi.

Van Eck (van Eck, 2006) sostiene che i giochi ed il giocare possono essere ambienti di apprendimento efficaci non perché sono divertenti ma perché sono coinvolgenti, richiedono al giocatore di prendere frequenti decisioni importanti, avere obiettivi intelligenti, adattarsi a ciascun giocatore individualmente e coinvolgere i social network.

Se consideriamo il modello di apprendimento basato sul gioco di Garris, Ahlers, and Driskell (Garris , Ahlers, & Driskell, 2002), la principale caratteristica dei giochi educativi è

che il contenuto didattico è sfocato all'interno delle caratteristiche del gioco. Gli studenti stanno giocando e si divertono, dimenticando la parte di "apprendimento" dell'esperienza anche se vengono costantemente presentati nuovi concetti che devono adattare per avere successo nel gioco. Si dovrebbe promuovere la motivazione già in fase di programmazione con la ripetizione dei cicli nel contesto del gioco. Ci si aspetta che il giocatore susciti comportamenti attesi in base a reazioni emotive e cognitive che derivano dall'interazione e dal feedback del gioco.

Gee (Gee, 2003) sostiene che le caratteristiche dei videogiochi con un alto potenziale di apprendimento rientrano in due categorie: le caratteristiche 'non di gioco', che possono anche apparire in contesti non di gioco, e le caratteristiche di 'gioco', che si riferiscono più alla 'giocosità' dei giochi. Nonostante questa distinzione, non si dovrebbe presumere che le funzionalità 'non di gioco' funzionerebbero anche per l'apprendimento se fossero staccate dal 'gioco'.

Le caratteristiche 'non di gioco' dei giochi con alto potenziale di apprendimento sono:

- Empatia per sistemi complessi. Guardare un sistema complesso dall'interno per capire come interagiscono le sue variabili;
- Simulazioni di esperienza e preparazione per le azioni. Nei videogiochi i giocatori vedono il mondo virtuale in termini di come offre il tipo di azioni che devono intraprendere per raggiungere l'obiettivo di vincere;
- Intelligenza distribuita tramite la creazione di strumenti intelligenti. I buoni videogiochi distribuiscono intelligenza tra una persona del mondo reale e personaggi virtuali artificialmente intelligenti per rappresentare la visione macro e micro della situazione;
- Lavoro di squadra interfunzionale. I buoni videogiochi possono essere in grado di insegnare la collaborazione e il lavoro di squadra interfunzionale per istituzioni come scuole e luoghi di lavoro. I giocatori interagiscono tra loro non in termini di caratteristiche del mondo reale, ma attraverso le loro identità di gioco funzionali. Possono anche scegliere di utilizzare l'etnia, la classe, la cultura e il genere del mondo reale come risorse strategiche, ma non sono costretti a categorie predefinite;
- Significato contestualizzato.

Il dialogo e l'esperienza sono essenziali affinché le persone siano in grado di mettere in relazione le parole con le azioni, le funzioni e la risoluzione dei problemi reali. Poiché i videogiochi sono simulazioni di esperienza, possono collocare il linguaggio in contesti specifici;

- Fondere il personale e il sociale. Nei buoni giochi aperti, i giocatori costruiscono i propri obiettivi che si basano sui desideri, stili e cultura. La combinazione di obiettivi personali e di gioco produce uno stato di forte motivazione.

Le caratteristiche di un "buon gioco" che si riferisce a un apprendimento efficace sono le seguenti:

- Motivazione
- I videogiochi motivano profondamente i giocatori ed è importante comprendere le fonti di questa motivazione se si vuole fornire una base per l'apprendimento;
- Il ruolo del fallimento.
- Il prezzo del fallimento viene abbassato ed è spesso visto come un modo per apprendere il modello sottostante. Queste caratteristiche di fallimento nei giochi consentono ai giocatori di correre rischi che potrebbero essere troppo costosi;
- Competizione e collaborazione.
- Molti giocatori, compresi i giovani, amano la competizione con altri giocatori nei giochi, ma potrebbero non vedere la competizione come piacevole e motivante a scuola. La concorrenza nei videogiochi è vista dai giocatori come sociale, tanto quanto il gioco nel vincere e perdere.

La programmazione dei giochi si riferisce a:

- interattività – il giocatore non si limita a consumare passivamente la conoscenza, ma ha il controllo sui contenuti;
- personalizzazione – basata su stili di apprendimento e fornendo più percorsi per il successo;
- identità forti - I buoni giochi offrono ai giocatori identità che innescano un profondo investimento da parte del giocatore e che sono chiaramente associate alle funzioni, abilità e obiettivi che uno deve portare a termine nel mondo virtuale;

- problemi ben ordinati – in qualche approccio all'apprendimento, si sostiene che il sequenziamento sia cruciale per un apprendimento efficace in domini complessi;
- un piacevole livello di frustrazione - adattare le sfide in modo tale che una vasta gamma di giocatori possa sperimentare il gioco come impegnativo ma fattibile;
- un ciclo di competenze – cicli ripetuti di pratica estesa e prove di padronanza;
- 'profondo' e 'equo' – il gioco deve essere impegnativo, ma impostato per avere successo.

Gli elementi di gioco dovrebbero essere inizialmente semplici e facili da imparare e diventare più complessi man mano che il giocatore arriva a padroneggiarli.

È dimostrato che i giochi per computer utilizzati in ambito educativo aumentano la motivazione. La motivazione sa sola non è sufficiente affinché avvenga un apprendimento di qualità. Abbiamo anche bisogno di buoni materiali di apprendimento in modo che gli studenti acquisiscano effettivamente nuove conoscenze dai materiali presentati in una forma di gioco per computer.

Nonostante l'influenza positiva del gioco sulla motivazione, gli insegnanti esitano ancora a utilizzare *serious games* nell'insegnamento. Come hanno sottolineato nella nostra indagine, i giochi possono richiedere troppo tempo per essere utilizzati in una classe. Pertanto, spesso li usano come attività di apprendimento a casa o come motivazione nelle lezioni introduttive.

2.3. Metofìdologia del Design thinking

Le idee del design thinking sono emerse alla fine degli anni '60, ma come concetto, il "design thinking" è apparso alla Stanford University alla fine degli anni '80 e '90 s. Nel mondo degli affari, il metodo è diventato molto diffuso dopo essere stato promosso da Tim Brown di IDEO, che ha detto: "Il design thinking è un approccio all'innovazione incentrato sull'uomo che attinge dagli strumenti del designer per integrare le esigenze delle persone, le capacità tecnologiche e le esigenze di successo aziendale" (IDEO Design thinking, 2008). All'inizio del 21° secolo, il pensiero progettuale è diventato estremamente popolare e le aziende IT hanno iniziato ad applicare l'approccio allo sviluppo dei loro prodotti. Dal 2005, ad esempio, SAP ha applicato il *design thinking* "come filosofia di *problem solving* e come

approccio innovativo incentrato sull'utente finale" (Innovation Starter, 2015), come risultato, lo sviluppo di nuovi prodotti porta al risultato desiderato. Aziende come P&G, IBM Design, GOOGLE, AMAZON e Cisco integrano il *design thinking* nella loro organizzazione (Naiman, 2019)

Il concetto sviluppato dalla Stanford University identifica il pensiero progettuale come "una metodologia importante per la risoluzione dei problemi creativi e la creazione di innovazione ed è utilizzata quotidianamente da migliaia di aziende e organizzazioni in tutto il mondo. L'obiettivo è sviluppare negli adolescenti le capacità del 21 ° secolo: lavoro di squadra, problem solving, creatività, empatia, fiducia, pazienza, concentrazione, sperimentazione" (Karakehayova, 2019).

Il modello di Stanford consiste in 5 fasi (Fig. 2.1.)

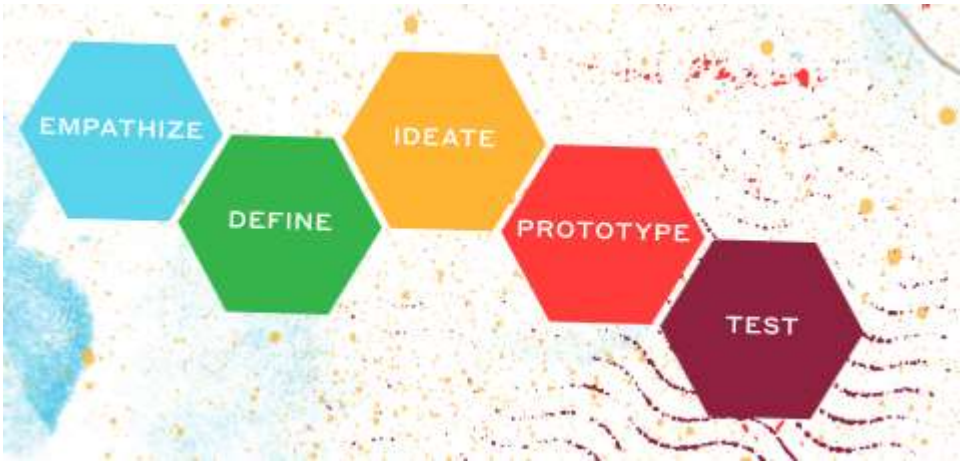


Fig. 2.1. Le fasi del modello di design thinking elaborato dalla Stanford University (Doorley, Holcomb, Klebahn, Segovia, & Utley, 2018)

Le fasi e le caratteristiche principali del design thinking potrebbero essere riassunte come di seguito (Tabella 2.1.) *(Georgieva & Tuparova, 2020)*

Tabella 2.1. Fasi e caratteristiche del design thinking basate su (Innovation Starter, 2015), (Naiman, 2019), (Un progetto dell'iniziativa K-12 presso la Stanford University, 2009)

Pratica	**Principi**	**Fasi**		
		Processo breve	**Processo esteso**	**Scuola di Stanford**
Sviluppare una comprensione profondamente empatica delle esigenze degli utenti	Scoperta - affrontare una nuova sfida. Come risolverla?	Osservare e studiare il comportamento e l'esperienza degli utenti, ricercare e definire un problema ed il punto di vista attraverso l'empatia.		Empatizzare

Formare il gruppo eterogeneo	Interpretazione - Cosa ho imparato e come interpretarlo?	Generare molte idee in un breve periodo di tempo		Definire
Conversazioni basate sul dialogo	Ideazione - vedo un'opportunità, come trasformarla in un'idea?	Selezionare e classificare le idee		Ideare
Generare scelte attraverso la sperimentazione	Sperimentazione – Ho un'idea di come costruirlo?	Realizzare il prototipo - un prodotto che riceve un riscontro rapido da parte degli utenti		Realizzare il prototipo
Usare un processo strutturato e facilitato	Evoluzione - Ho provato qualcosa di nuovo, come svilupparlo?	Testare - come migliorare il prototipo, se necessario		Testare
			Verificare se l'idea funziona	
			Apprendimento e miglioramento in base al riscontro	

Il pensiero progettuale va a vantaggio non solo delle aziende, ma anche dei settori non governativo, educativo e sanitario. Sempre più istituzioni educative vi si stanno rivolgendo. Gradualmente, le buone pratiche per la sua applicazione nell'istruzione sono in aumento. (Georgieva & Tuparova, 2020). Nell'implementazione dell'approccio del pensiero progettuale l'insegnante deve stabilire una connessione tra le singole fasi del modello di pensiero progettuale e i metodi di insegnamento appropriati che possono essere applicati. Il design thinking richiede l'applicazione di un'ampia gamma di metodi di insegnamento.

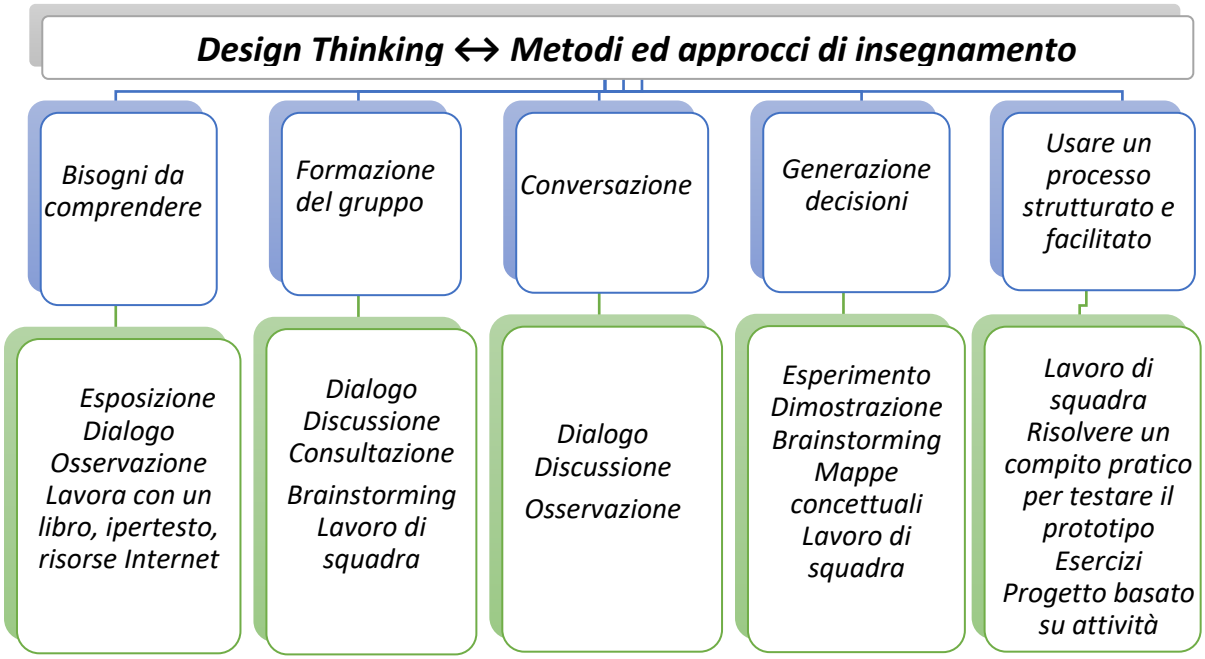


Fig. 2.2. Metodi di insegnamento e pensiero progettuale (Georgieva & Tuparova, 2020)

2.4. Teorie di apprendimento e apprendimento basato sul gioco

In passato molti giochi per computer educativi sono stati progettati secondo la teoria comportamentista dell'apprendimento (Rugelj & Lapina, 2019). Sono stati implementati sotto forma di istruzioni programmate. Agli studenti è stato offerto uno stimolo sotto forma di una domanda o qualsiasi altro tipo di compito o problema da risolvere. Gli studenti rispondono immediatamente selezionando una delle risposte offerte. Se la risposta è corretta, il gioco fornisce una sorta di risposta positiva sotto forma di reazione positiva del personaggio o melodia felice che stimola emozioni positive. Questo esempio di Stimolo-Risposta (S-R) si basa sulla connessione domanda-risposta corretta. In caso di risposta sbagliata, viene fornito un riscontro sotto forma di stimolo negativo e la connessione viene indebolita. I giochi basati sui quiz sono rappresentativi di giochi basati sul comportamentismo. Sono adatti per apprendere operazioni aritmetiche di base o per supportare la memorizzazione di dati, ovvero obiettivi di apprendimento relativi ai livelli più bassi della tassonomia di Bloom.

La teoria dell'apprendimento cognitivo enfatizza l'attività cognitiva dello studente e la formazione di modelli mentali appropriati. Apprendono concetti fondamentali dal suo insegnante o formano i saperi utilizzando la deduzione logica per acquisire nuove

conoscenze. Puzzle e giochi di strategia sono rappresentativi dell'approccio cognitivista. Le forme più avanzate di giochi basati sulla teoria cognitivista si basano su sistemi di tutoraggio intelligenti, in cui vengono impiegati algoritmi di apprendimento automatico per modellare le conoscenze degli studenti e degli esperti al fine di fornire materiale di apprendimento personalizzato.

Il costruttivismo ha una visione alternativa secondo la quale gli studenti costruiscono la propria conoscenza: una serie di rappresentazioni della conoscenza costruite individualmente, tutte ugualmente valide. L'apprendimento è un processo attivo di costruzione (piuttosto l'acquisizione di conoscenza), basato sulla conoscenza che l'utente già possiede. In un processo di costruzione, i dati sensoriali vengono combinati con la conoscenza esistente al fine di creare nuovi modelli mentali praticabili che sono a loro volta la base per ulteriori costruzioni.

L'apprendimento costruttivista enfatizza la scoperta e l'apprendimento basato sull'indagine, sostenendo che gli studenti dovrebbero essere collocati in un ambiente (che può essere modellato con un gioco per computer) dove costruiscono la propria conoscenza. Tre principi fondamentali definiscono la visione costruttivista dell'apprendimento:

- ogni persona forma la propria rappresentazione della conoscenza;
- l'apprendimento si verifica quando l'esplorazione degli studenti scopre un'incongruenza tra la loro attuale rappresentazione della conoscenza e la loro esperienza;
- l'apprendimento avviene in un contesto sociale e l'interazione tra gli studenti e i loro pari è una parte necessaria del processo di apprendimento.

I materiali di apprendimento forniscono istruzioni che consistono nel supportare la costruzione della conoscenza piuttosto che espicitare i saperi sulla base del paradigma teorico comportamentista. Le simulazioni di giochi per computer replicano vari scenari di vita reale nel formato di gioco. Presentano un modello di realtà astratta in cui lo studente riveste un certo ruolo. Il compito dell'insegnante è fornire guida e dare un riscontro durante la fase di apprendimento dello studente.

I giochi possono portare a cambiamenti in atteggiamenti, comportamenti ed abilità del giocatore. Shute e Ke (Shute & Ke, Games, Learning, and Assessment, 2012) hanno scoperto che esiste una convergenza tra gli elementi centrali di un buon gioco e le caratteristiche dell'apprendimento produttivo. Il *game design* ha molto da insegnarci sull'apprendimento e

la teoria dell'apprendimento contemporanea può insegnarci a progettare giochi migliori (Shute, Rieber, & Van Eck, 2012). Marshall McLuhan, filosofo canadese della teoria della comunicazione, ha previsto il World Wide Web negli anni Sessanta del secolo precedente, quando ha parlato di villaggio globale ha affermato: "Chiunque faccia una distinzione tra giochi e apprendimento non sa cosa siano." (Shute, Rieber, & Van Eck, 2012)

Whitton e Moseley (Whitton & Moseley, 2012) hanno proposto un quadro per le buone pratiche nella progettazione di *serious games* a una prospettiva di apprendimento attivo. Secondo le sue linee guida, l'ambiente di gioco dovrebbe supportare l'apprendimento attivo incoraggiando l'esplorazione, la risoluzione dei problemi e l'indagine, generare coinvolgimento con obiettivi espliciti e raggiungibili, essere appropriato per il contesto di apprendimento, supportare e fornire opportunità di riflessione, fornire pari opportunità a tutti gli studenti , fornire supporto continuo con una graduale introduzione di complessità crescente ed essere supportati con aiuto o suggerimenti.

Diversi studi hanno dimostrato che i giochi seri possono supportare l'apprendimento con motivazione, impegno e divertimento (Hijon-Neira, Velazquez-Iturbide, Pizarro-Romero, & Carrico, 2015). La programmazione dell'apprendimento richiede molte competenze come il pensiero logico, la risoluzione dei problemi e la capacità di comprendere concetti astratti. Per questo motivo, molti studenti trovano difficile imparare la programmazione per computer. Questo fatto può portare a una bassa motivazione a partecipare a corsi introduttivi di programmazione. Al fine di supportare la motivazione e migliorare l'atteggiamento degli studenti nei confronti dello studio della programmazione, gli insegnanti sono alla ricerca di approcci stimolanti all'apprendimento.

La programmazione si apprende meglio con la pratica e, se gli studenti vogliono imparare in modo efficace, almeno una parte di questa pratica dovrà essere auto-appresa o sviluppata in collaborazione con i colleghi. Il ruolo chiave dell'insegnante è motivare gli studenti a farlo (Feldgen & Clua, 2004).

Gli studenti del corso introduttivo di programmazione informatica progettano e sviluppano programmi (Rugelj & Lapina, 2019). Questo è un tipico approccio di apprendimento attivo: se introduciamo il lavoro di progetto in un gruppo, che si è dimostrato molto efficace ad apprendere la programmazione (Nančovska Šerbec, Kaučič, & Rugelj, 2008), possiamo effettivamente parlare del principio di apprendimento trialogico. Il piano di apprendimento trialogico consiste in forme di apprendimento in cui gli studenti

sviluppano, modificano o creano in modo collaborativo artefatti comuni (ad esempio un programma per computer nel nostro caso) in un processo sistematico (Kafai, 1995). Si concentra sull'interazione che avviene con la creazione di artefatti concreti, non solo tra le persone ("approccio dialogico"), o all'interno della propria mente (approccio "monologico") (Paavola & Hakkarainen, 2005).

3. RAGAZZE, ICT E SERIOUS GAME

3.1. Posizione delle donne nel settore ICT– stato dell'arte

La società digitale pone esigenze crescenti sulle competenze dei popoli moderni. Vi è un crescente bisogno di persone che non solo siano alfabetizzate digitalmente, ma abbiano anche le capacità per creare e mantenere applicazioni software in vari campi dell'attività umana. I media commentano costantemente la carenza di specialisti qualificati nel settore delle tecnologie dell'informazione e della comunicazione (ICT). C'è una crescita continua di questo settore.

Nel 2019, circa 7.8 milioni di persone lavoravano come specialist TIC nell'Unione Europea (EU) (Eurostat, 2020). Essi rappresentano solo il 3,9% di tutti gli impiegati in EU..

Secondo i dati (Eurostat, 2020) nell'UE-27 nel 2011 le donne nel settore IT erano il 17% e nel 2019 - circa il 17,9%. Per il periodo 2007-2019 in media per i 27 Stati membri dell'UE, le donne nel settore IT sono diminuite dal 22,5% al 17,9%. Nel 2019, le donne sono il 17,3% del numero totale di persone occupate nell'UE con un'istruzione ICT, in calo dal 20,2% nel 2009 (Eurostat, 2020).

Lo stato attuale dell'impiego delle donne nel settore ICT nei paesi partner del progetto Coding4girls viene presentato nella Fig. 3.1.

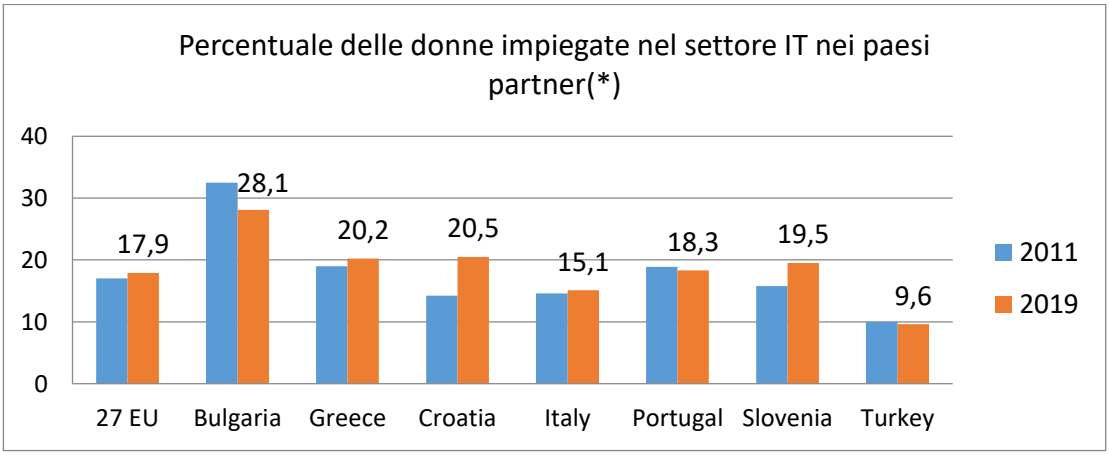


Fig. 3.1. Donne impiegate nel settore TIC (in percentuale) nel 2011 e 2019

3.2. Attitudini delle ragazze ai serious game

I risultati dello studio condotto da Vermeulen et al. (Vermeulen, Van Looy, Courtois, & De Grove, 2011) hanno mostrato che le differenze di genere erano costantemente presenti, ma l'esperienza precedente ha influenzato sostanzialmente i risultati (Rugelj & Lapina, 2019). Le donne giocano più frequentemente ma per periodi di tempo più brevi rispetto agli uomini. Preferiscono giochi astratti, brevi e facili da padroneggiare come *casual* (ad esempio Tetris) e giochi di *social network*, mentre gli uomini hanno maggiori probabilità di giocare a generi *core*. Questi ultimi sono basati sulle abilità che richiedono molto tempo e generalmente presentano una grafica tridimensionale di alta qualità come sparatutto, combattimento, azione-avventura, sport, corse, strategia, giochi di ruolo e giochi MMO. I generi *non-core* includono piattaforma, avventura, simulazione, feste, *serious*, giochi classici e *casual*. I generi *core* sono giocati più dai maschi che dalle femmine. Lo studio ha scoperto che le donne amano i giochi di puzzle, mentre le corse, il ritmo, la simulazione e i giochi virtuali sono giocati sia da donne che da uomini. Un altro studio ha scoperto che le donne preferiscono i giochi di società (come giochi di musica e danza) e i classici giochi 'retrò'.

Alserri et al (Alserri, Zin, & Wook, 2018) hanno condotto un'ampia rassegna di articoli correlati su elementi di *serious game* efficaci come elementi motivazionali di giocatori di giochi digitali, elementi di giochi educativi efficaci e elementi di preferenza femminile. Hanno utilizzato i risultati per creare un modello concettuale per il coinvolgimento di genere *serious games*. Usiamo questo modello nel progetto Coding4Girls al fine di aumentare la motivazione delle ragazze verso la programmazione attraverso la selezione appropriata di giochi che saranno progettati e implementati dagli studenti nell'ambito dell'apprendimento basato sul *game design*.

Gli autori hanno affermato che lo studio ha rivelato che la motivazione a giocare a un tipo specifico di gioco digitale dipende dal genere. Lo stereotipo secondo cui le donne non giocano ai videogiochi non è più valido. Le differenze nelle preferenze di genere per i giochi digitali, scoperte in questa ricerca, sono molto simili alle differenze già presentate, identificate da Vermeulen et al. (Vermeulen, Van Looy, Courtois, & De Grove, 2011). Le femmine preferiscono il gameplay esplorativo e creativo più dei maschi. Giocano più frequentemente ma per meno tempo rispetto agli uomini e le loro preferenze per i generi di gioco sono diverse: le donne preferiscono i puzzle, giochi sociali con ricompense offerte,

giochi educativi, simulazioni, generi collaborativi e di esplorazione, vita virtuale, mondo virtuale. Gli elementi di preferenza della motivazione ai giochi sono sfida, evasione, divertimento, interazione sociale, motivazione, fantasia, competizione e eccitazione.

Phan et al. (Phan, Jardina, Hoyle, & Chaparro, 2012) sono giunti a conclusioni molto simili nel loro studio sulle differenze di genere tra giocatori maschi e femmine in termini di utilizzo, preferenza e comportamento del gioco per computer.

Secondo lo studio di Tuparova at all. (Tuparova, Tuparov, & Veleva, 2019 b), si registrano differenze nelle preferenze di ragazze e ragazzi in merito alle caratteristiche ed agli elementi dei giochi come: grafica, suono, possibilità di giocare con differenti strumenti (computer, tablet, smart phone, ecc.), possibilità di coinvolgere moti giocatori, possibilità di giocare on-line, possibilità di passare a livelli di gioco più complessi. La caratteristica preferita per le ragazze è la *possibilità di passare a livelli di gioco più complessi* mentre per i ragazzi è la *logica e trama / scenario del gioco.* La caratteristica di minore interesse per le ragazze è il suono nei giochi mentre per i ragazzi sono i premi. Per quanto riguarda il dispositivo di utilizzo lo smartphone è quello più utilizzato sia dai ragazzi che dalle ragazze. Invece, il secondo dispositivo è il laptop per le ragazze seguito da un tablet mentre per i ragazzi il secondo è il computer seguito dal laptop. Gli autori hanno osservato che i ragazzi mostrano una maggiore preferenza per i giochi di storia rispetto alle ragazze mentre le ragazze per i giochi di memoria. Indipendentemente dal sesso si registra la preferenza dei seguenti giochi: giochi con elementi sociali, attenzione / concentrazione, giochi per velocità di reazione, giochi di ruolo, giochi d'avventura. Il tipo di gioco con il minor interesse è rappresentato dal puzzle.

Dobbiamo prendere in considerazione la situazione specifica nel progetto Coding4Girls: si fonda sull'apprendimento basato sul design del gioco e deve costruire compiti che siano interessanti e motivanti per le ragazze. L’ambiente educativo presenta la programmazione come un mezzo per creare film d'animazione (cioè narrazione) o giochi semplici perché le ragazze di loro possono sviluppare l'idea per una storia o un semplice gioco che vorrebbero creare (Wolz, Barnes, & Bayliss, 2009). Entrambe le attività sono naturalmente sequenziale ed è improbabile che richiedano immediatamente concetti di programmazione avanzati. Sono una forma di autoespressione e offrono l'opportunità di sperimentare ruoli diversi, gli amici che non programmano possono capire e apprezzare prontamente una storia o un gioco animati nonché fornire un'opportunità per un riscontro positivo. Kelleher et all

(Kelleher , Pausch, & Kiesler, 2007) usarono rispettivamente *Storytelling Alice* e *Generic Alice* per entrambi i casi. Affermano che in diversi studi su bambini programmatori è emerso che quando ragazze e ragazzi hanno un'esperienza simile con la programmazione di computer sono ugualmente interessati e l'apprendimento della programmazione è efficace. Le prestazioni di programmazione sono correlate alla quantità di tempo che gli utenti hanno trascorso a programmare ed alla loro precedente esperienza di programmazione.

4. INSEGNAMENTO DELLA PROGRAMMAZIONE ATTRAVERSO I GIOCHI

Questa sezione si concentra sulla presentazione ed il confronto di diversi approcci, piattaforme ed ambienti di gioco che possono essere utilizzati per insegnare le abilità di programmazione ai bambini.

4.1. Gli approcci a supporto dell'insegnamento della programmazione attraverso i giochi

There is currently a wide variety of approaches for teaching programming to novice coders.

Attualmente esiste un'ampia varietà di approcci per insegnare la programmazione a programmatori inesperti.

Gli approcci potrebbero essere classificati come:

- apprendimento della programmazione tramite la progettazione basata sul gioco, l'uso dei linguaggi di programmazione per imparare a programmare progettando e creando giochi;
- apprendimento della programmazione in un ambiente basato sul gioco;
- apprendimento della programmazione tramite la progettazione del gioco in un ambiente basato sul gioco. La metodologia Coding4Girls appartiene a questa categoria di ambienti di apprendimento della programmazione.

Un esempio è l'uso di linguaggi di programmazione visiva basati su blocchi che sono particolarmente intuitivi, facili da usare (su base drag-and-drop), prevengono errori di sintassi e logica (Ouahbi, Kaddari, Darhmaoui, Elachqar, & Lahmine, 2015) e presentano i contenuti in modo molto modo intuitivo. Scratch (Resnick, et al., 2009), https://scratch.mit.edu/ , Snap! (Weintrop & Wilensky, 2015) https://snap.berkeley.edu/ , or Alice 2/3 https://www.alice.org/about/ sono esempi ben noti e di successo di linguaggi di programmazione visuale. Successivamente, abbiamo una tabella che confronta le caratteristiche dei più noti linguaggi di programmazione visiva (e ambienti).

Table 4.1. Caratteristiche di alcuni ambienti programmazione basati su blocchi visivi.

	PIATTAFORMA	GRUPPO DI RIFERIMENTO	LINGUAGGI	GRATIS / A PAGAMENTO
Scratch	Basato sul Web (ma con versione offline)	8-16	Linguaggio di programmazione visiva per computer	Gratuito
Snap!	Basato sul Web (ma con versione offline)	12-20	Linguaggio di programmazione visiva per computer	Gratuito
Alice	Windows, Mac OS X, Linux	12-20	Linguaggio di programmazione visiva per computer	Gratuito
Tynker	Web-based, iOS	7+	Linguaggio di programmazione visuale per computer, JavaScript, Python	A pagamento

I giochi per computer possono essere utilizzati anche per imparare a programmare, incoraggiando gli studenti a sviluppare e creare i propri videogiochi (apprendimento tramite *game design*) o consentendo loro di giocare ai *serious games* i cui risultati di apprendimento comprendono risultati di apprendimento relativi alla programmazione (apprendimento basato sul gioco). I giochi per computer proposti per attività educative hanno il potenziale per creare un ambiente di apprendimento motivante e divertente, poiché contengono attività che soddisfano standard educativi, obiettivi di apprendimento, forniscono riscontri e possono ottenere risultati educativi elevati (Georgieva & Tuparova, 2019).

L'idea alla base dell'utilizzo dei giochi per insegnare la programmazione deriva dal fatto che questi, essendo più coinvolgenti e motivazionali, consentono agli studenti di apprendere il pensiero computazionale e le abilità di programmazione in ambienti divertenti e familiari prima di trasferire tali abilità all'apprendimento di un linguaggio di programmazione (Kazimoglu, Kiernan, Bacon, & Mackinnon, 2012). Ancora secondo Kazimoglu et al. (Kazimoglu, Kiernan, Bacon, & Mackinnon, 2012), i *serious games* per l'apprendimento della programmazione e del pensiero computazionale non dovrebbero essere creati solo per divertimento, ma dovrebbero anche prendere in considerazione il curriculum e le abilità, facendo la differenza tra i diversi costrutti di programmazione e incoraggiando buone pratiche di programmazione (attraverso modalità ludiche come il raggiungimento di un punteggio alto).

I *serious games* possono anche essere usati per la programmazione trasformando operazioni spiacevoli in esperienze interessanti —Shabanah et al (Shabanah, Chen, Wechsler, Carr, & Wegman, 2010) crearono un algoritmo per la definizione del gioco mirato a facilitare la creazione di un algoritmo al fine di migliorare il coinvolgimento nei sistemi di visualizzazione degli algoritmi. Grivokostopoulou et al (Grivokostopoulou, Perikos, & Hatzilygeroudis, 2016) hanno sviluppato un gioco per insegnare algoritmi di intelligenza artificiale, basato sul gioco Pacman, in modo che attraverso le visualizzazioni e le animazioni degli algoritmi gli studenti possano essere aiutati nel loro processo di apprendimento, dimostrando il modo in cui opera un algoritmo, le sue funzioni e come prendere decisioni appropriate in base a parametri specifici.

Con la crescente adozione diffusa di velocità di connessione Internet più elevate e la crescente disponibilità di computer nelle case di tutti, le piattaforme basate sui giochi online come CodeCombat (https://codecombat.com/) e LightBot (https://lightbot.com/) hanno iniziato ad apparire. Combéfis et al. (Combéfis, Beresnevičius, & Dagiene, 2016) hanno esaminato i principali tipi di piattaforme online utilizzate per insegnare le abilità di programmazione ed hanno concluso che i seguenti fattori possono rendere un *serious game* più motivante e di successo: 1) un processo di riscontro e valutazione; 2) estetica; 3) collaborazione tra più giocatori; 4) guida attraverso il gioco; 5) nessuna conseguenza negativa; 6) musica; 7) un livello medio di sfida per mantenere l'interesse dei giocatori.

Per quanto riguarda i risultati di apprendimento dei *serious games* per apprendere la programmazione, secondo un'estesa revisione di 49 *serious games* di programmazione di Miljanovic et al (Miljanovic & Bradbury, A Review of Serious Games for Programming, 2018), la maggior parte di questi si concentrano sulla risoluzione dei problemi e sui concetti fondamentali di programmazione, con pochi giochi incentrati sulle strutture di dati, metodi di sviluppo e progettazione del software.

Ci sono anche giochi che, sebbene possano non insegnare abilità di programmazione, sono in grado di insegnare in modo indiretto, ma efficace abilità chiave del pensiero computazionale, come l'astrazione (es. Tetris), la decomposizione (es. Sudoku), il pensiero algoritmico (es. Puzzle), valutazione (ad es. Memory Games) e generalizzazione (ad es. Pattern Games) (Frankovic, Hoic-Bozic , & Nacinovic-Prskalo, 2018).

Nell'apprendimento basato sulla progettazione di giochi, spesso abbiamo la combinazione dell'uso di linguaggi di programmazione visiva e dei loro ambienti basati su

blocchi con il processo di progettazione e codifica dei giochi. In uno studio che ha confrontato l'utilizzo degli ambienti di codifica Scratch e Pascal (linguaggio procedurale tramite riga di comando o editor di testo) da parte di programmatori alle prime armi, è emerso che gli studenti erano molto più motivati a continuare i loro studi in scienze informatiche quando utilizzavano Scratch (Ouahbi, Kaddari, Darhmaoui, Elachqar, & Lahmine, 2015). Lo studio ha anche evidenziato che la creazione di giochi e storie ha reso gli studenti più creativi e autonomi durante l'apprendimento della programmazione. Un altro studio, condotto da Weintrop et al (Weintrop & Wilensky, 2015), confrontando l'uso di Snap! e Java degli studenti delle scuole superiori hanno scoperto che l'approccio basato su blocchi aveva una curva di apprendimento più semplice di Java, quindi è in accordo con la visione che la programmazione basata su blocchi (come Scratch e Snap!) è più accessibile ai programmatori alle prime armi. Secondo i risultati dello studio, ciò era dovuto al fatto che i blocchi sono più facili da leggere, a causa della natura visiva dei blocchi che forniscono spunti su come possono essere utilizzati, sono più facili da comporre e servono come aiuti per la memoria.

Viene discusso l'approccio integrato per la programmazione dell'insegnamento e l'influenza sui risultati degli studenti di 14 anni (Tuparova, Nikolova, & Tuparova, 2020). Questo approccio combina due fasi:

- “utilizzo di giochi educative esistenti per il computer - per acquisire conoscenze e abilità, motivazione delle attività di apprendimento, sviluppo del pensiero algoritmico; per informarsi sulle regole del gioco, sugli elementi dell'interfaccia e sulle loro proprietà ed eventi al fine di sperimentare il gioco;
- Progettazione e sviluppo di giochi per computer educativi, compreso il modello matematico del gioco, progettazione dell'interfaccia utente grafica (GUI); programmazione, testare e verificare.”
- Per l’implementazione dell’approccio è usato l’ambiente di programmazione C#. Il modello è testato con 126 studenti con la specializzazione in informatica, divisi in due gruppi – gruppo sperimentale e gruppo do controllo. I risultati degli studenti vengono misurati attraverso compiti pratici e test cartacei. I risultati mostrano che non c'è differenza tra i risultati di ragazzi e ragazze nel gruppo sperimentale. Ma gli autori hanno scoperto che le ragazze nel gruppo

sperimentale hanno ottenuto risultati più elevati rispetto alle ragazze nel gruppo di controllo.

4.2. Gli ambienti basati sul gioco per l'insegnamento della programmazione

Successivamente, presentiamo alcuni esempi dell'uso di diversi ambienti che utilizzano i giochi per insegnare il *coding* e la programmazione.

4.2.1. Apprendimento tramite il game-design

Scratch

Fig.4. SEQ Immagine * ARABIC 1 Esempio di testi Scratcht

Scratch (è un ambiente di programmazione basato su blocchi, ovvero consente agli studenti di assemblare programmi agganciando blocchi di codice e ricevendo feedback visivo (Weintrop & Wilensky, 2015). In questo modo, gli studenti acquisiscono familiarità con i fondamenti della programmazione senza doversi preoccupare della sintassi (Ouahbi, Kaddari, Darhmaoui, Elachqar, & Lahmine, 2015).

Scratch può essere utilizzato come introduzione alla programmazione, soprattutto tra gli studenti più giovani, ma anche come un modo per facilitare il passaggio ad altri ambienti di programmazione, introducendo e stimolando così l'interesse per l'informatica (Wolz, Maloney, & Pulimood, 2008). Secondo uno studio di ricerca condotto da Meerbaum-Salant et al. (2013), la maggior parte degli studenti può raggiungere un livello ragionevole di concetti di informatica tramite Scratch. Tuttavia, sorgono difficoltà quando si insegnano

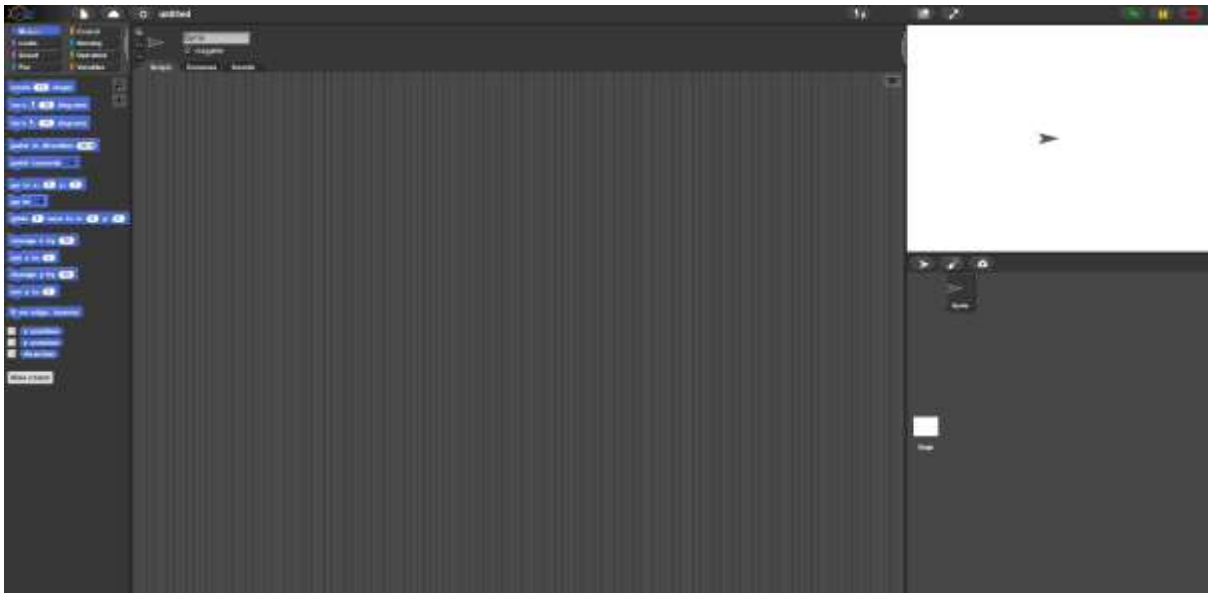
argomenti specifici che richiedono un maggiore livello di astrazione. Questo può, tuttavia, essere risolto se gli studenti sono accompagnati nel processo dall'insegnante.

Il sito web di Scratch supporta una rete mondiale di utenti, ospitando una comunità di programmatori che consente agli utenti di condividere progetti (Meerbaum-Salant , Armoni, & Ben-Ari, 2013). Scratch è disponibile in più di 50 lingue, tra cui bulgaro, croato, inglese, greco, italiano, portoghese, sloveno e turco. Ha anche un editor offline che consente di scaricare il programma su un computer e di eseguirlo senza una connessione Internet.

Snap!

Snap! (https://snap.berkeley.edu/) .) è anche un linguaggio di programmazione basato su blocchi visivi il cui design è impostato su Scratch, ma inserisce alcune funzionalità aggiuntive.

Snap!, in modo simile a Scratch, è basato sul web, ma ha anche la possibilità di essere eseguito offline tramite un browser. Oltre alle funzionalità di Scratch, Snap! aggiunge,



elenchi di classi, procedure di classe, *sprites* di classe, costumi di classe, suoni di classe e continuazioni di classe, rendendolo così più adatto a un pubblico più avanzato di Scratch. Snap! è disponibile in più di 40 lingue, inclusi bulgaro, croato, inglese, greco, italiano, portoghese, sloveno e turco.

Alice

Fig. 4. SEQ Immagine * ARABIC 3 Alice 3 Codice Editor

Alice https://www.alice.org/ è un blocco basato sull'ambiente di che mira a motivare gli studenti a programmare stimolando la loro creatività attraverso la creazione di animazioni, narrazioni interattive o semplici giochi 3D (Kelleher , Pausch, & Kiesler, 2007). Consente la creazione di mondi virtuali con un Virtual World Editor in cui gli studenti possono aggiungere oggetti 3D e aggiungere funzioni e metodi agli oggetti 3D esistenti. Una volta creato il mondo virtuale, lo studente può scrivere il codice per sviluppare la logica del gioco / narrativa / animazione (Sykes, 2007). Alice rappresenta un buon linguaggio di programmazione per studenti senza esperienze precedenti, poiché consente loro di vedere i programmi animati eseguiti mentre scrivono il loro codice (Cooper, Dann, & Pausch, 2000).

Inoltre, un altro studio incentrato su Storytelling Alice (un ambiente di programmazione basato su Alice 2 con ulteriori strumenti creativi per la scrittura di storie) ha rilevato che sebbene sia Generic Alice che Storytelling Alice abbiano avuto lo stesso successo nell'insegnare concetti di programmazione alle ragazze, con Storytelling Alice le ragazze erano più motivate e ha incrementato il tempo della programmazione e ciò ha avuto un positivo impatto sulla programmazione.

Alice 3, la puntata più recente della serie, è disponibile in 13 lingue, tra cui inglese, portoghese, greco, sloveno e bulgaro, inglese, spagnolo, portoghese e tedesco.

Tynker

Tynker https://www.tynker.com/ è una piattaforma educative di programmazione basata sul linguaggio di programmazione visuale drag-and-drop. A differenza degli altri ambienti di programmazione presentati, questo è un prodotto commerciale. Rispetto a Scratch aggiunge un modo diverso in cui si approccia alla codifica ossia come un gioco, in cui gli utenti sono tenuti a creare un programma per far muovere i personaggi, interagire ed eseguire compiti diversi. Presenta problemi che i giocatori devono risolvere in modo che i bambini inizino a riconoscere i modelli (Geist, 2016). Tynker fornisce anche un tutor integrato per dare istruzioni passo passo, in modo che lo studente possa imparare come applicare i concetti di codifica. Lo studente è anche in grado di visualizzare i blocchi in codice JavaScript, consentendo di comprendere il blocco in un linguaggio di programmazione basato su testo.

Tynker fornisce più di 23 corsi di programmazione, 11 corsi su iPad e più di 2000 attività di *coding* e fornisce piani individuali e familiari (per più di 4 membri). E' solamente disponibile in inglese.

Fig.4. 4 L'interfaccia utente Tynker

4.2.2. L'apprendimento della programmazione in ambiente basato sul gioco

Presentiamo alcune piattaforme basate su giochi pensati per sviluppare abilità di programmazione.

Code Combat

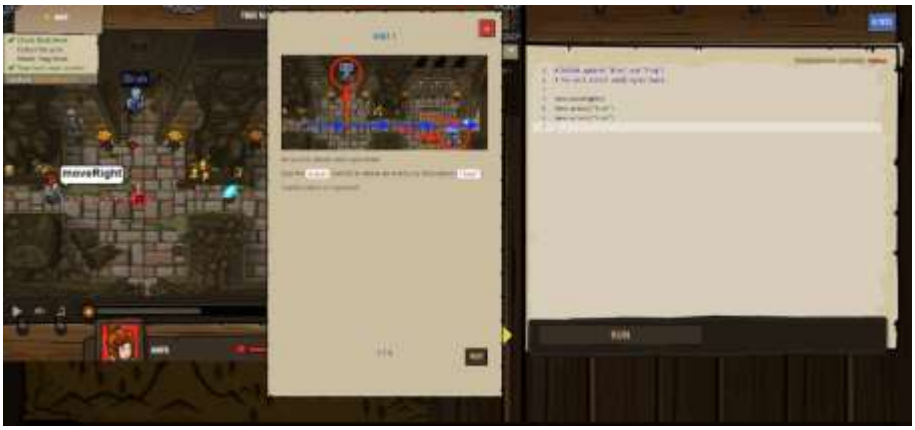


Fig. 4. SEQ Immagine * ARABIC 5 Code Combat Gameplay

Code Combat https://codecombat.com/ è un gioco di avventura basato sul web. Ha piani sia per i singoli studenti che per le classi, fornendo anche risorse per gli insegnanti da utilizzare durante le lezioni. Ha più di 100 livelli. E' disponibile in più di 50 lingue, inclusi il bulgaro, croato, inglese, greco, italiano, portoghese, sloveno e turco.

Agli studenti è richiesto di scrivere il proprio codice (in JavaScript o Python) per far muovere i personaggi, interagire e svolgere compiti diversi. Non è basato su blocchi e richiede agli studenti di scrivere il proprio codice. Fornisce suggerimenti lungo il percorso e introduce i concetti gradualmente. Il gioco è diviso in 5 mondi diversi che introducono concetti differenti man mano che il giocatore avanza nel gioco: 1) *Kithgard Dungeon* – sintassi, metodi, parametri, stringhe, cicli e variabili; 2) *Backwoods Forest* – Se/altro, logica booleana, operatori relazionali, funzioni, proprietà degli oggetti, gestione degli eventi, gestione degli input; 3) *Sarven Desert* – aritmetica, contatori, *while-loops*, interruzione, continua, matrici, confronto stringhe, ricerca di min/max; 4) *Cloudrip Mountain* – *Object literals*, metodo remoto, invocazione, *for-loops*, funzioni complesse, disegno, modulo; 5) *Kelvintaph Glacier* – tecniche avanzate.

Secondo Miljanovic et al. (Miljanovic & Bradbury, A Review of Serious Games for Programming, 2018) il contenuto educativo del gioco comprende diversi aspetti concettuali

della progettazione degli algoritmi e della risoluzione dei problemi, concetti fondamentali di programmazione (come sintassi e semantica, variabili e tipi di dati primitivi; espressioni e assegnazioni, input e output, condizionali e iterativi, funzioni e parametri, e ricorsione) e strutture di dati fondamentali (come array, aggregati eterogenei e tipi di dati astratti).

Human Resource Machine

Human Resource Machine http://tomorrowcorporation.com/humanresourcemachine è un gioco puzzle basato sul linguaggio di programmazione visuale. In questo gioco, il giocatore deve automatizzare un'attività programmando un impiegato, procedendo

attraverso enigmi sempre più difficili (Johnson & all, 2016). Il giocatore deve creare una sequenza di mosse, trascinando e rilasciando le istruzioni, con l'introduzione graduale di nuovi comandi per eseguire operazioni più complesse. Attraverso questo linguaggio di programmazione visuale si insegnano concetti fondamentali di programmazione attraverso il confronto degli algoritmi.

LightBot

LightBot https://lightbot.com/ è un gioco puzzle con meccaniche simili a Human Resource Machine, richiedendo il pensiero logico per progredire attraverso i livelli. Il giocatore deve guidare un robot per illuminare le tessere e risolvere 40 livelli. I comandi utilizzati in Lightbot vengono visualizzati come icone.

Sebbene non vi sia un apprendimento esplicito di un linguaggio di programmazione, i giocatori "... sviluppano una comprensione del sequenziamento e dell'implementazione di algoritmi" concentrandosi sulle capacità di problem solving (Miljanovic & Bradbury, A

Review of Serious Games for Programming, 2018)(p.6) Tuttavia, attraverso il gioco gli studenti imparano diversi concetti fondamentali di programmazione, come condizionali, iterativi, ricorsione e strategie di *debugging*. Poiché lo spazio per le tessere è limitato, lo studente deve anche considerare l'efficienza del codice.

Uno studio condotto da Mathrani et al (Mathrani, Christian, & Ponder-Sutton, 2016) ha usato Lightbot con gli studenti della scuola di istruzione di livello secondario e registrò che gli studenti si divertivano a giocare e che questo rappresentava un modo efficace per apprendere costrutti di programmazione come funzioni, procedure, condizionali e ricorsioni. La maggior parte degli studenti coinvolti ha anche convenuto che l'approccio era un modo efficace per comprendere concetti che altrimenti sarebbero stati più difficili da afferrare.

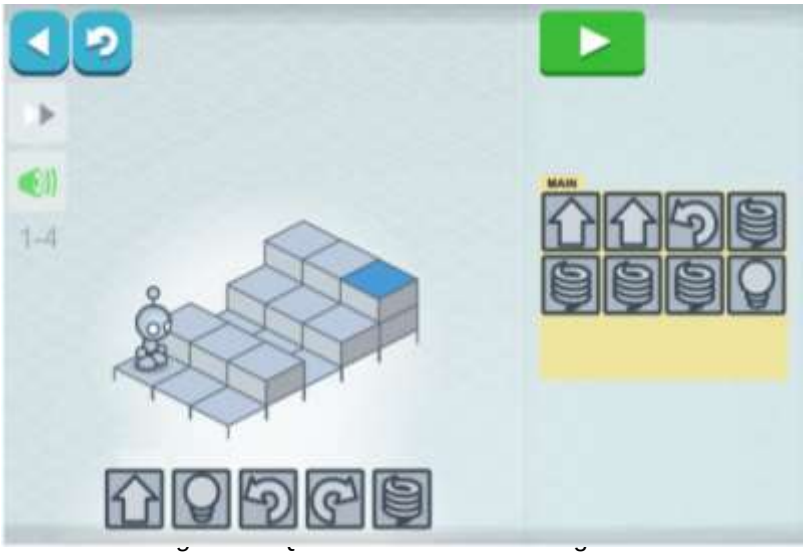


Fig. 4.1 – L'interfaccia di Lightbot

May’s Journey

May’s Journey è l'unico gioco di questo elenco rivolto direttamente alle ragazze delle scuole medie. Si tratta di un puzzle in 3D in cui i giocatori risolvono un labirinto attraverso il linguaggio di programmazione personalizzato del gioco, che si ispira a Java. Il gioco è stato

Fig. 4. SEQ Immagine * ARABIC 8 May's Journey Gameplay

progettato per attrarre le ragazze verso l'informatica insegnando le basi della programmazione. Lo sviluppatore utilizza uno pseudo-codice per facilitare la transizione tra la programmazione visiva e i linguaggi di programmazione reali. Il contenuto didattico include istruzioni di base e logica di sequenza, cicli, variabili, istruzioni *if*, comparatori e logica booleana, operazioni su numeri interi e operazioni su stringhe (Jemmali , 2016).

Ci sono due fasi nel gioco, una fase di esplorazione con le meccaniche di un gioco tipico e una fase di codifica. Nella fase di codifica, l'interfaccia è suddivisa in modo che il giocatore possa digitare il codice mentre riceve un feedback visivo del programma eseguito. Suggerimenti per il codice vengono forniti anche durante la fase di esplorazione. Nella storia, l'eroe è una ragazza che vive in un mondo che sta cadendo a pezzi ed è separata dalla sua amica che ha bisogno di aggiustare il mondo del gioco e risolvere misteri. Ogni parte del mistero viene svelata gradualmente, motivando i giocatori a giocare di più. Poiché i pre-adolescenti sono il principale gruppo target, il personaggio principale è stato rappresentato con le fattezze di una ragazza delle scuole medie, in modo che il giocatore possa immedesimarsi. La progettazione aveva individuato le preferenze delle ragazze nei confronti del design (elevata leggerezza, combinazione di colori caldi e illustrazione meno aggressiva. Anche il gruppo di prova sembrava apprezzare la trama.

No Bug's Snack Bar

No Bug's Snak Bar è un *serious game* di ricerca con un approccio basato su blocchi *drag and drop*, incentrato sulla risoluzione dei problemi. I risultati dell'apprendimento riguardano la manipolazione delle variabili, la sequenza di azioni, i condizionali e le iterative e le

Fig. 4. SEQ Immagine * ARABIC 9 No Bug's Snack Bar Gameplay

strategie di *debugging* (Vahldick, 2017).

Nel gioco, il personaggio principale lavora in uno snack bar e deve completare diverse missioni usando il codice. Un'innovazione integrata in questo gioco è rappresentata da uno strumento a disposizione degli insegnanti per monitorare il modo in cui gli studenti stanno progredendo.

Fig. 4.9 - No Bug's Snack Bar Gameplay

C'è anche un approccio ludico: lo studente guadagna punti man mano che progredisce nei giochi e trova modi per migliorare le proprie soluzioni ai problemi presentati. Il gioco incorpora un assistente con suggerimenti e un sistema amministrativo per l'insegnante supportandolo nel vedere chi sono gli studenti che hanno bisogno di aiuto in modo può inviare loro suggerimenti personalizzati. La presenza di un insegnante era, quindi, vista come fondamentale.

Robot ON!

! È un tipo di gioco basato sul puzzle di ricerca rivolto a studenti universitari che stanno imparando C++. In questo gioco, il giocatore è uno scienziato che deve attivare un "Mech Suit" attraverso una serie di compiti. Una volta che il giocatore finisce un livello, attiva un nuovo sistema robotico. Gli insegnanti possono creare i propri livelli utilizzando altri linguaggi di programmazione.

Il giocatore ha diversi strumenti, che sono codificati a colori, con il colore sul codice corrispondente ai diversi strumenti. Il giocatore viene introdotto a diversi strumenti uno alla volta, accompagnato da tutorial (Miljanovic & Bradbury, 2016).

Invece di concentrarsi sulla scrittura di un codice, questo gioco si concentra sulla comprensione della programmazione, al fine di insegnare abilità di *debug* e comprendere il codice scritto da altri. Man mano che il gioco procede, al giocatore vengono forniti i seguenti strumenti: 1) strumento attivatore (per apprendere il flusso di controllo); 2)

Fig. 4. SEQ Immagine * ARABIC 10 Robot ON! Gameplay

strumenti per 'commenter' e 'un-commenter' (per apprendere il comportamento del codice); 3) strumento 'namer' (per apprendere scopo variabile); 4) strumento di verifica (per apprendere il flusso di dati).

Gioco educativo Pacman

Il gioco educativo Pacman è un serious game di ricerca programmato per insegnare la ricerca di algoritmi, consentendo agli studenti di vedere come si comportano i diversi algoritmi di ricerca attraverso una loro rappresentazione grafica annotata (Grivokostopoulou, Perikos, & Hatzilygeroudis, 2016).

Il gioco ha due modelli:

1) modalità didattica - lo studente può leggere una descrizione testuale e il diagramma di flusso grafico e lo pseudocodice di un algoritmo di sua scelta. Il gioco consente inoltre allo studente di apprendere l'algoritmo tramite visualizzazioni sui diversi Pacman Maze;

2) modalità di gioco - lo studente deve risolvere livelli di labirinto in diverse condizioni e con un vincolo di tempo. Questi livelli sono fatti in modo che lo studente debba applicare uno specifico algoritmo di ricerca per spostare il Pacman nel labirinto - allo studente viene chiesto, da una posizione casuale, di spostare il personaggio con l'algoritmo specifico.

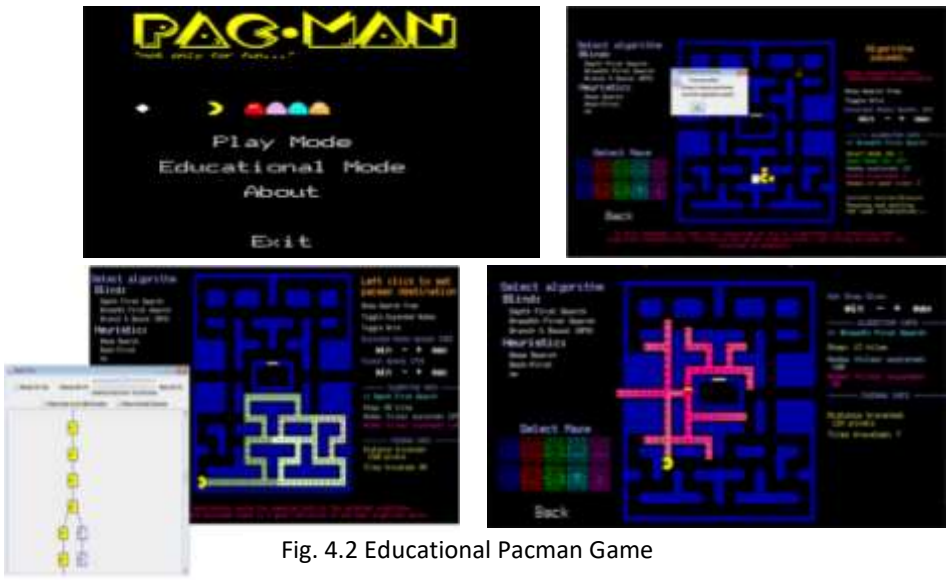


Fig. 4.2 Educational Pacman Game

CMX

CMX è un gioco di ruolo online che prevede molti giocatori (MMORPG) per l'apprendimento della programmazione. Nel gioco ci sono due squadre - cracker e hacker - che competono l'una contro l'altra per trovare le password in una fabbrica globale di rifiuti tossici. Sono assistiti da Senseis nell'apprendimento del linguaggio di programmazione C. Il gioco include tre livelli di Senseis raggiungibili sbloccando una password nel livello precedente. Gli studenti sembravano aumentare il loro livello di comprensione di C - non solo potevano rispondere a domande di teoria, ma anche creare programmi eseguibili trascinando e rilasciando strumenti, oltre a scrivere programmi in C (Malliarakis, Satratzemi, & Xinogalos, 2014).

Tabella 4.2. Confronto degli ambienti di gioco per l'insegnamento della programmazione

GIOCO	PIATTAFORMA	GRUPPO DI RIFERIMENTO	LINGUE	UNITA' DI APPRENDIMENTO	TIPO DI GIOCO
Code Combat	Web-based	9+	JavaScript, Python	sintassi, metodi, parametri, stringhe, cicli e variabili, *if / else*, logica booleana, operatori relazionali, funzioni, proprietà oggetto, gestione eventi, gestione input, aritmetica, contatori, cicli *while*, interruzione, continuità, matrici, confronto stringhe, trovare min / max, valori letterali oggetto, metodo remoto, invocazione, *for-loops*, funzioni complesse, disegno, modulo	Gioco commerciale (*gratis*)
Human Resource Machine	Windows, Mac OS X, Linux, iOS, Android	9+	basato sul linguaggio di programmazione visuale	basato sul linguaggio di programmazione visuale	basato sul linguaggio di programmazione visuale
Lightbot	iOS, Android, Windows, Mac OS X	9+	basato sul linguaggio di programmazione visuale	basato sul linguaggio di programmazione visuale	basato sul linguaggio di programmazione visuale
May's Journey	Windows	12-18	Custom Programming Language (inspired from an Object-Oriented Language like Java)	Custom Programming Language (inspired from an Object-Oriented Language like Java)	Custom Programming Language (inspired from an Object-Oriented Language like Java)
No Bug's Snack Bar	Web-based	18+	basato sul linguaggio di programmazione visuale	basato sul linguaggio di programmazione visuale	basato sul linguaggio di programmazione visuale
Robot ON!	Windows	18+	C++	Sintassi e semantica, tipi di dati variabili e primitivi, espressioni e assegnazioni, condizionali e iterativi, comprensione del programma	Gioco di ricerca (*gratis*)
Educatio	Windows	18+	Gioco per	Algoritmi (descrizione testuale di	Gioco di

nal 'Pacman' Game			l'apprendimento Ricerca algoritmo	base degli algoritmi e diagramma di flusso grafico corrispondente insieme allo pseudocodice)	ricerca
CMX	Windows	18+	C	Teoria sulle matrici, outputs di variabili dopo l'esecuzione del programma di una matrice, sintassi dei programmi di varietà, immissione di valori di variabili, calcolo della somma della matrice, calcolo del valore massimo della matrice	Gioco di ricerca MMORPG

5. LA PIATTAFORMA DI GIOCO CODING4GIRLS PER GLI INSEGNANTI E STUDENTI

5.1. La piattaforma di Coding4Girls e l'approccio design thinking

Una perfetta combinazione di TIC e *serious game* mostra che il risultato del progetto di *O2 - Promuovere lo sviluppo delle abilità di programmazione tra le ragazze attraverso i serious games* mira a sviluppare abilità di programmazione tra i giovani, principalmente le ragazze, nell'istruzione di base e secondaria utilizzando un software sviluppato nel quadro del progetto.

Il software fu programmato per superare il divario esistente tra la partecipazione maschile e femminile alla formazione ed alle carriere informatiche introducendo interventi di apprendimento metodologico tali da rendere l'informatica attraente per tutti. Questo software e le attività di apprendimento da svolgere sono stati sviluppati con l'obiettivo di incentivare il coinvolgimento delle ragazze nell'informatica. Sono stati progettati seguendo l'approccio del *design thinking*, inteso come un processo creativo che aiuta gli studenti a progettare soluzioni significative in collaborazione con i loro coetanei.

Questo processo di progettazione è molto efficace grazie alla sua struttura volta ad identificare le sfide, raccogliere informazioni, generare potenziali soluzioni, perfezionare idee e testare soluzioni.

Il processo è ricorsivo per natura e richiede l'interazione. Ogni fase del processo richiede la rivisitazione e il richiamo durante un'esperienza di apprendimento per incoraggiare la sperimentazione, la fattibilità della soluzione e la riflessione (Luka, 2014)

Le attività di *design thinking* consentono un'esperienza altamente collaborativa dentro e fuori la classe. Gli studenti sono direttamente coinvolti nella raccolta di informazioni, nella generazione di conoscenza, nella comunicazione e nella presentazione.

Nel progetto Coding4Girls l'approccio del pensiero progettuale sfrutta i principali vantaggi dell'apprendimento basato sul gioco per incoraggiare e motivare gli studenti nel loro processo di apprendimento. Utilizza alcuni importanti elementi di gioco (es. Punti e sfide) e scenari che rendono più divertente una determinata attività aumentando il grado di coinvolgimento, motivazione e risultati raggiunti.

In particolare, l'uso delle sfide permette agli studenti di vedere il quadro generale prima di progettare una soluzione dettagliata, li incoraggia e li sfida a pensare in modo imprenditoriale alle tecnologie digitali ed al modo in cui possono essere utilizzate per affrontare i problemi del mondo reale.

Il software, sviluppato nel quadro del progetto, consiste in due differenti parti interconnessi: la piattaforma degli insegnanti e l'ambiente di gioco degli studenti.

5.2 La Piattaforma degli Insegnanti

La piattaforma degli insegnnati è una piattaforma web in cui gli insegnanti possono progettare e preparare corsi specifici per sviluppare le capacità di programmazione dei propri studenti utilizzando Snap!.

All'interno della piattaforma, ogni insegnante accede sia all'area privata che pubblica. Nella prima, possono preparare i loro corsi in base alle esigenze dei propri studenti. Il secondo è un archivio di tutti i corsi creati da altri colleghi insegnanti. Lo scopo è condividere, usare o lasciarsi ispirare dale attività realizate dagli altri colleghi (Figure 3.2). Ovviamente ogni insegnante può personalizzare tutti i corsi pubblici in base alla specificità delle proprie classi.

Fig. 5.1 La piattaforma degli insegnanti ed alcuni corsi disponibili.

Questi corsi sono costituiti come uno spazio di raggruppamento per attività tematicamente correlate, tutte connesse a un problema generale. Il problema viene

presentato all'inizio del corso agli studenti che possono fare un *brainstorming* tutti insieme per elaborare in modo collaborativo soluzioni provvisorie. Questa è una fase molto importante del processo di apprendimento in cui gli studenti possono generare nuove idee ed essere creativamente immersi in un processo di risoluzione dei problemi.

La traccia del *brainstorming* può essere preparata dagli insegnanti nella piattaforma degli insegnanti utilizzando alcune domande chiave o alcune osservazioni e riflessioni da utilizzare durante la discussione degli studenti. Il software offre la possibilità di preparare post-it sulla lavagna virtuale utilizzando testo, immagini o video come nella Figura 5.2:

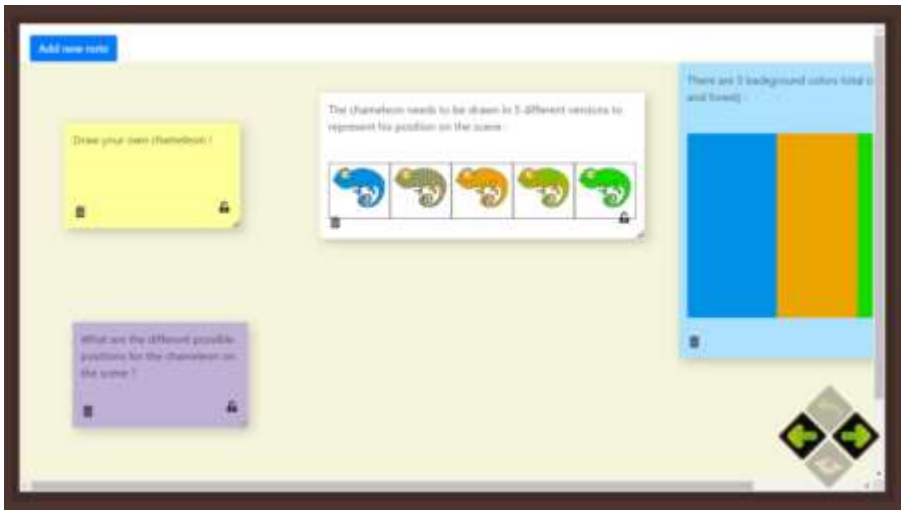


Fig. 5.2. Zona per il brainstorming nell'ambiente di gioco degli studenti
Area di contributi e discussione.

Tutti i post-it hanno l'icona di un lucchetto che può essere aperto quando il post-it va modificato oppure è bloccato quando il post-it non può essere modificato. Solo gli insegnanti hanno la possibilità di bloccare e sbloccare i post-it, gli studenti possono solo sapere se un post è bloccato o meno senza poter agire su di esso.

Dopo la fase di brainstorming, gli studenti vengono istruiti passo dopo passo rispetto alle attività specifiche (presentate in ordine consecutivo) volte a presentare gli strumenti necessari per risolvere il problema generale. Queste attività di codifica vengono presentate utilizzando Snap!, attraverso le attività semi-strutturate predisposte per sfidare gli studenti nella soluzione del problema e nella presentazione della soluzione finale.

Nella figura 5.3 la struttura di ogni corso C4G sono pubblicate nella piattaforma degli insegnanti come di seguito è mostrato.

Fig. 5.3 Struttura di un corso C4G.

Ogni attività raggruppata nel corso è chiamata sfida la quale viene presentata agli studenti in uno specifico ordine stabilito, in precedenza, dall'insegnante (Figura 5.4). Ad esempio, se un insegnante intende creare un corso sulla programmazione di base, la prima attività potrebbe riferirsi al concetto di *booleans*, alla struttura dei condizionali ed infine ai cicli di ripetizione. Ciò consentirà agli insegnanti di preparare fasi di apprendimento personalizzate per i loro studenti nella piattaforma degli insegnanti e gli studenti saranno guidati gradualmente all'obiettivo di apprendimento finale nell'ambiente di gioco degli studenti.

Ovviamente, gli studenti devono giocare e risolvere tutte le sfide nell'ordine stabilito dagli insegnanti prima di individuare la soluzione finale.

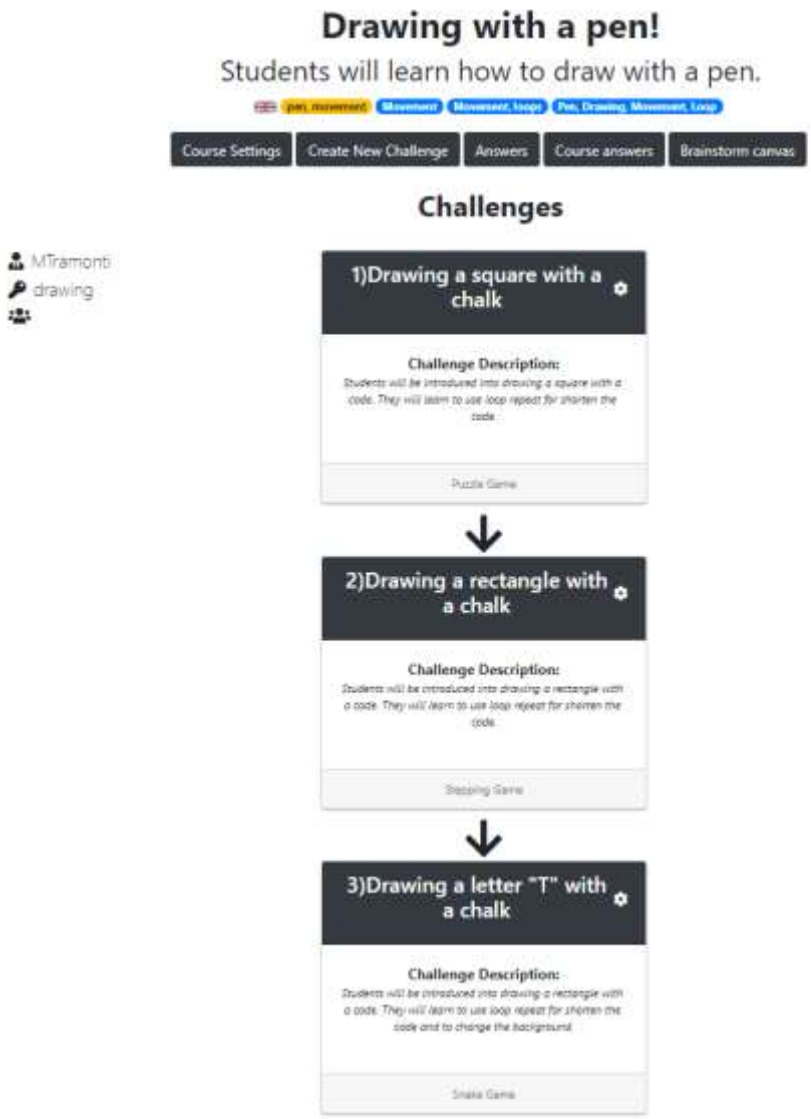


Fig. 5.4 Un esempio di sfide in un corso di coding nella piattaforma degli insegnanti.

Ogni sfida potrebbe essere associata a un mini-gioco sviluppato dal team del progetto. Lo scopo è aiutare gli studenti a comprendere meglio quale potrebbe essere il risultato finale di una specifica proprietà di codifica studiata.

Ad esempio, se una sfida riguarda lo studio della proprietà "*loop*", Match-3 è uno dei minigiochi sviluppati che potrebbero essere associati ad essa. Pertanto, ogni sfida potrebbe avere, oltre allo Snap! per risolvere il compito, un mini-gioco che lo studente dovrà giocare seguendo una pagina con le istruzioni definite dall'insegnante durante la loro preparazione nella piattaforma insegnanti (Figura 5.5).

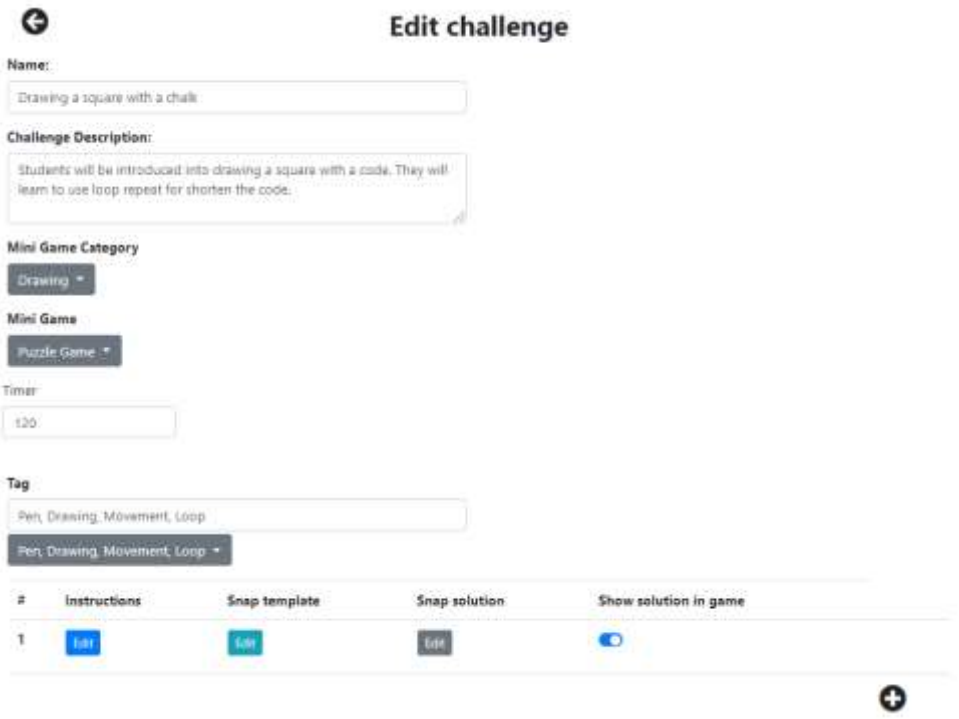


Fig. 5.5 Una sfida nella piattaforma degli insegnanti.

Tutti i giochi proposti (attualmente 11) agli studenti sono legati e semplificano i concetti di programmazione reali su cui sono stati progettati i corsi e gli scenari C4G. Tuttavia, non è obbligatorio includere un mini-gioco nella sfida: questa è solo una scelta dell'insegnante.

Ogni sfida è costituita da tre parti:

1. istruzioni - in cui l'insegnante può dare alcuni indizi o spiegazioni sul compito che deve essere svolto dai propri studenti;
2. modello di Snap - in cui gli studenti cercheranno di risolvere il compito assegnato utilizzando la codifica a blocchi dell'*open source* Snap! nell'ambiente di gioco degli studenti;
3. soluzione Snap – in cui gli insegnanti possono decider di mostrare o meno la soluzione finale del compito.

Se la sfida è complessa, in termini di conoscenza della programmazione, può essere strutturata in più compiti (Figura 5.6 In questo modo i docenti potranno guidare, passo dopo passo, i propri allievi, attraverso diverse fasi, al fine di risolvere un'attività complessa scomponendola in parti più piccole.

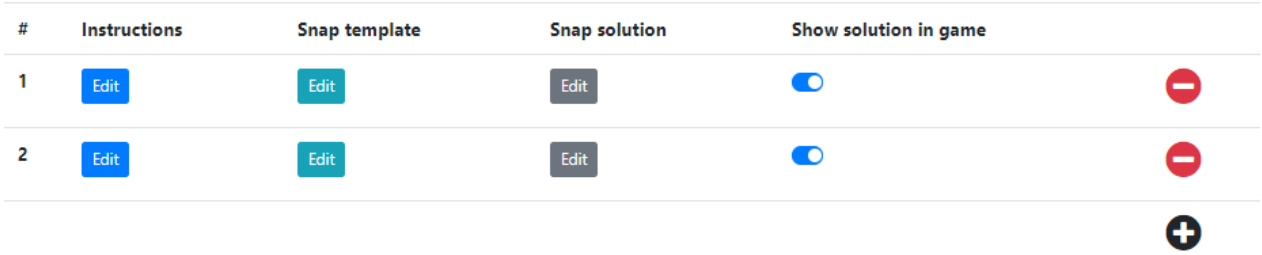


Fig. 5.6 Suddivisione in compiti più piccoli di una sfida complessa.

Inoltre, dopo la loro presentazione in ogni sfida, gli insegnanti possono raccogliere tutte le risposte Snap! dei propri studenti visualizzate in una tabella all'interno della piattaforma degli insegnanti. La tabella contiene l'elenco degli utenti nonché i dettagli di ogni sfida presente nel corso. Ogni riga riporta il nome utente, il cognome, il nome della sfida o delle sfide risolte.

Se non è presente una soluzione inviata, la colonna "Link soluzione" ("*Solution link*") sarà vuota altrimenti verrà evidenziata la riga e nell'ultima colonna comparirà la scritta "Soluzione" e, cliccandoci sopra, verrà mostrato l'invio dell'utente (Figura 5.7).

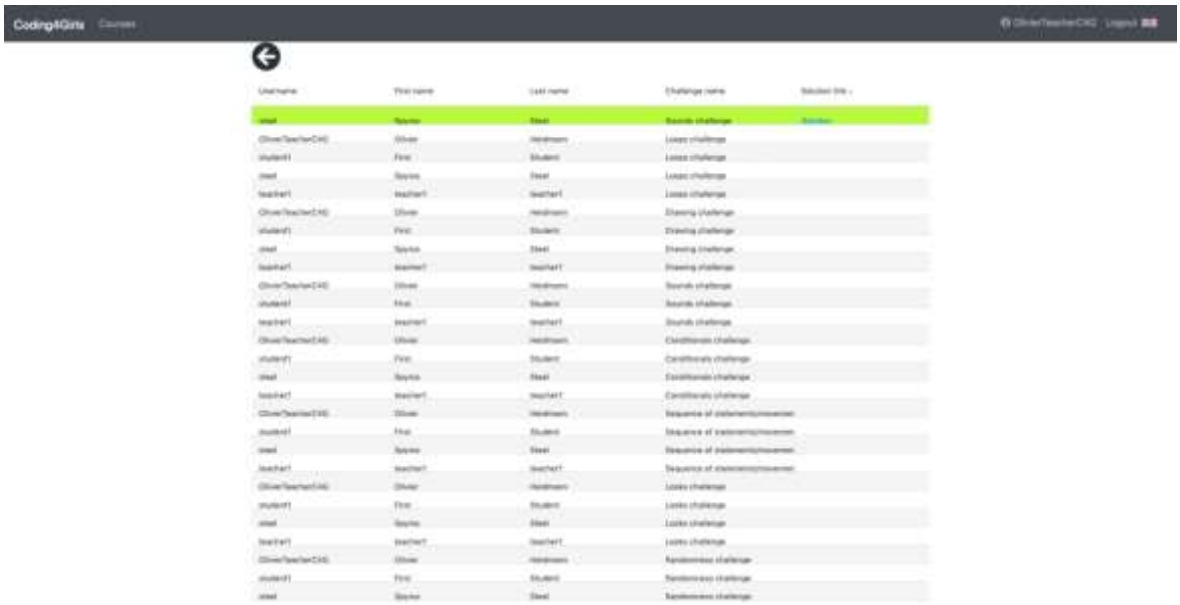


Fig. 5.7 Le risposte alla sfida da parte degli utenti con una soluzione

In questo modo, gli insegnanti possono sempre monitorare e tracciare il progresso ed i risultati dei loro studenti.

5.3. L'ambiente di Gioco degli Studenti

Gli studenti accedono a queste sfide attraverso l'ambiente di gioco per gli studenti ossia un software scaricabile per videogioco Unity 3D dove gli studenti possono scoprire e completare i corsi preparati dai loro insegnanti in modo divertente, coinvolgente e giocoso.

I corsi, utilizzando elementi dell'approccio del *design thinking*, presentano agli studenti un problema generale da risolvere e presentano gli strumenti per risolverlo con un approccio graduale. Come accennato in precedenza, sono preparati dagli insegnanti e pubblicati nella piattaforma degli insegnanti, ma gli studenti possono accedervi attraverso l'ambiente di gioco degli studenti.

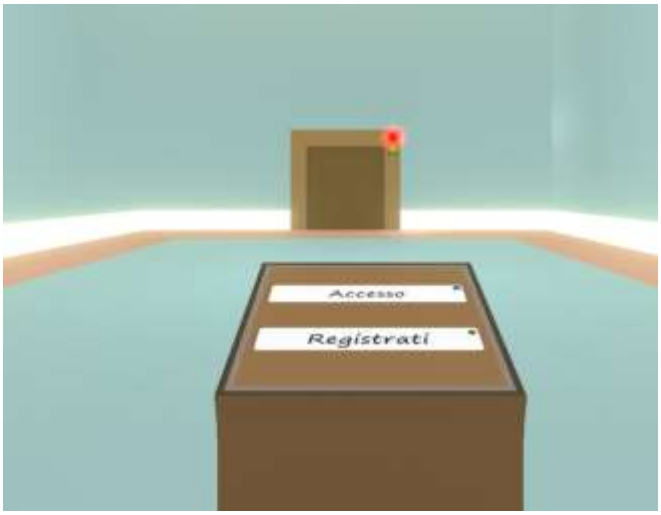


Fig. 5.8 La prima stanza di accesso negli ambienti di gioco degli studenti

Gli studenti possono partecipare a un corso inserendo il codice corretto nel terminale del portale. Il codice è unico e dovrebbe essere fornito dagli insegnanti che hanno già preparato il corso sulla propria piattaforma. Una volta inviato il corso, il suo nome è visibile nella parte superiore del portale, che indica anche il corso attualmente selezionato (Figura 5.9).

Fig. 5.9 - La seconda stanza con due terminali: uno per iscriversi ad un corso per la prima volta ed uno per selezionare un corso tra quelli già registrati

Gli studenti sono incoraggiati a progettare e programmare giochi che affrontino esigenze o problemi specifici (a seconda della scelta degli insegnanti). Si tratta di un approccio a livelli che consente agli alunni di iniziare con problemi facili fino a impegnarsi in compiti più impegnativi. Gli insegnanti progettano e presentano ai loro studenti scenari "semi-strutturati" in cui una soluzione è parzialmente pronta sfidandoli a completare l'attività. Pertanto, gli insegnanti possono preparare moduli piccoli e gestibili utilizzando il software Coding4Girls (C4G) adatto alle esigenze e alle conoscenze dei loro studenti.

Seguendo l'approccio del pensiero progettuale, il team di progetto ha preparato alcuni corsi e scenari di apprendimento progettati per sfidare gli studenti a risolvere uno specifico problema di codifica. Attualmente, i 21 scenari di apprendimento, raccolti nel rapporto "Collezioni di schede di apprendimento basate sulla progettazione di giochi rivolti agli insegnanti", sono stati riadattati all'approccio del pensiero progettuale seguendo la struttura del software C4G.

Gli scenari hanno diversi livelli da quello base a quell avanzato e sono disponibili in inglese, sloveno, italiano, croato, bulgaro e greco.

I compiti possono essere risolti individualmente o in modo collaborativo suscitando discussioni del gruppo in classe oppure virtualmente nell'ambiente di gioco degli studenti. Il software offre uno spazio di brainstorming in cui gli studenti possono discutere e condividere le loro idee inserendo e gestendo post-it multimediali come mostrato nella

Figura 5.11. Tutti gli studenti coinvolti nel corso possono scrivere il proprio contributo visibile a tutti gli iscritti al corso specifico.

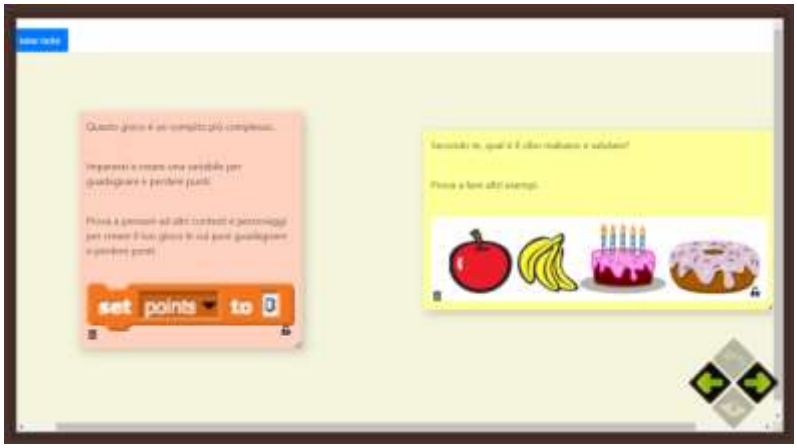


Fig. 5.11: Area per il brainstorming nello scenario di apprendimento Acquista cibo per un picnic nell'ambiente di gioco degli studenti.

La figura seguente mostra il pannello delle sfide nell'ambiente di gioco degli studenti del corso creato dal docente nella propria piattaforma. I numeri da 0 a 13 rappresentano le sfide. In questo pannello, la sfida numero 0 rappresenta l'istruzione generale fornita ad inizio gioco ed all'apertura conduce gli studenti alle istruzioni sul problema globale, sul relativo modello Snap!, sulla traccia di brainstorming. Le altre sfide, dalla numero 1 alla 13, rappresentano i compiti preparati dagli insegnanti nella loro piattaforma.

Fig. 5.12. Il pannello delle sfide nell'ambiente di gioco degli studenti

Selezionandone uno, i dettagli della sfida appariranno nella parte inferiore dello schermo, con il nome della sfida a sinistra, il minigioco associato alla sfida al centro e un pulsante per avviare la sfida. La figura 3.12 mostra il nome della sfida correspondente all'argomento da studiare, in questo caso "*Conditional challenge*". Oltre all'argomento della sfida, il sistema mostra il minigioco 3D ad esso associato, ad es. "*Find your path*".

Gli insegnanti possono decidere a quale mini-gioco (opzionale) giocheranno i loro studenti selezionando uno dei mini-giochi esistenti, come Match3 game, Dice game, Inventory game, Find your path game, Stepping game, Sound game, Snake game, puzzle, gioco di corrispondenza dei modelli e quiz con domande a scelta multipla.

Fig. 5.13 Un esempio di un mini-game disponibile – Il gioco del dado

Ogni sfida nell'ambiente di gioco degli studenti è strutturato come segue: un mini-gioco introduttivo che illustra il concetto di codifica; una pagina di istruzioni per l'attività da svolgere in Snap; due schemi Snap! di cui uno da utilizzare per risolvere l'attività e l'altro per visualizzare la soluzione dell'attività.

La pagina di istruzioni è in HTML (Figure 5.14), solitamente arricchita da immagini o video, presenta il contesto e l'obiettivo specifico del compito di apprendimento da svolgere in Snap!.

Fig. 5.14 Un esempio della pagina didattica della sfida nell'ambiente di gioco degli studenti.

Questa pagina sarà seguita da una scheda Snap! (Figura 14), basata su un modello fornito dal docente contenente l'attività o il problema di codifica che deve essere eseguito o risolto dagli studenti.

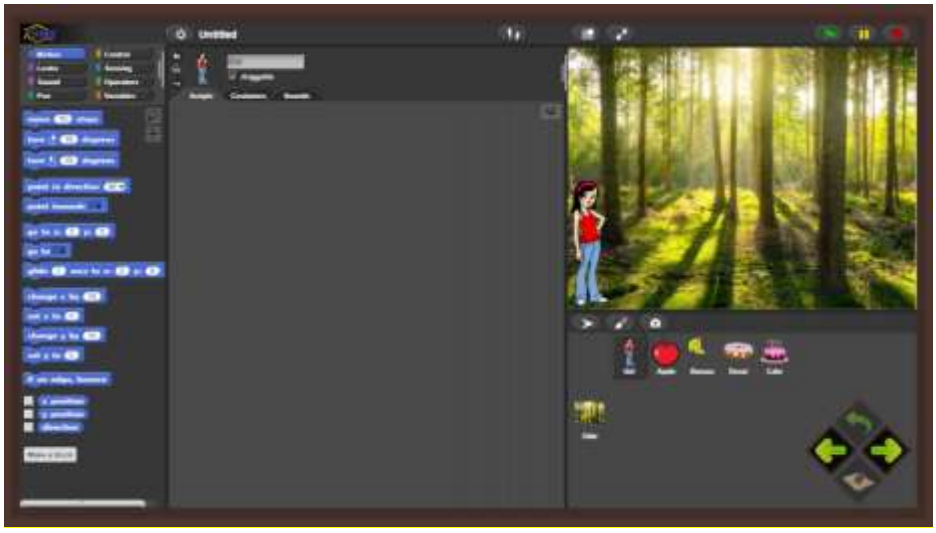


Fig. 5.15: Un esempio del modello di Snap!' fornito dagli insegnanti per risolvere il compito di coding task nell'ambiente di gioco degli student.

Un'altra scheda Snap! Può essere fornita per visualizzare la soluzione della sfida proposta. Tuttavia, solo l'insegnante può decidere se mostrare o meno la soluzione. Dipenderà dalla gestione del processo di insegnamento selezionata dall'istruttore.

Fig. 5.16 Un esempio di scheda Snap! Che fornisce la soluzione della sfida predisposta dall'insegnante per i propri studenti nell'ambiente di gioco

Poiché ogni corso raccoglie più sfide, questi passaggi verranno ripetuti tante volte quante sono le sfide totali identificate dagli insegnanti. Ciò consente di suddividere un'attività di *coding* complessa in semplici passaggi elementari in cui la risposta e/o la soluzione all'attività precedente diventa il modello con cui deve essere eseguita l'attività successiva.

Alla fine, il corso è costituito da un certo numero di sfide che presentano l'insegnamento della programmazione attraverso un processo di scaffolding incrementale.

Una volta che gli studenti hanno completato tutte le sfide del corso, vengono ricondotti al problema di *coding* iniziale e verrà chiesto di sintetizzare le nuove conoscenze che hanno appena acquisito. Alla fine del corso, i giocatori possono vedere anche tutte le soluzioni al problema proposte dagli altri studenti suscitando momenti di confronto.

Questo approccio e strumenti C4G consentono agli studenti di sviluppare, migliorare e rafforzare le capacità di codifica attraverso attività di formazione personalizzate che combinano attività divertenti e di apprendimento.

6. ESEMPI DI LEZIONI (SCHEDE DI APPRENDIMENTO)

6.1. Gli scenari di apprendimento;

Nell'ambito del progetto Coding4Girls sono stati sviluppati 22 schede di scenari di apprendimento. Descrivono attività di apprendimento misto *end-to-end* che implementano il *serious game* CODING4GIRLS e gli approcci di pensiero progettuale. Le schede di apprendimento sono disponibili in inglese e nelle lingue nazionali dei partner del progetto: bulgaro, croato, greco, italiano, portoghese, sloveno e turco. Sono tutti disponibili sul sito web del progetto: (https://www.coding4girls.eu/results_03.php)

Le schede di apprendimento preparate presentano in modo conciso informazioni che aiutano gli istruttori a integrare i *serious games* proposti e le metodologie di apprendimento del pensiero progettuale nelle loro pratiche di insegnamento. Seguono il progetto di apprendimento basato sul gioco di CODING4GIRLS e includono informazioni per ogni attività di apprendimento da realizzare al fine di sviluppare abilità di programmazione per ragazze e ragazzi. Sono disponibili le seguenti informazioni:

- obiettivo educativo generale della corrispondente attività di apprendimento;
- concetti coperti dall'attività di apprendimento;
- obiettivi di apprendimento specifici;
- risultati di apprendimento attesi;
- uso graduale dell'approccio di apprendimento basato sulla progettazione del gioco CODING4GIRLS;
- metodi di verifica per valutare la conoscenza sviluppata;
- domande per avviare la discussione tra gli studenti nel contesto della collaborazione in classe.

Gli insegnanti possono utilizzare gli scenari e i giochi nella sequenza proposta o selezionarli liberamente in base alle proprie preferenze e necessità. L'insegnante ha l'opportunità di adattare alcuni scenari alla conoscenza degli studenti rispetto ad alcuni concetti appresi. Ad esempio nell'ambito della matematica gli studenti devono conoscere il sistema di coordinate, il concetto di numeri negativi, delle frazioni prima di presentare loro lo scenario di apprendimento in cui si chiede di applicare tali concetti.

Le schede di apprendimento illustrano sia la funzionalità generica del gioco proposto, inclusi i processi di interazione dell'utente e la generazione di feedback, sia le descrizioni di tutte le attività di apprendimento che saranno implementate. Sono strutturate su concetti di base della programmazione a quelle con concetti di programmazione più avanzati. La tabella seguente rappresenta l'ordine proposto delle attività.

Table 6.1. Elenco delle schede di apprendimento sviluppate durante il progetto Coding4Girls

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**	UL

	Usare i cicli, condizionali, variabili	
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

Tutte le 22 schede di apprendimento sono presentate nell'Appendice 1. Nell'Appendice 2 sono forniti i codici per tutti i giochi presentati nella sezione pubblica dei giochi nell'ambiente Coding4Girls.

6.2. Esempi con utilizzo dell'ambiente di gioco sviluppato nell'ambito del progetto Coding4Girls

L'ambiente di gioco degli studenti è un videogioco basato su Unity disponibile su Windows e Mac OS X. Durante il gioco, l'utente scopre prima una scena introduttiva che presenta le dichiarazioni di non responsabilità europee richieste con il progetto e il logo Erasmus + : informazioni visualizzate per 4 secondi.

Il giocatore arriva in una grande stanza vuota chiamata *lobby*, controlla e visualizza l'ambiente attraverso una prospettiva in prima persona. Una console è posizionata al centro della stanza, consentendo all'utente di accedere al server C4G utilizzando le proprie credenziali o creando un nuovo account. Una volta effettuato l'accesso, una porta si apre e il giocatore può procedere in un'altra stanza, dove può iscriversi a un nuovo corso o accedere a un corso già attivato. Una volta selezionato il corso desiderato, l'utente attraverserà un portale mobile per procedere con il gioco e scoprire il contenuto esatto del corso selezionato.

Il corso è formato da una raccolta di sfide di codifica collegate utilizzando la piattaforma Snap!, ciascuna illustrata da un minigioco. Idealmente, ogni sfida illustra un aspetto o una fase della lezione da apprendere durante l'attività.

Per impostazione predefinita, C4G ha indicato alcuni concetti di codifica di base, ciascuno illustrato da un mini-gioco: cicli, condizionali, variabili, istruzioni, parallelismi, operatori, eventi. Inoltre, gli insegnanti possono anche decidere di sostituire il mini-gioco con un quiz con domande a scelta multipla oppure di lasciare solo gli esercizi definiti nelle sfide.

Fig. 6.1 La categoria mini-giochi disponibile nella piattaforma degli insegnanti

Ogni sfida include attività di codifica da risolvere nelle schede Snap! e può essere associata al mini-gioco visualizzato nel pannello della sfida (Figura 6.2).

Fig. 6.2 – Il pannello di sfida nell'ambiente di gioco degli studenti

L'integrazione del mini-game nella sfida specifica consiste nell'illustrare il concetto di programmazione alla base del compito pianificato dagli insegnanti. Non è obbligatorio. Ciò significa che l'insegnante può decidere di avere o meno un mini-gioco per i propri studenti nella sfida e quale mini-gioco utilizzare poiché lo seleziona da un elenco di mini-giochi esistenti.

La necessità di associare un mini-gioco alla sfida è un'opportunità che va considerata se porta un valore aggiunto in termini di comprensione del concetto, visualizzazione della meccanica e, naturalmente, coinvolgimento e motivazione degli studenti nel loro percorso di apprendimento.

Per il momento esistono 11 differenti mini-games che vanno da un gioco Match3 a un quiz con domande a scelta multipla.

Fig. 6.3 I mini-giochi disponibili nella piattaforma degli insegnanti

Quando l'utente entra in una sfida collegata a un mini-gioco, verrà teletrasportato nel luogo in cui si svolge il mini-gioco. Ci sono molti luoghi per i diversi mini-giochi che variano dalle spiagge sabbiose alle montagne innevate.

Fig. 6.4 Un esempio di luogo associate con un mini-game.

Tra i minigiochi disponibili, ci sono Snake game, Match3 game, Dice game, Inventory game, Find your path game, Stepping game, Sound game, Puzzle game, Pattern matching game e un quiz con domande a scelta multipla.

Ogni mini-gioco presenta le sue difficoltà e la gamma di mini-giochi disponibili risponde alle diverse possibili fasce d'età degli studenti e / o agli argomenti di programmazione insegnati nei corsi.

Uno dei minigiochi più semplici è il gioco del serpente, ambientato sulla riva di un fiume e dove il giocatore deve seguire il segno della freccia in legno fino a un cerchio rosso per iniziare l'attività. (Figura 6.5.)

Fig. 6.5 Un esempio di luogo associato al gioco del serpente

Il gioco del serpente (Figura 6.6) si svolge nell'acqua. Controlli il serpente usando le 4 frecce della tastiera e cerchi di sopravvivere il più a lungo possibile. Il serpente aumenta di dimensioni mangiando i punti verde brillante, ma può morire se questi circolano sulla coda, colpiscono il bordo dello schermo o mangiano un punto rosso. Ogni punto verde mangiato aumenta il punteggio di una unità.

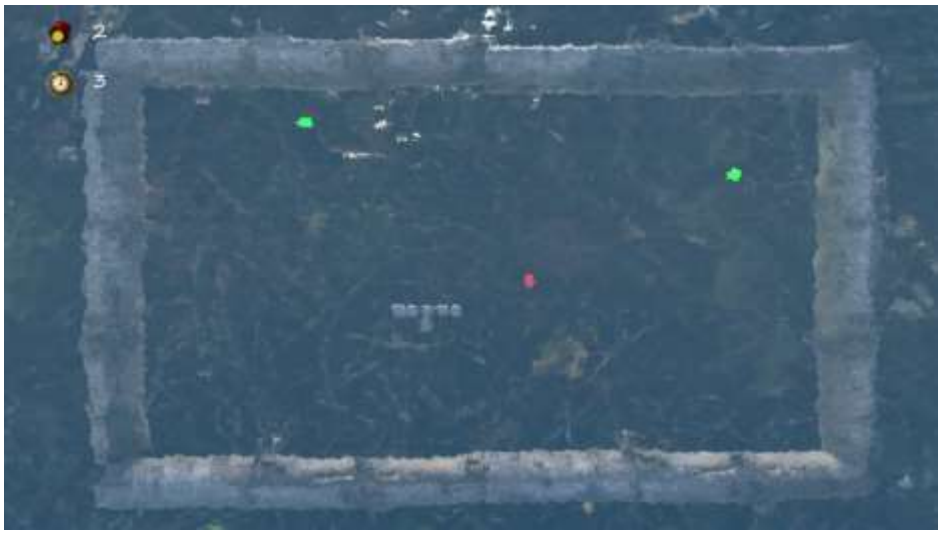


Fig. 6.6 Il gioco del serpente

Questo semplice minigioco offre una buona illustrazione di diversi concetti di programmazione come loop, movimenti, modifica della grafica e così via.

Un altro esempio è il gioco Match3. Si svolge in una regione innevata ed elevata dove i giocatori possono muoversi liberamente.

Fig. 6.7 Il luogo associato al gioco Match3.

Il mini-game Match3 è basato su un'attività tipica dove devi trascinare e rilasciare quadrati adiacenti per farli sparire se formano un gruppo di tre o più tessere dello stesso tipo.

Fig. 6.8 Match3 mini-game

Il gioco Dice si svolge in boschi oscuri. La parte interattiva del mini-gioco è all'interno di una capanna abbandonata, sulla sedia di fronte alla porta. È una tavola di legno con 2 dadi sopra.

Sia il giocatore che l'Intelligenza Artificiale dovrebbero tirare i dadi a turno e quello con la somma più alta dei due dadi vince il round.

Fig. 6.9 Gioco del dado

Il gioco dell'inventario si svolge nei campi, vicino a un piccolo bosco. Ci sono sfere colorate che galleggiano intorno all'area e il giocatore deve raccoglierle, secondo le istruzioni fornite dall'insegnante sul cartello in legno.

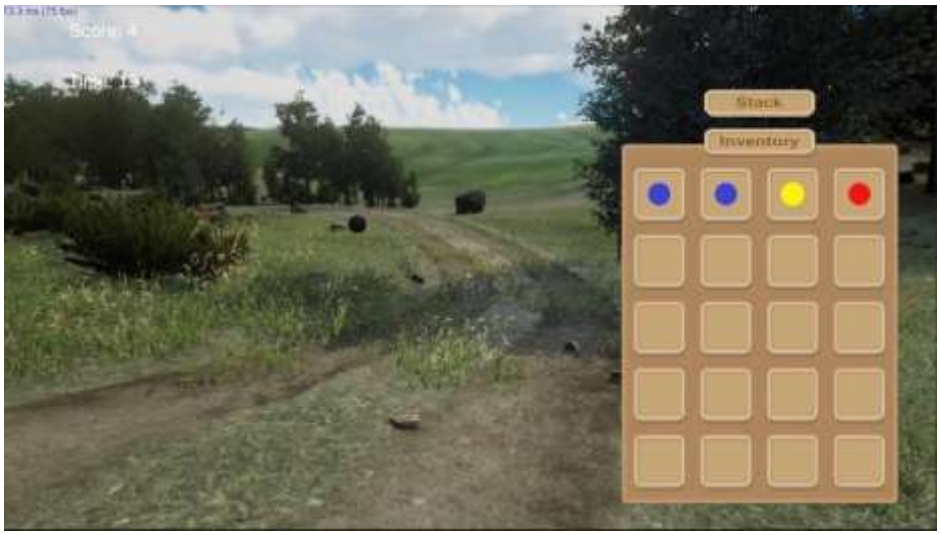


Fig. 6.10 Il gioco dell'inventario

Il gioco di passi Stepping mini-game si svolge su una spiaggia di un'isola al tramonto. Il giocatore deve rispondere alla domanda scritta sul grande scoglio al centro della spiaggia calpestando la lettera scrivendo la risposta corretta.

Quando il giocatore calpesta una delle pietre con una lettera che forma la risposta, la lettera sarà evidenziata in verde e apparirà sulla seconda grande roccia accanto a quella in cui è scritta la domanda. Se l'utente calpesta una pietra sbagliata, questa verrà evidenziata in rosso e dovrà ricominciare l'ortografia della risposta.

Fig. 6.11 Stepping game

Il quiz a scelta multipla è ambientato in un campo sul lato di una scogliera e i giocatori sono liberi di vagare all'inizio e per iniziare il mini-gioco.

Le domande vengono visualizzate in cima alla scogliera e ogni campo conterrà le quattro possibili risposte, insieme alle immagini, se presenti.

L'icona della medaglia rappresenta il punteggio attuale dello studente, in base alle risposte corrette e l'orologio conta alla rovescia i secondi per terminare questa sfida.

Il pulsante freccia in basso consente di passare alla domanda successiva.

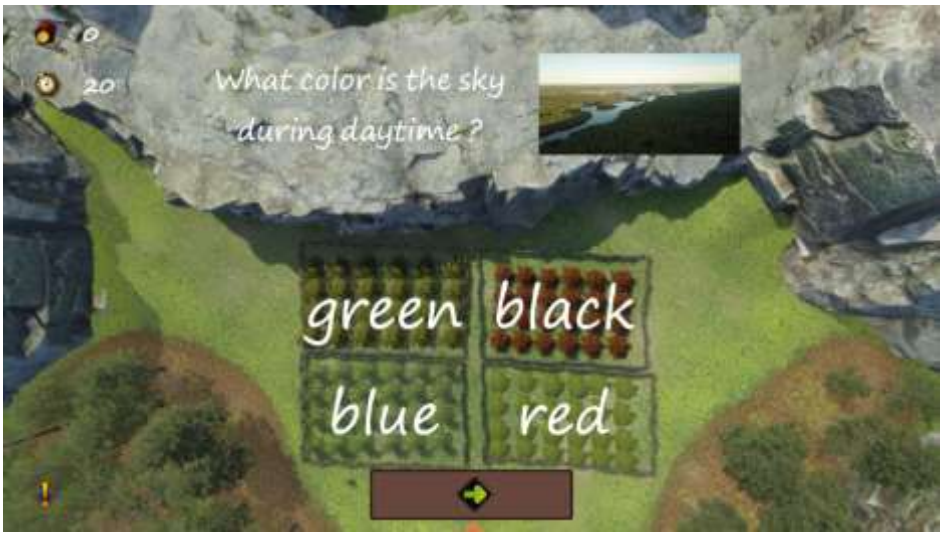


Fig. 6.12 Quiz a risposta multipla

Il gioco del puzzle si svolge in una zona rocciosa della mappa. Lo scopo è trovare i pannelli sparsi nell'area e risolvere gli enigmi creando una linea che parte dal cerchio bianco del pannello fino al quadrato rosso.

Fig. 6.13 Un pannello nel gioco del puzzle

Il gioco della corrispondenza del modello si svolge sulle rive di un fiume. Ci sono alcune caselle con i numeri scritti sul lato che iniziano a galleggiare sul fiume verso il mare. L'obiettivo del gioco è che il giocatore li raccolga per completare una determinata operazione di matematica.

In altri termini, i giocatori devono utilizzare i numeri flottanti e le operazioni disponibili per raggiungere il numero definito scritto su un tavolo blu.

I numeri di destinazione e i numeri mobili vengono generati in modo casuale ma in un modo che garantisce che ci sia sempre una possibile combinazione per raggiungere il numero di destinazione.

Le operazioni disponibili dipendono dall'insegnante e possono essere una delle seguenti:

- operazioni base (+, -, *, /);
- operazioni avanzate (potere, radice quadrata, modulo);
- trigonometria (Cos, Sin, Tan);
- Bbooleans (AND, OR, XOR).

Fig.6.14 Gioco di corrispondenza dei modelli

Un altro esempio di mini-giochi è quello del suono “Sound” che richiede molta attenzione da parte dei giocatori al fine di eseguire i propri compiti. Si svolge in una foresta tropicale e il giocatore deve trovare i cinque pannelli sparsi e risolvere per ciascuno di essi un puzzle basato sui suoni che possono sentire nella giungla.

Fig. 6.15 Uno dei pannelli nel gioco dei suoni

Ogni Colonna del pannello rappresenta un suono e ogni riga il tono del suono. I pulsanti in alto sono per i toni alti e quelli in basso per i bassi. Quando l'utente è vicino a un pannello, verrà riprodotta una combinazione di 3 possibili suoni (un richiamo di uccello a tono basso, un richiamo di uccello a tono medio e un richiamo di uccello a tono alto). La

combinazione dei richiami degli uccelli viene ripetuta in un ciclo dove ogni ripetizione è separata dalle altre da un silenzio di 3 secondi.

Il giocatore dovrà prestare molta attenzione all'ordine in cui vengono riprodotti i richiami degli uccelli e riprodurlo di conseguenza sul pannello.

Ad esempio, sul primo pannello (Figure 6.16) l'utente può sentire prima una chiamata con tono basso e poi uno con tono alto. La risposta corretta è quindi premere il pulsante basso a sinistra del pannello e il pulsante alto a destra.

Fig. 6.16 Risposta degli studenti visualizzata sul pannello nel gioco “Sound”.

Per selezionare un determinato pulsante, l'utente deve semplicemente puntare il cursore del mouse su di esso e il pulsante verrà evidenziato in giallo. Se il pulsante premuto è quello corretto, apparirà verde, altrimenti apparirà nero.

Il suono è spesso importante per qualsiasi gioco, ma è fondamentale in questo infatti è al centro del ciclo del gioco. I giocatori devono riconoscere i toni alti, medi e bassi e, quindi, devono concentrarsi su di essi.

L’ambiente di gioco degli studenti può essere utilizzato da tutti gli studenti, principalmente dai 10 ai 15 anni, e mira a incoraggiare un'equa partecipazione alle attività di programmazione all'interno e all'esterno della classe. Mostra i concetti di programmazione attraverso un ambiente virtuale 3D che gli studenti possono esplorare, mini-giochi che dimostrano i concetti, un ambiente collaborativo per condividere idee tra i membri della classe verso una potenziale soluzione, revisioni delle soluzioni di altri per costruire ulteriori

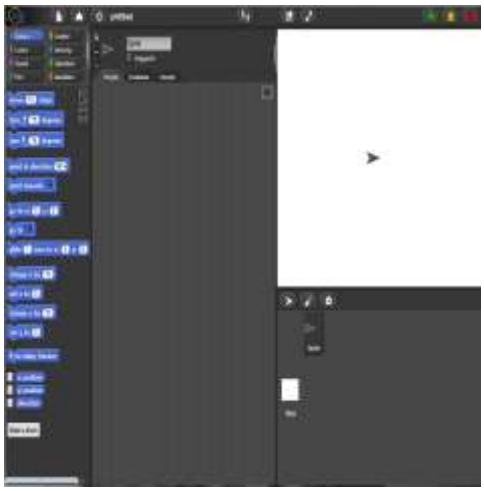
prospettive su come affrontare un problema e confronto delle soluzioni rispetto a quella introdotta dall'insegnante.

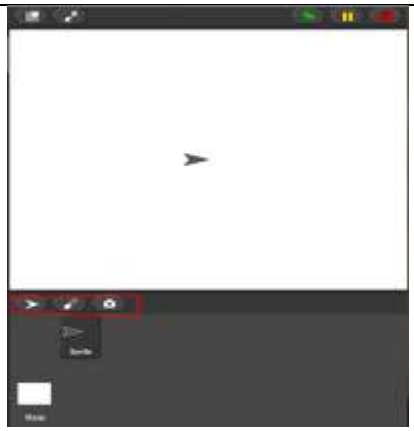
Si possono prevedere sviluppi e miglioramenti futuri per arricchire il software C4G con molti altri minigiochi per aiutare ad illustrare ulteriori concetti di programmazione. La natura modulare del *serious game* C4G e il modo in cui è stato progettato in Unity consente questo tipo di estensioni future.

APPENDICE 1. SCHEDE 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.

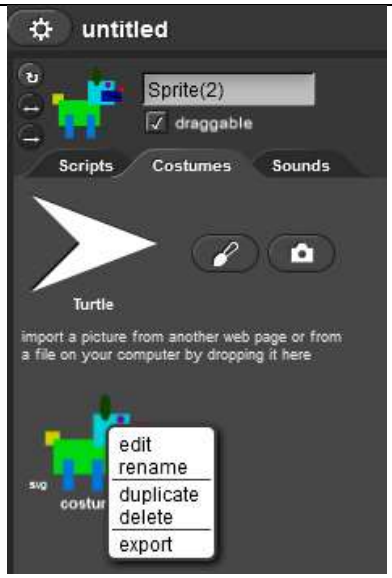
Dopo aver disegnato lo *sprite*, dovresti assicurarti che il centro di rotazione dello *sprite* sia dove vuoi che sia. Per farlo usa questo .

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

[Fase 3]

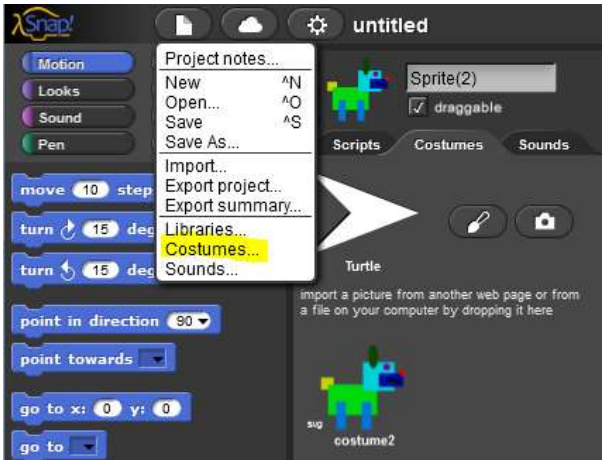
Per modificare il tuo *sprite*, scegli l'etichetta "Costumi", questo è visibile solo quando il tuo *sprite* è cliccato.

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 …

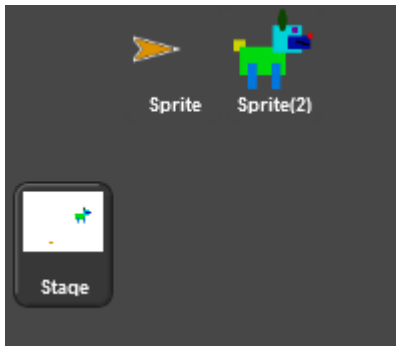
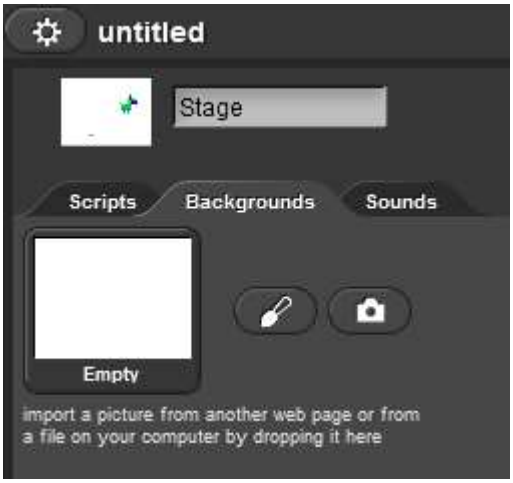


Questa opzione verrà mostrata solo quando si fa clic sul tuo *sprite* sotto lo *stage*.

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

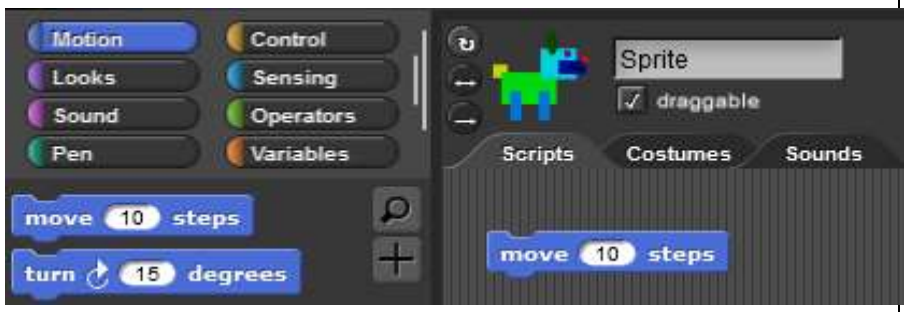
.

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.

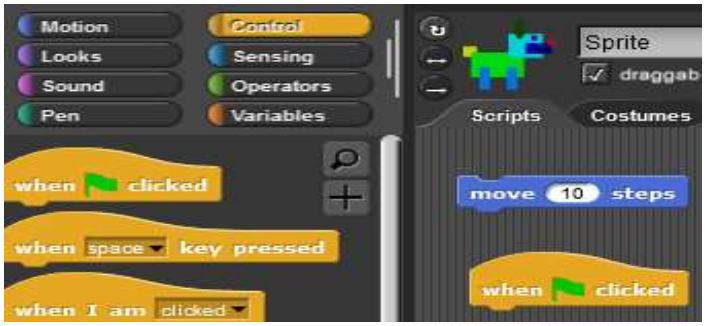
[Fase 3]

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]

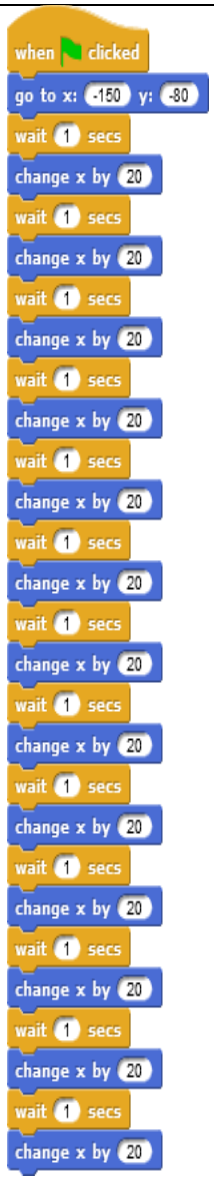
	Hai trovato entrambi i comandi dire nella categoria “Aspetto” (Looks). La principale differenza è che con non si dice al programma di “aspettare n___ secondi” prima che il codice continui o smetta di dirlo in qualsiasi momento. [Fase 7] Prendi il tuo personaggio precedentemente creato. Trascinalo sullo *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	Dimostrazione dell'insegnante Lavoro individuale

insegnamento	
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.

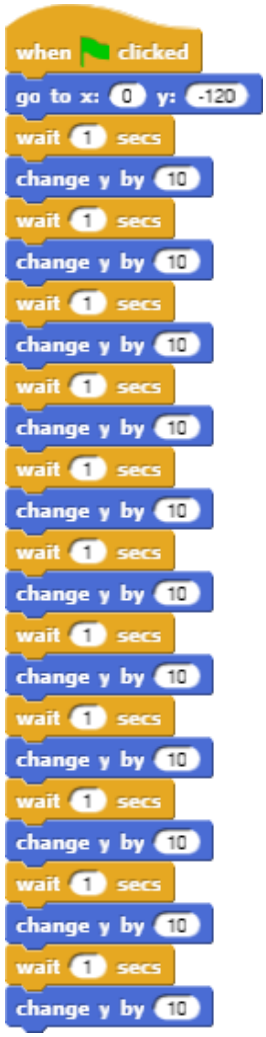
[Fase 2]

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.

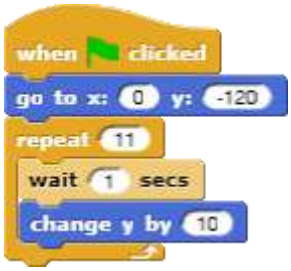
Una possibile soluzione del compito:



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:

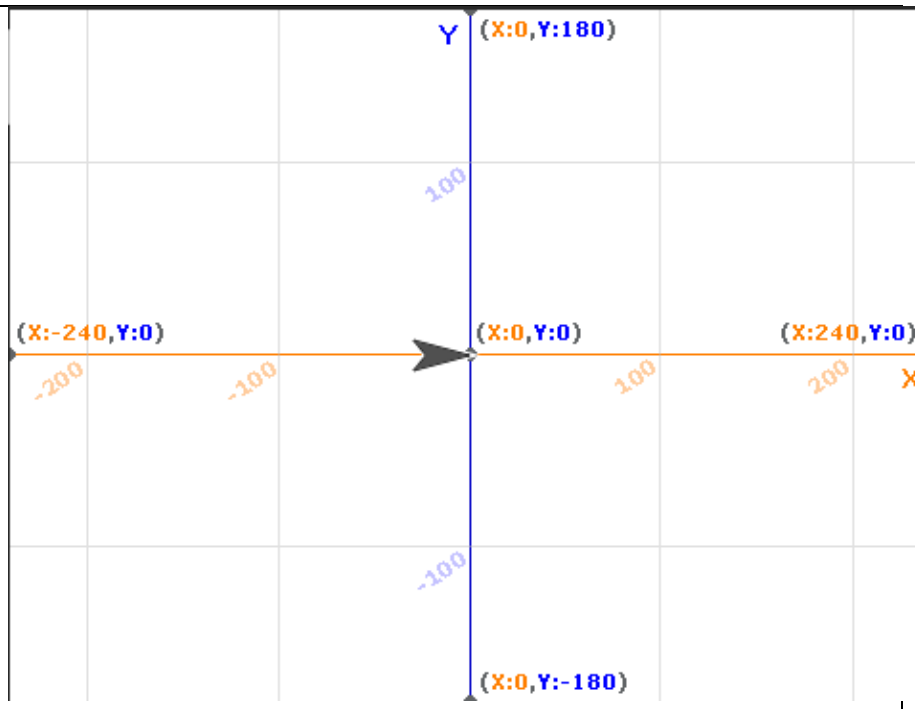


Compito: cerca di far correre il cane verso la palla e tornare indietro.

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

Cosa ti è piaciuto di più?

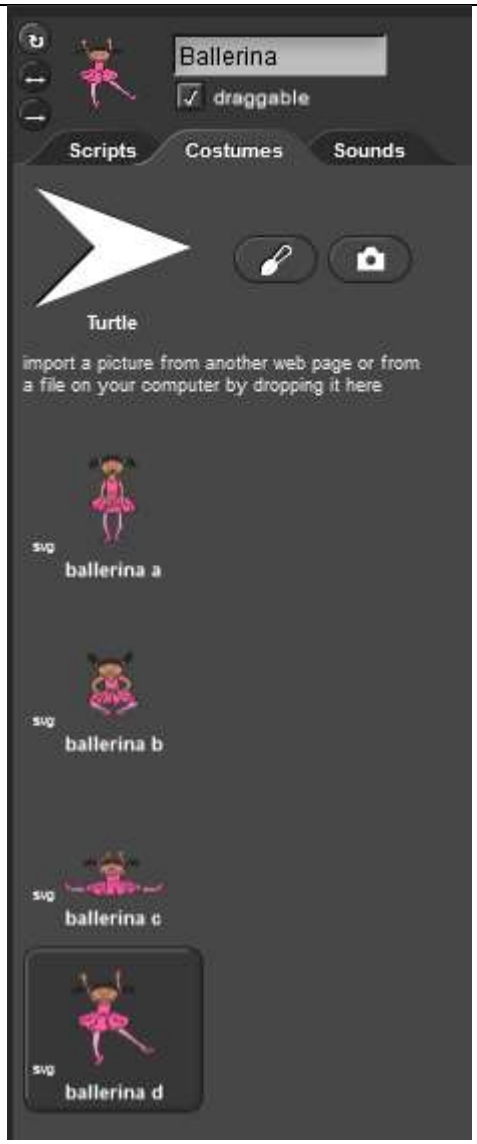
Puoi aiutarti con la posizione x e y dello *sprite* usando lo sfondo XY Grid in Snap:

	
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

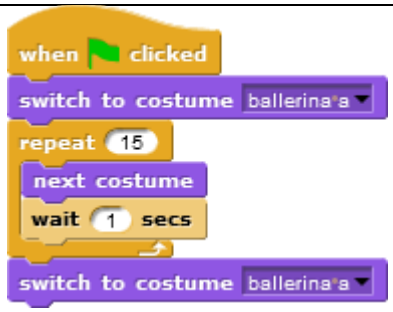
Riepilogo dell'insegnamento	(Motivazione-Introduzione, Implementazione, Riflessione e valutazione) Imparerai a realizzare un'animazione per uno sprite in modo che sembri camminare, ballare ... [Fase 1] Apri un nuovo progetto vuoto, fai clic sull'icona che sembra un pezzo di carta bianca e seleziona Costumi "*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.

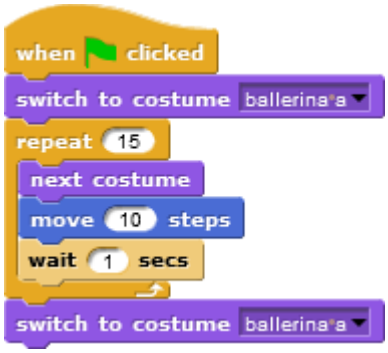
Soluzione:



[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]

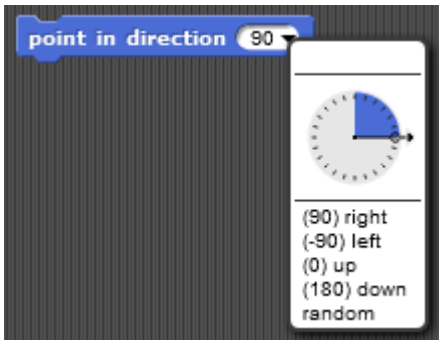
Apri un nuovo progetto vuoto e importa i costumi di “Avery che cammina”. Aggiungi uno sfondo adatto per far camminare Avery. Crea un’animazione per Avery che cammina dal lato sinistro a quello destro della scena. Prova a capire come animare Avery in modo tale che i suoi passi sembrino collegati come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

Una possibile soluzione:

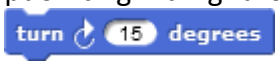
[Fase 4]

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.

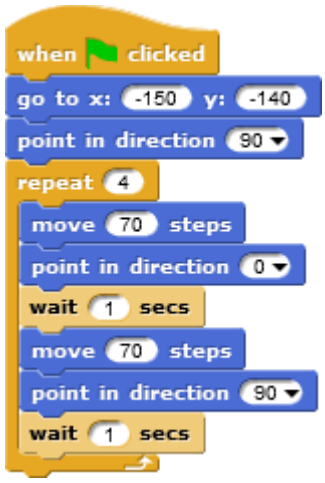
Apri

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,

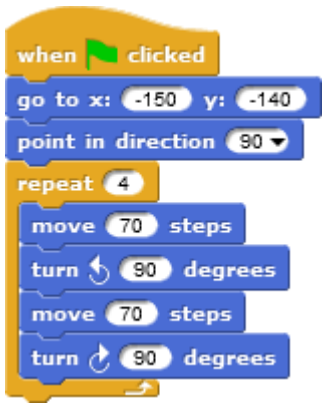
deve camminare solo sull'area verde. Posiziona il topo nella direzione in cui si sta dirigendo e spostalo di n __ passi. Per vedere come il topo si muove, usa "attendere 1 secondo" tra le due stringhe.

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

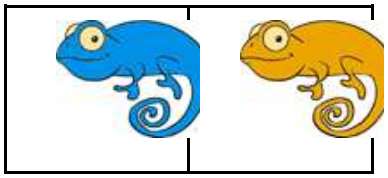
	da cambiare il suo aspetto "*looks*" (costume) e dire dove si trova rispetto a cinque diverse situazioni: 1) quando nuota nel mare, deve cambiare il suo colore in blu e dire "Sto nuotando nel mare", 2) quando si trova tra il mare e la spiaggia, la sua pelle diventa blue color sabbia e dice "Sono tra 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.

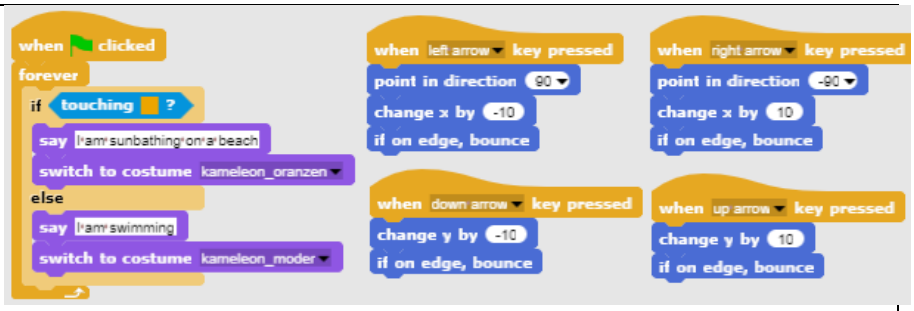
Successivamente discuteremo con gli studenti delle diverse posizioni del Camaleonte sulla scena e come possiamo realizzarle usando il blocco "*touching color* ".

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]



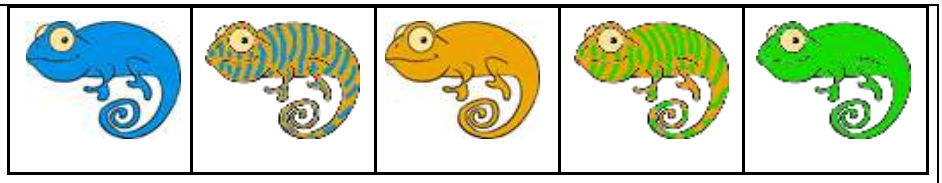
[Versione completa]

[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:

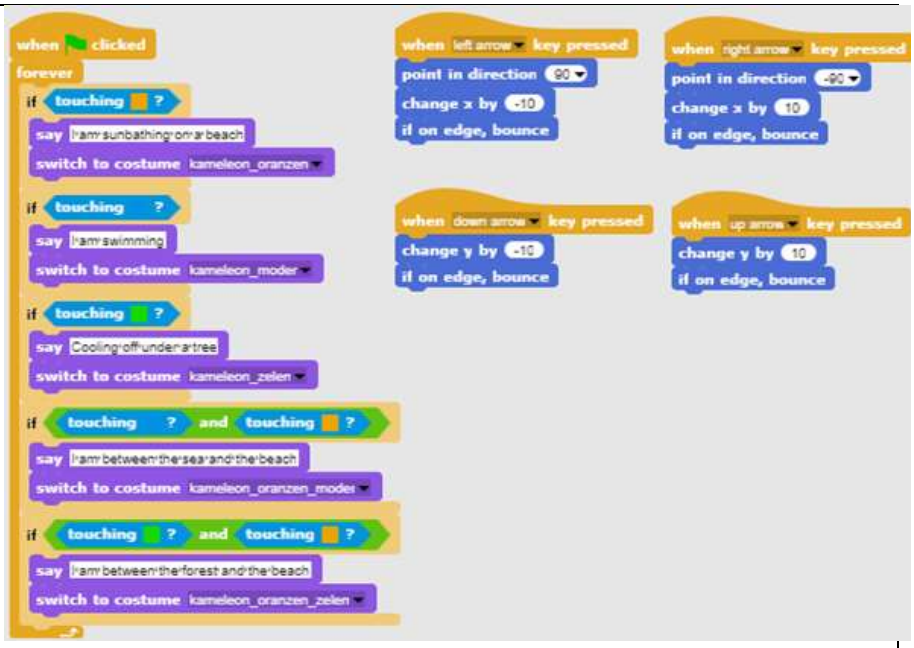
Successivamente formiamo un'espressione logica con l'uso dell'operatore logico AND, perché vogliamo verificare se il Camaleonte sta toccando due colori contemporaneamente:

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).

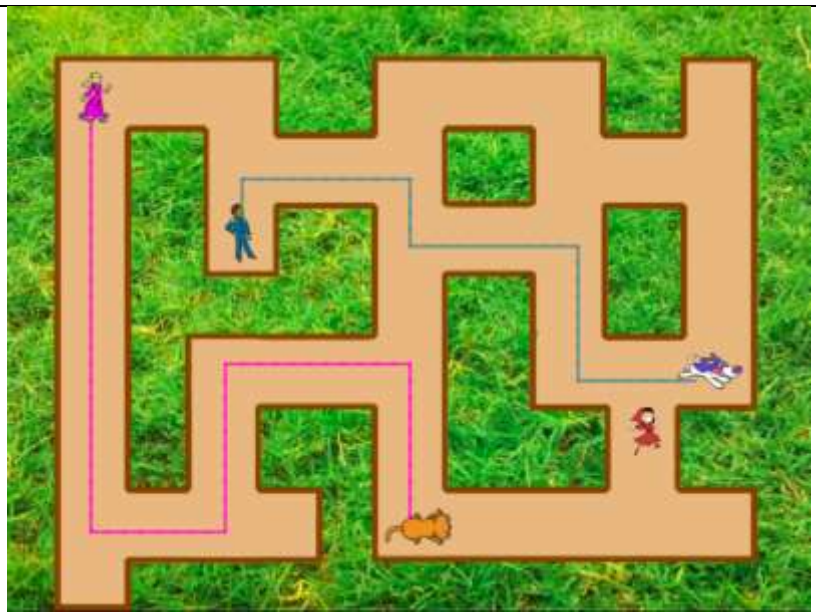
Per aiutarli a comprendere il concetto di ciclo continuo e come usarlo per rilevare il colore di sfondo, possiamo includere un codice per rilevare due situazioni: 1) l'oggetto è interamente su un colore, 2) l'oggetto tocca due colori contemporaneamente. Incarichiamoli di

	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.

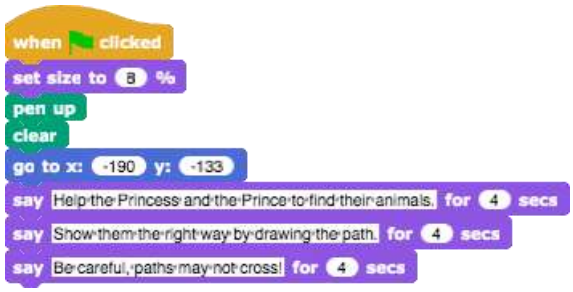
Allo stesso modo si imposta il colore della penna premendo il tasto:

[Fase 6]

Alla fine si programma quando cliccare sull'evento bandiera verde, dove gli studenti aggiungono alcune istruzioni che la ragazza dice all'inizio.

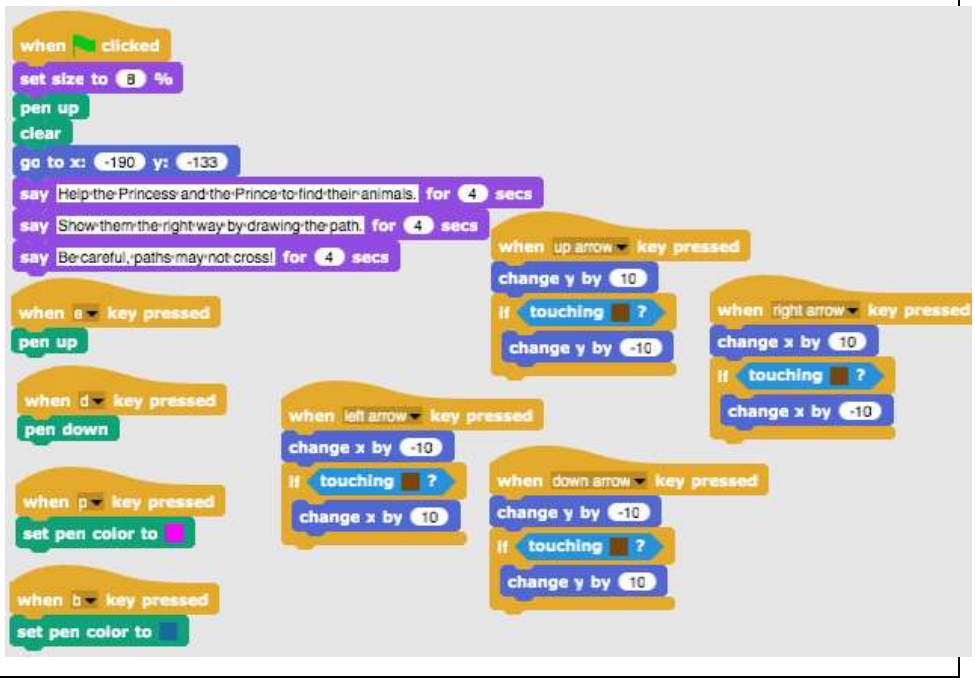
Quando si gioca è bene fermare il gioco e riprenderlo di nuovo, gli studenti vedranno che è utile aggiungere i seguenti blocchi: *penna su* (nel caso in cui fosse rimasto basso dal gioco precedente), *pulisci* (cancella il percorso del gioco precedente) e *vai a x, y* (la ragazza inizia sempre da queste 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]

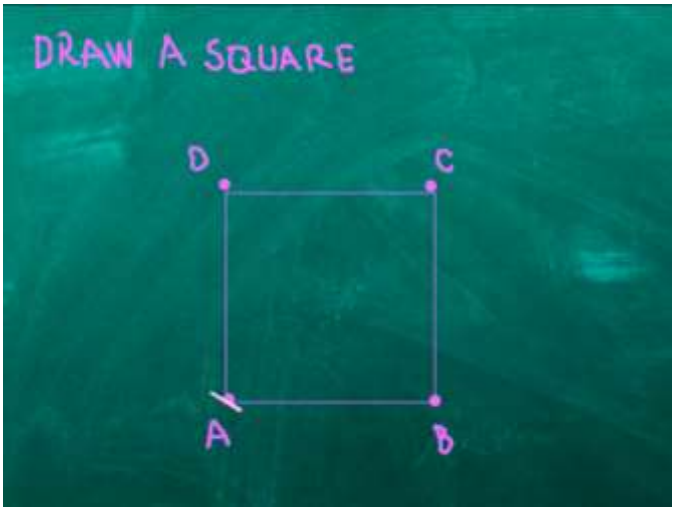
Ragazza



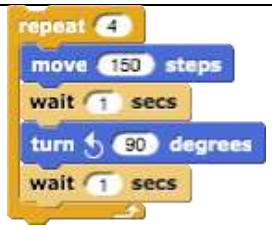
	Es. Principessa [Compiti aggiuntivi] Gli studenti possono aggiungere attività in base ai loro desideri o possono seguire quelle descritte di seguito: ● Imposta le coordinate iniziali per il Principe e la Principessa e scrivi un codice per il loro movimento. Impostare la loro dimensione appropriata. Dovrebbero 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]

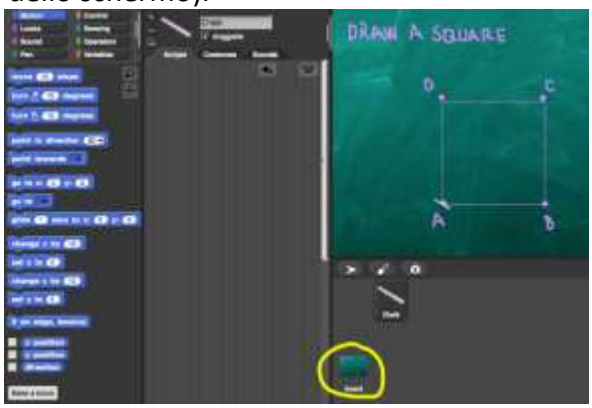
Per l'attivazione del codice, gli studenti utilizzano il blocco eventi, ad es., quando si preme il tasto "S". Possono anche impostare il colore della penna e, come già sanno dalle attività precedenti, i seguenti blocchi: *penna su* (nel caso in cui fosse rimasto basso dal ciclo precedente), *pulisci*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

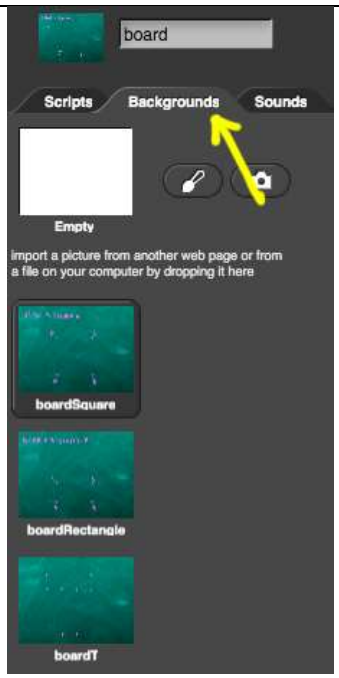
[Fase 3]

Dopo aver disegnato un quadrato, vogliamo disegnare un rettangolo. Questo significa che dobbiamo cambiare lo sfondo. Lo faremo con due passaggi:

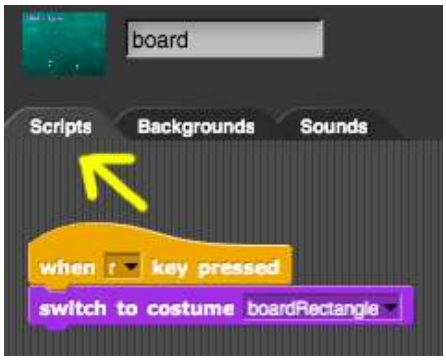
a) clicchiamo sullo sfondo (denominata scheda "*board*", sul lato destro dello schermo).



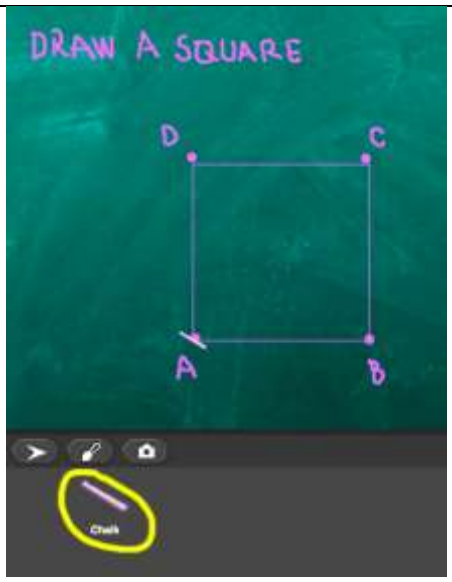
Cliccando su sfondo "*Backgrounds*" possiamo vedere tutti e tre gli sfondi necessari (*boardSquare, boardRectangle, boardT*), già preparati per questa attività.



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*".

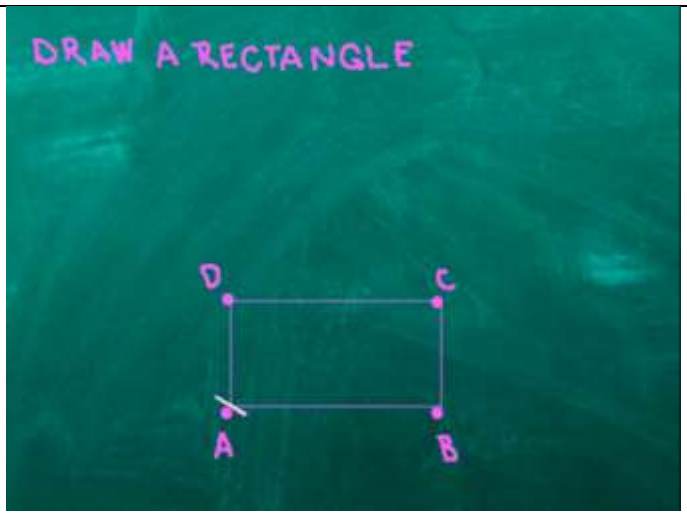


b) clicchiamo di nuovo sul gesso.



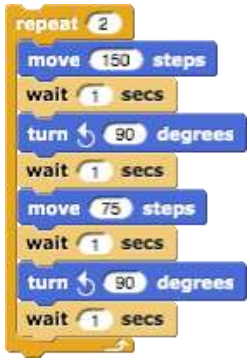
Sotto il codice della [Fase 2] gli studenti aggiungono un blocco, in cui diranno ad un giocatore cosa fare per cambiare lo sfondo, ovvero premere il tasto "R"

[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.



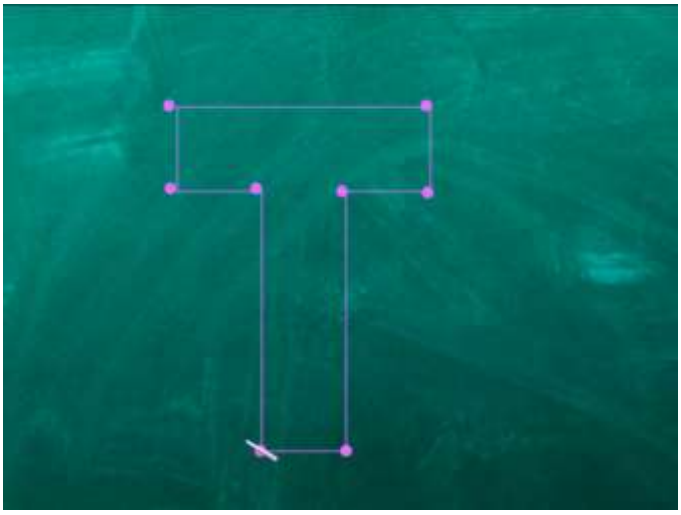
[Fase 5]

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.

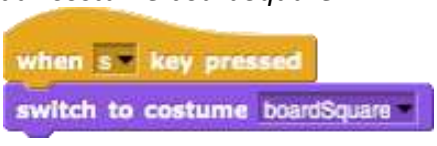
Gli studenti dovranno cambiare le coordinate iniziali, che non sono le stesse di prima. Sanno già come determinare le giuste coordinate dall'attività precedente.

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

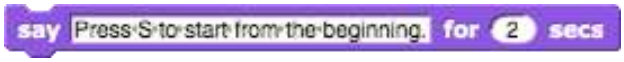
[Fase 7]

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].

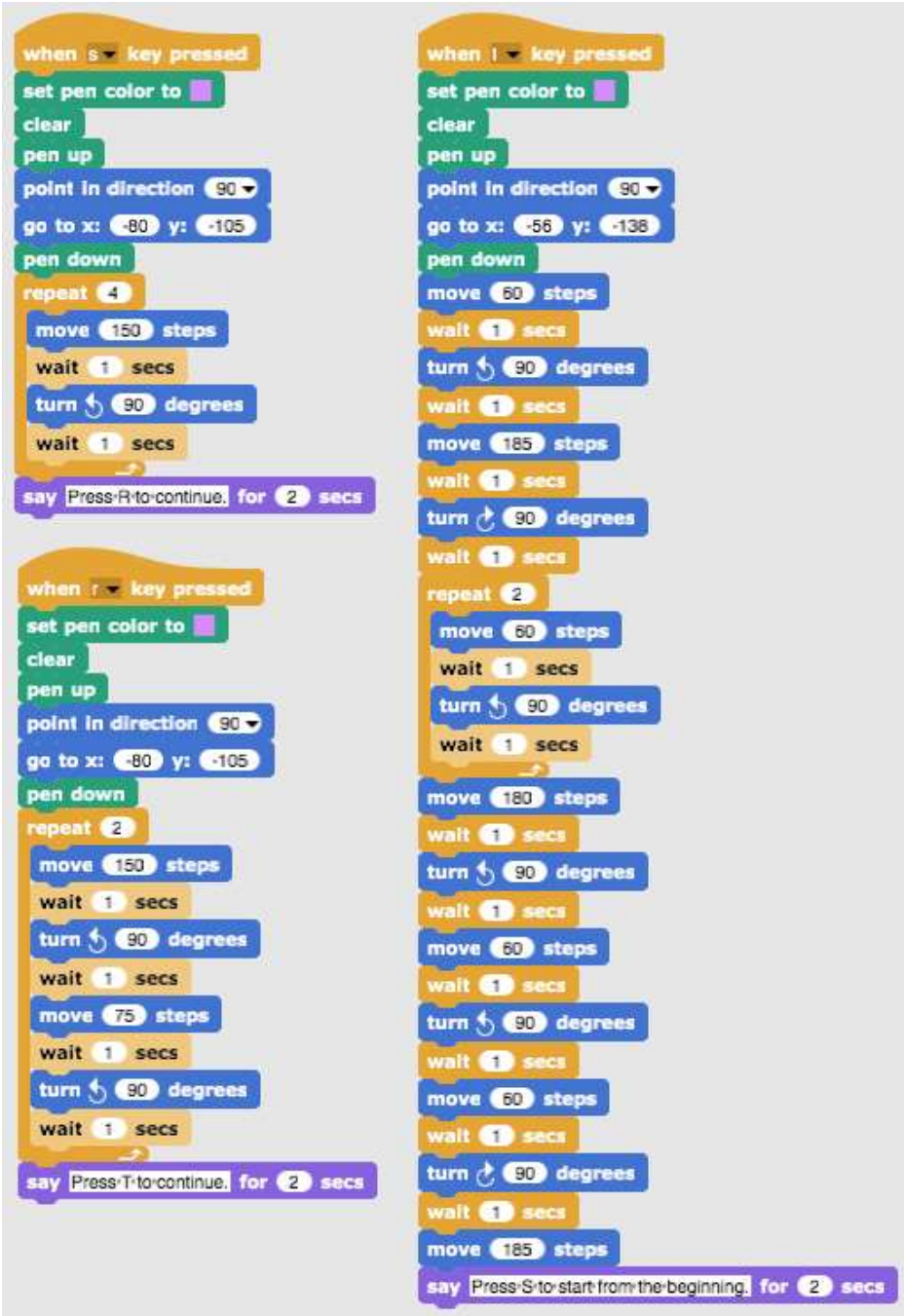
a) Cliccano sullo sfondo (denominato "*board*", sul lato destro dello screen), dove scrivono un codice per cambiare lo sfondo. Lo faranno quando si preme il tasto S, quindi passa alla scheda *"costume boardSquare"*.



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)

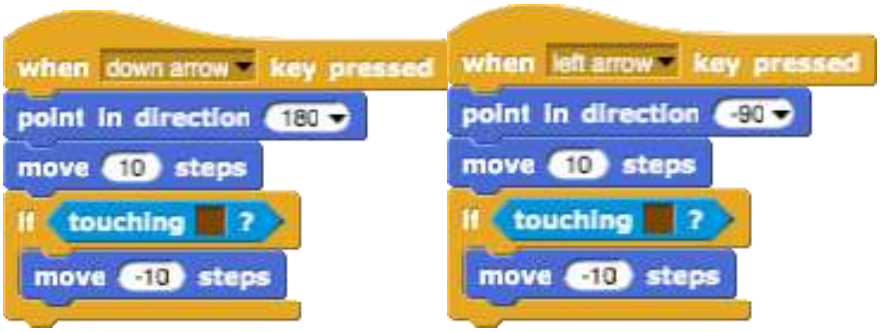
Scenario di apprendimento 9 - Raccogliere la spazzatura e pulire il parco

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

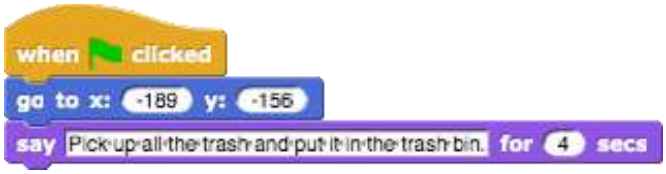
	punto). Quando la ragazza tocca la spazzatura, la raccoglie, la spazzatura si nasconde ed il conteggio dei punti aumenta di 1. Quando raccoglie tutta la spazzatura, si reca al cestino. Se non raccoglie tutta la spazzatura e si reca prima alla pattumiera, questa dice di tornare quando raccoglie tutta 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]

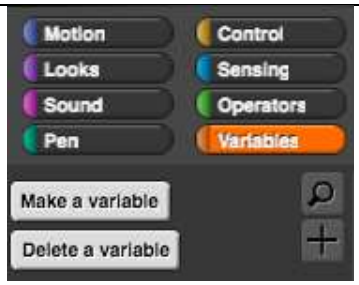
Per contare il numero dei rifiuti raccolti dalla ragazza, utilizzeremo le variabili.

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".

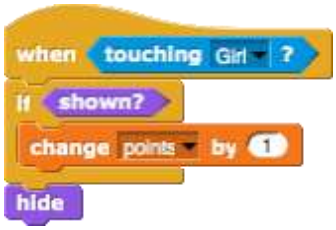
Se la casella è selezionata, la variabile con il suo valore sarà visibile sullo schermo:

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.



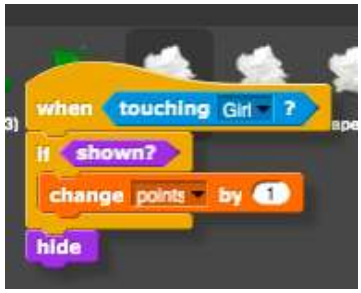
Ora possiamo verificare se il nostro codice sia corretto.

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?

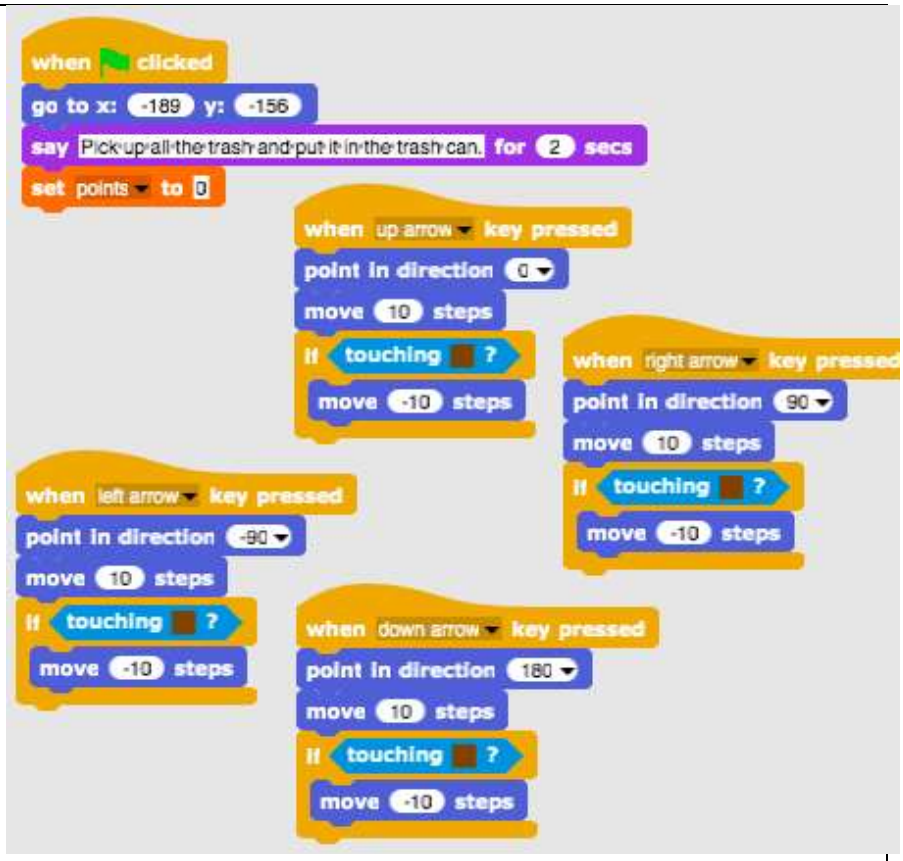
La bottiglia è nascosta, l'abbiamo nascosta prima. Quindi, all'inizio del gioco, dobbiamo programmare che la bottiglia sia mostrata. Lo facciamo selezionando il blocco mostra “*show*”.

[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	**Breve descrizione**: si programma un gioco in cui il giocatore dovrà eseguire dieci calcoli di moltiplicazione e contare le risposte corrette.

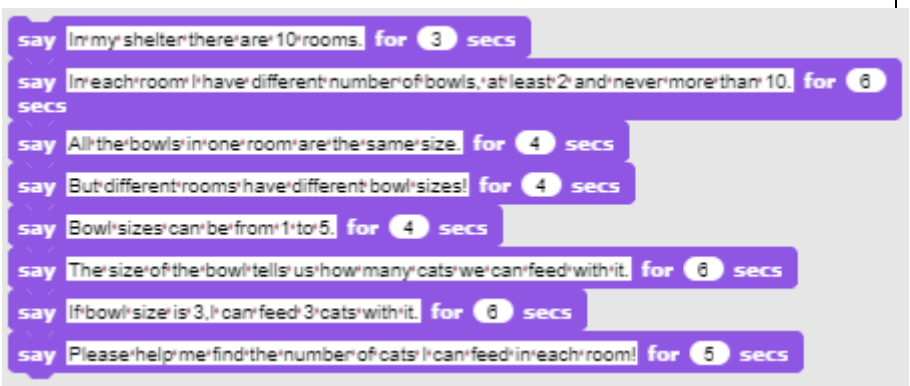
delle Attività	**Compiti**: si programma l'attività in cui Marta, il guardiano del rifugio, chiederà ripetutamente al giocatore il numero di gatti che può nutrire in una determinata stanza. Il numero dipende da quello delle ciotole e dalla loro dimensione. Questi due numeri devono essere assegnati in 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.

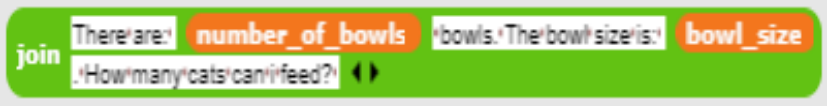
[Fase 4]

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



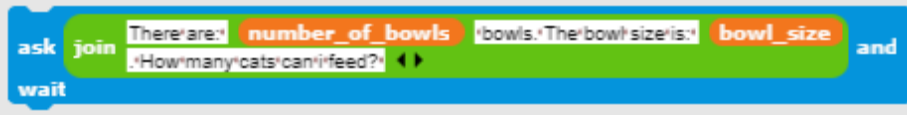
	[Fase 5] Avviamo una discussione con gli studenti in merito a quali sono le azioni che accadranno in ogni stanza. Si tratta di comandi che dovranno essere inseriti all'interno del blocco del ciclo per essere eseguiti in ogni iterazione del ciclo. Per prima cosa dovremo assegnare casualmente il valore (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.



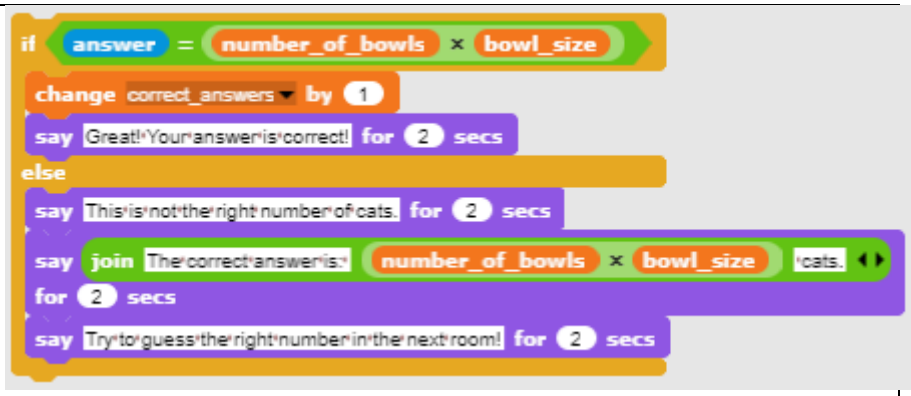
[Fase 8]

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.

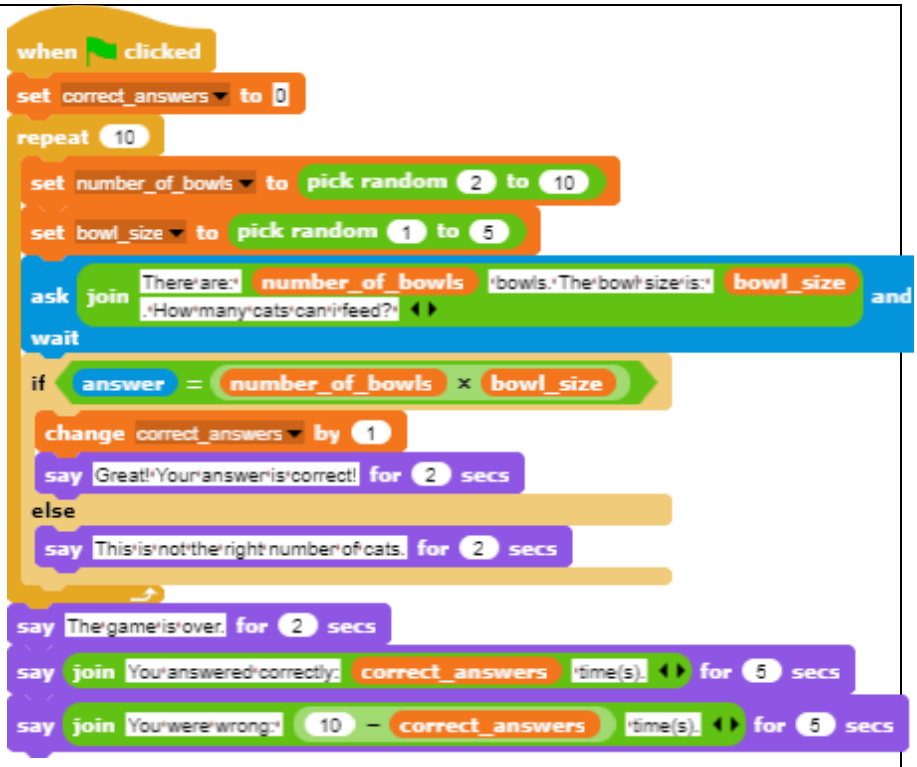
[Fase 12]

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	**Breve descrizione**: si programma un gioco semplice in cui all'inizio un numero casuale da 1 a 100 viene assegnato in modo casuale ad

delle Attività	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.

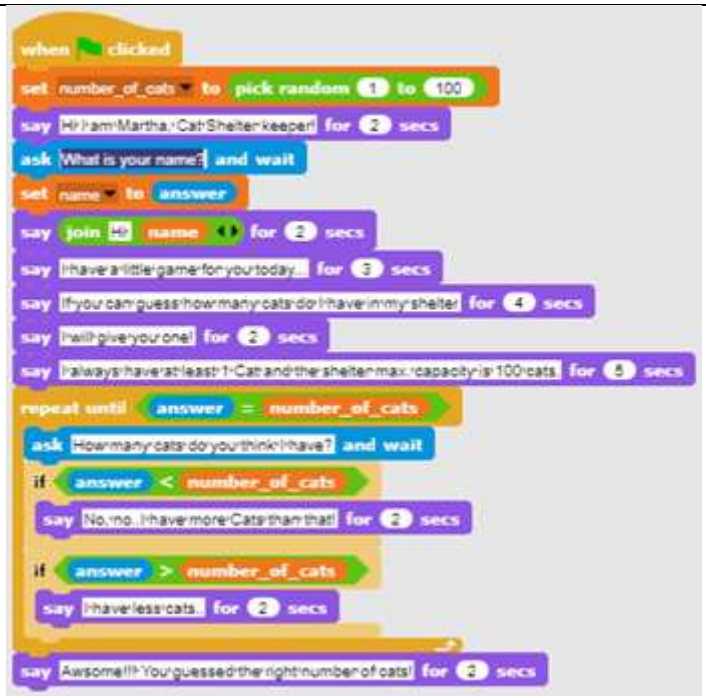
[Fase 5]

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]

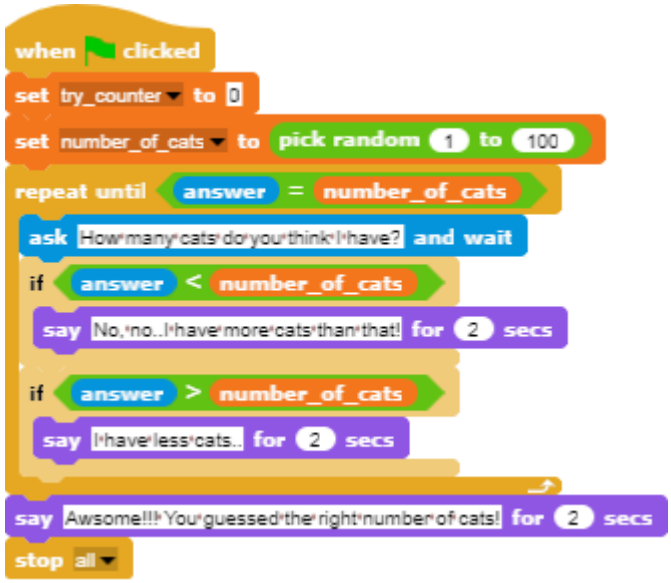
Dopo la risposta giusta, dobbiamo controllare il valore della variabile contatore per decidere se il giocatore otterrà il gatto o meno. Poiché Snap! ha solo un operatore logico inferiore (<) e non un

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 delle Attività	**Breve descrizione:** La ragazza sta prendendo cibo. Deve stare 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:

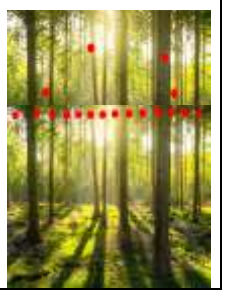
Se non vogliono che la mela si capovolga, possono scegliere la terza opzione che non ruota nel blocco di direzione.

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).

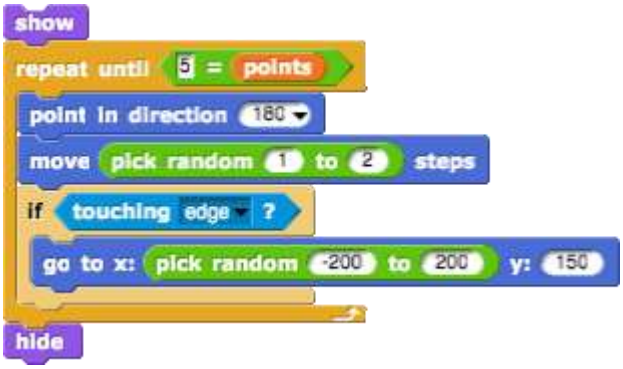
[Fase 4]

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]

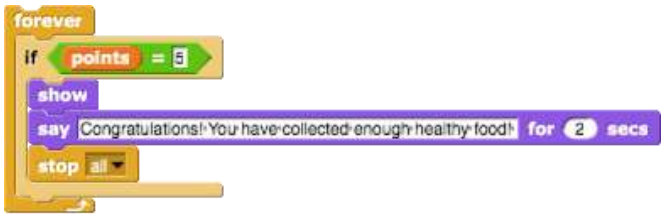
Torniamo alla ragazza.

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.

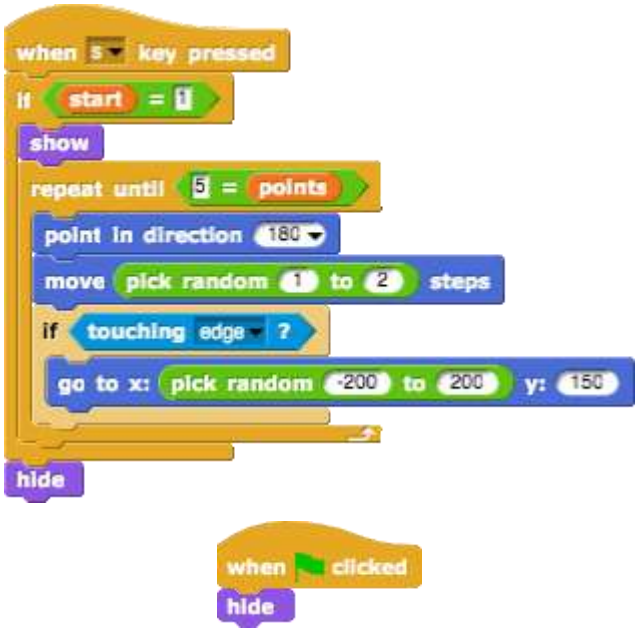
Per evitarlo, possiamo creare un'altra variabile (denominata start), che deve essere impostata su 0 all'inizio. Quindi, dopo le istruzioni della ragazza, la variabile inizia cambierà in 1.

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:



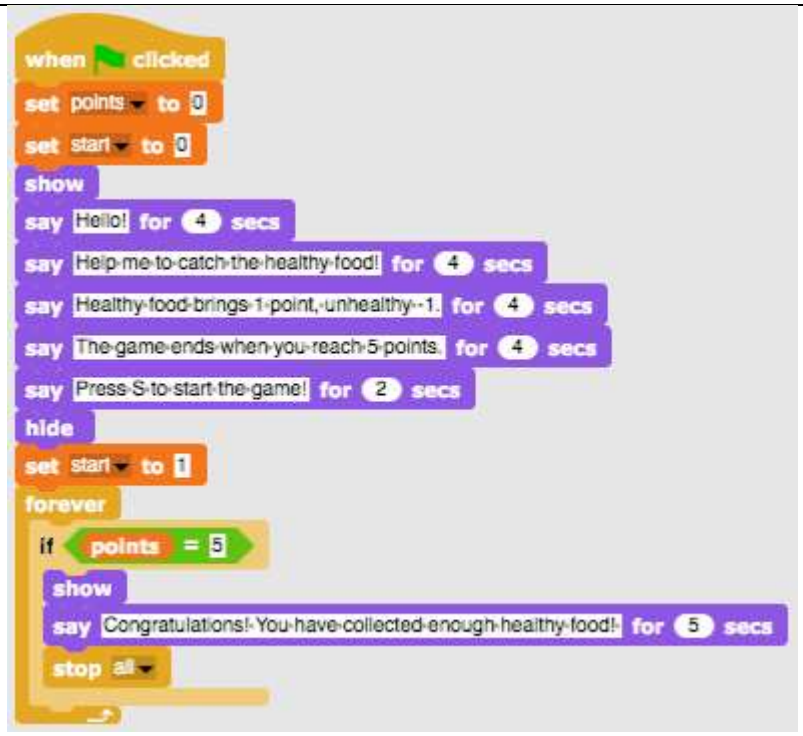
[Fase 9]

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

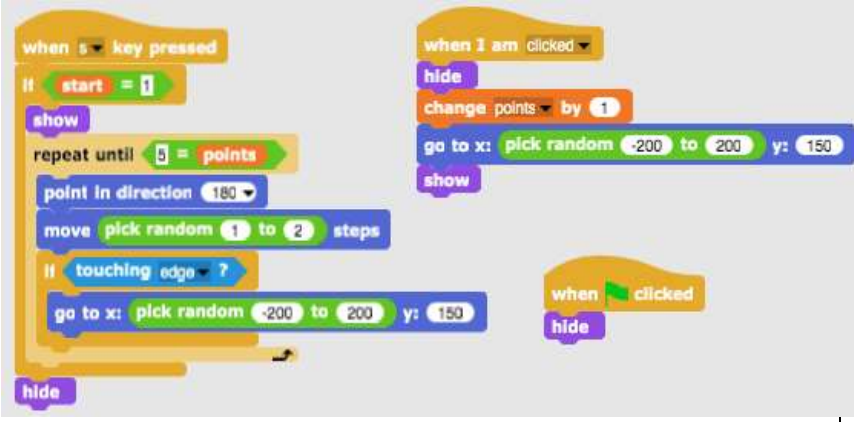
L'unica modifica è con il cibo malsano, dove perderanno 1 punto facendo clic su di esso.

[Codice finale]

Ragazza



Mela



[Compiti aggiuntivi]

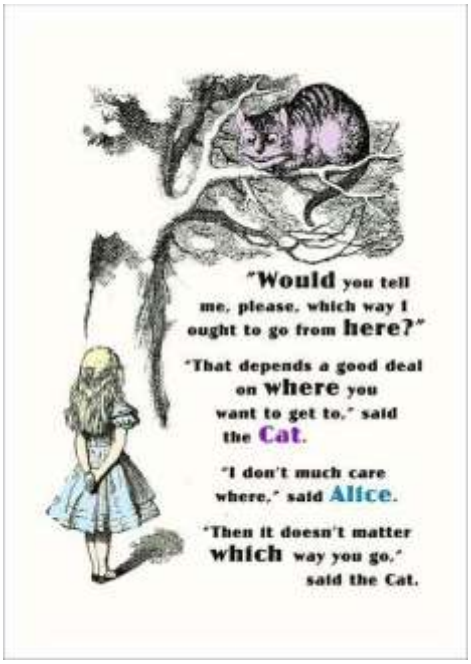
Gli studenti possono aggiungere attività in base ai loro desideri o possono seguire le seguenti:

- 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

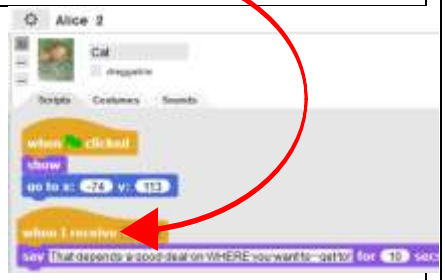
La conclusione è che la sincronizzazione con il blocco *aspetta n… secondi* può portare errori nel comportamento dei personaggi nella narrazione.

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*

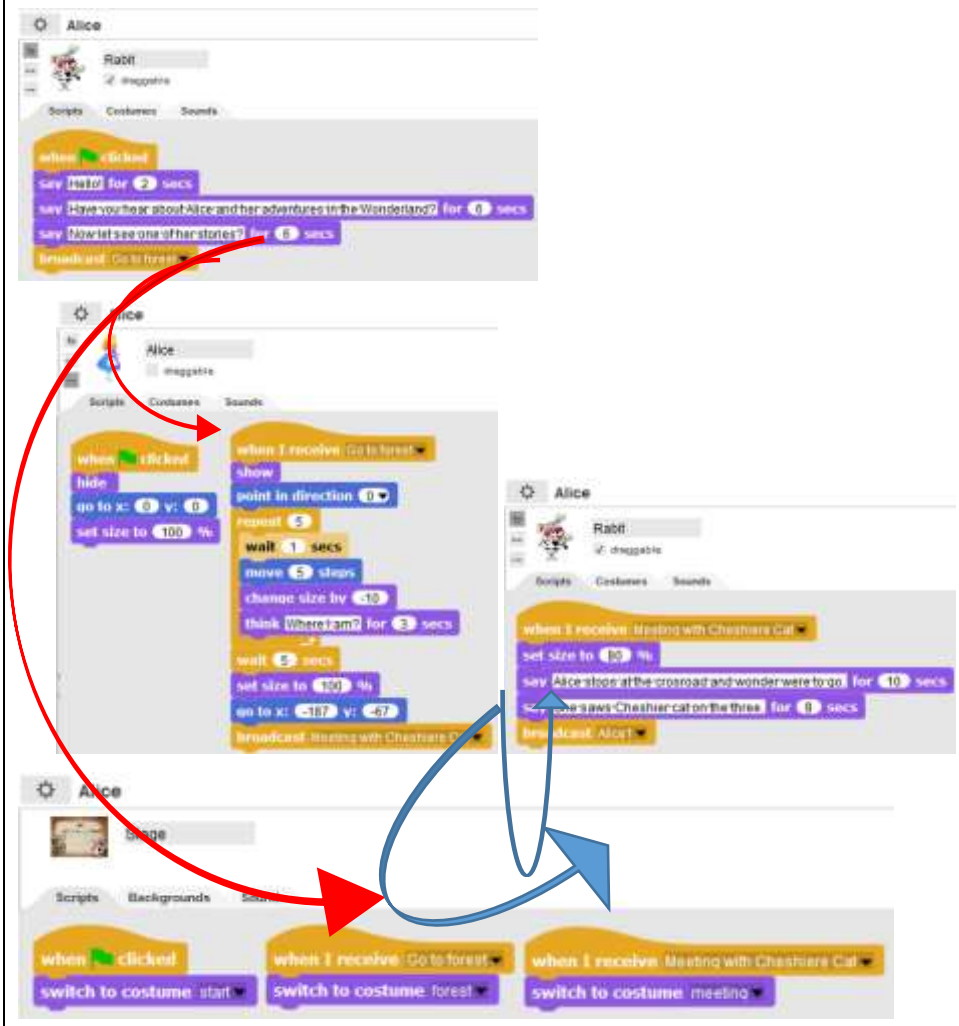
gradualmente man mano che si allontana. Quindi si ritrova ad un bivio e vede il gatto Cheshire. La conversazione tra i due inizia".

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*

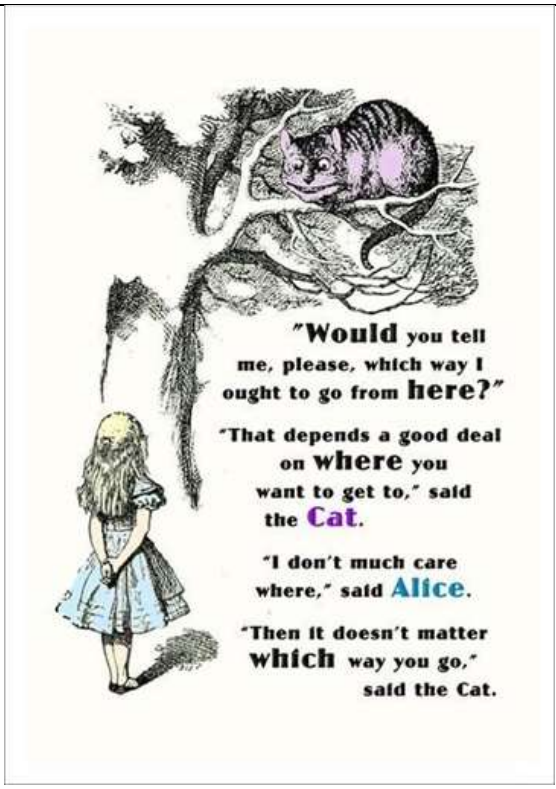
azioni? Quando appare il gatto e quali sono le sue azioni?".

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

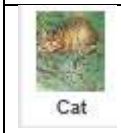
Risorse/materiali per gli studenti	 • https://snap.berkeley.edu/project?user=ddureva&project=Alice_1 • https://snap.berkeley.edu/project?user=ddureva&project=Alice_2 • Istruzioni per lo studente (C4G13_InstructionsForStudent.docx)

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 Cat*1*.	
	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 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) 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*.

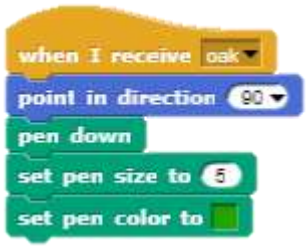
Deseleziona il pulsante "trascinabile" così lo *sprite* non può essere spostato.

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.

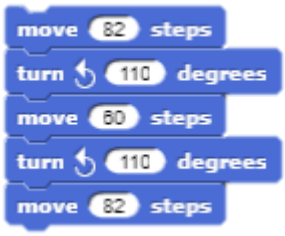
Al termine del disegno, è necessario sollevare la penna in modo che non tracci la linea quando si sposta lo *sprite* della matita.

[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.

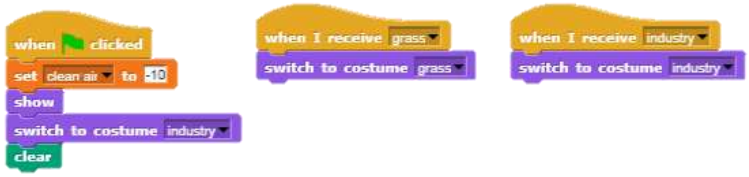
[Codice finale]

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.

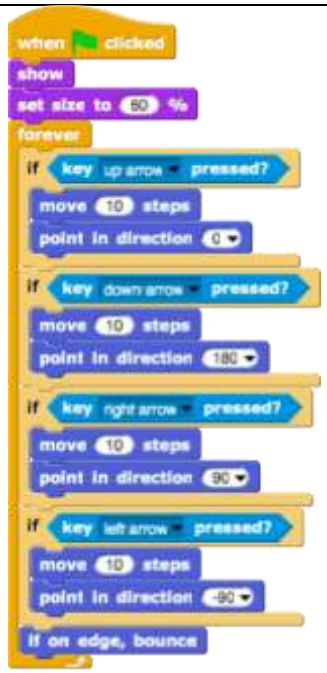
Gli studenti programmano i seguenti compiti in coppia/gruppo con il supporto dell'insegnante.

[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.



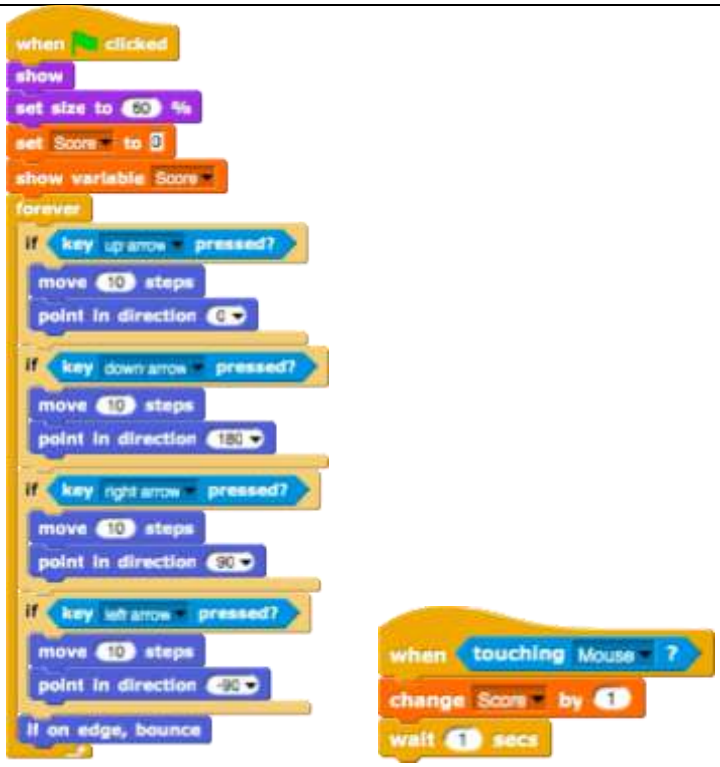
[Fase 3]

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]

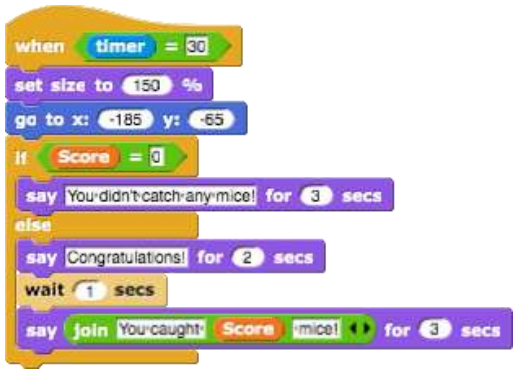
Gli studenti determinano quando finisce il gioco. Lo fanno aggiungendo il *timer*. Dopo un po' di tempo (ad es. 30 secondi) il topo e il gatto scompaiono, la variabile punteggio viene nascosta e il gioco termina.

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*

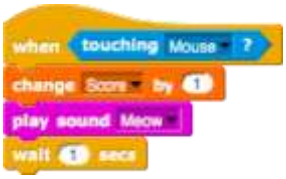
topi!"



[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:

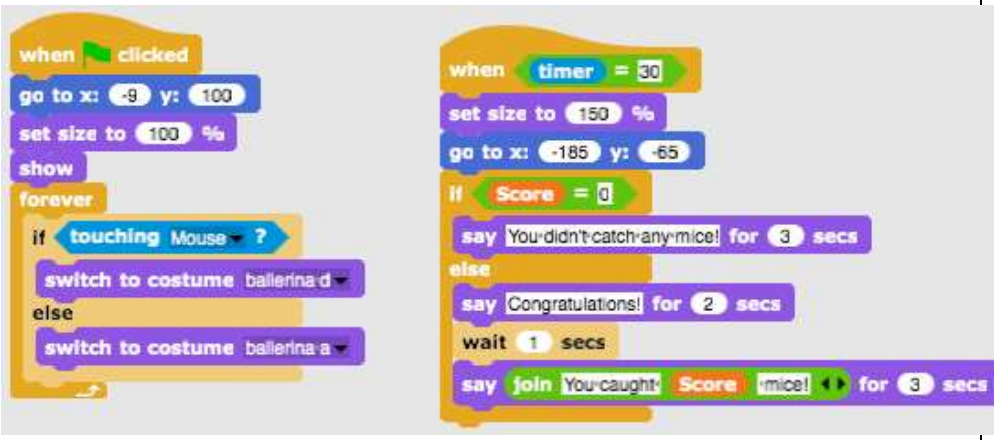
- il topo si muove da 20 a 60 passi per sempre

- 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*.

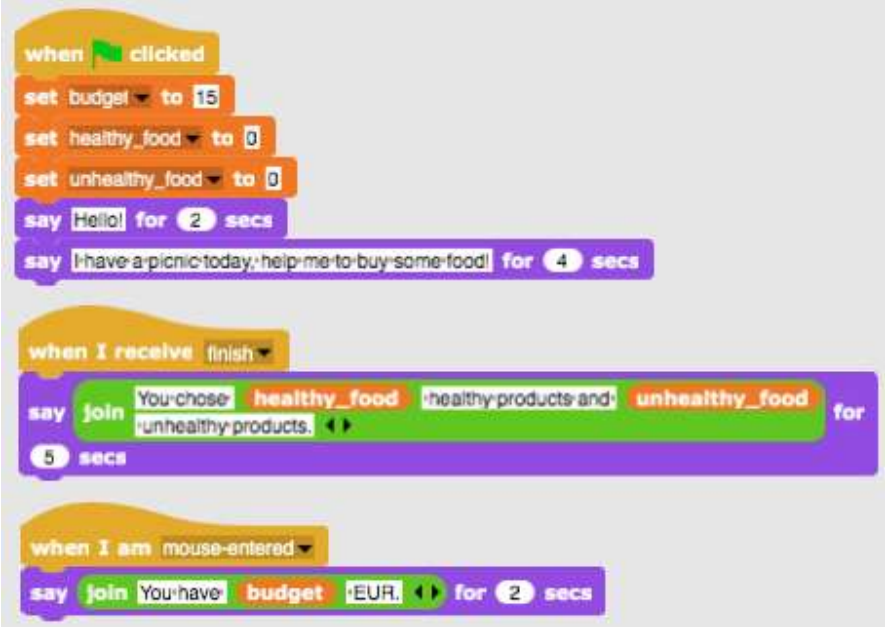
Quando la ragazza riceve il messaggio di *Fine*, dice, ad es. "Hai scelto X prodotti sani e Y prodotti non salutari".

[Fase 7]

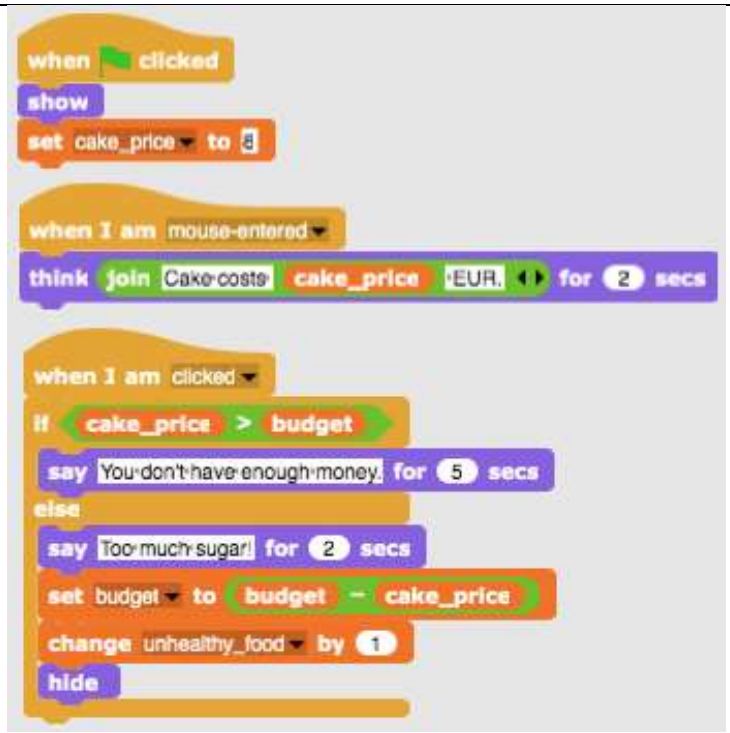
In qualsiasi momento del gioco, il giocatore può controllare il proprio budget utilizzando il *mouse-enter* sulla ragazza. Ad esempio, può dire / pensare qualcosa del genere:

[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.

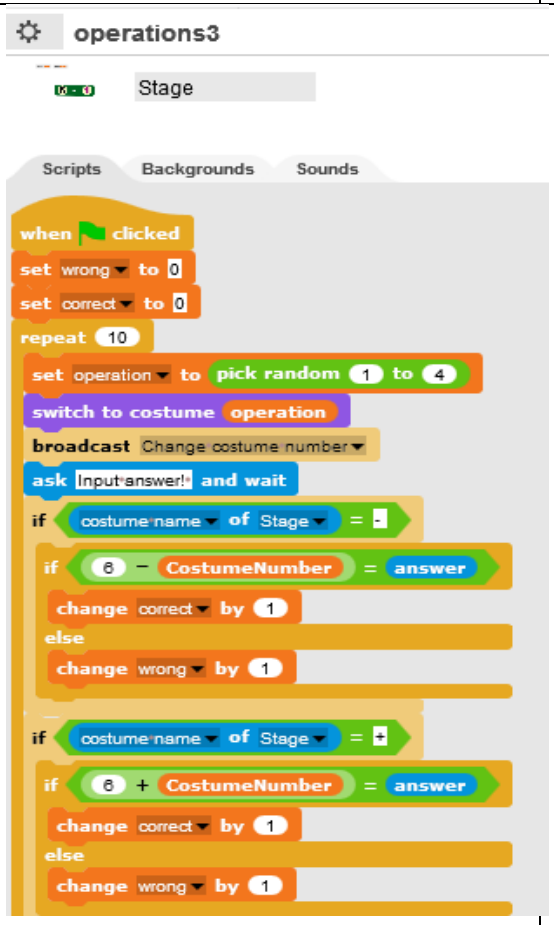
3. Le variabili sono commentate, così come il modo in cui sono definite, impostate e modificate.

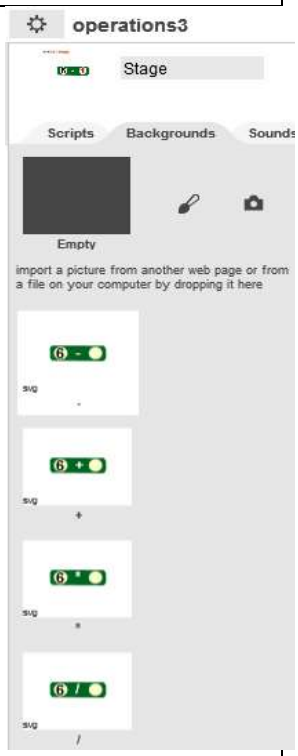
4. Vengono rivisti i comandi dei numeri casuali, le operazioni aritmetiche e logiche, i comandi degli eventi di trasmissione.

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*"

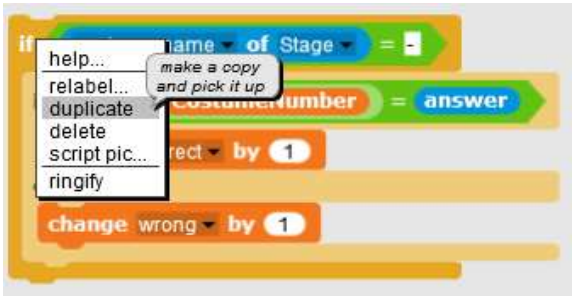
Se viene selezionata l'operazione "-" viene eseguito un controllo se il risultato di 6 - "Numero costume Sprite" corrisponde alla risposta. Se corrisponde la variabile corretta aumenta altrimenti aumenta la variabile per il conteggio delle risposte errate.

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



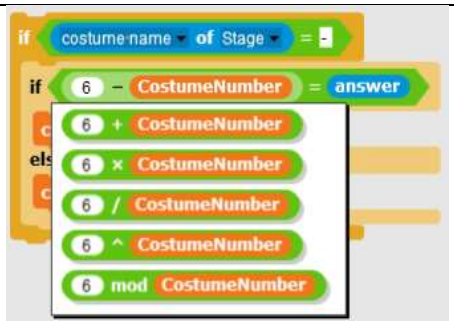
3. utilizzare il mouse per posizionare lo *script* duplicato nella posizione corrispondente.

   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	Mostrare e nascondere lo *sprite* Usare le variabili per il conteggio dei punti

programmazione	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.

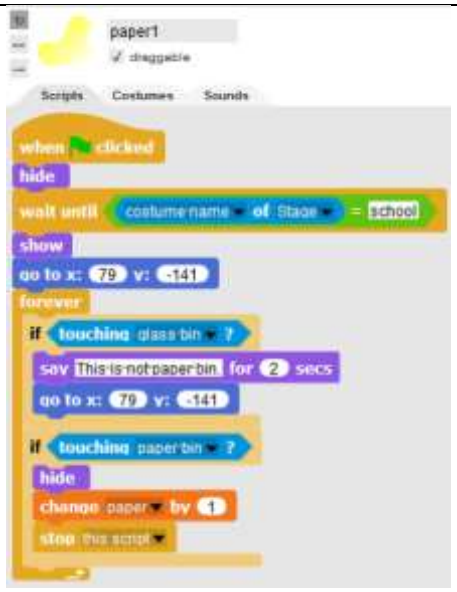
2. Esame dei codici di scene e personaggi.

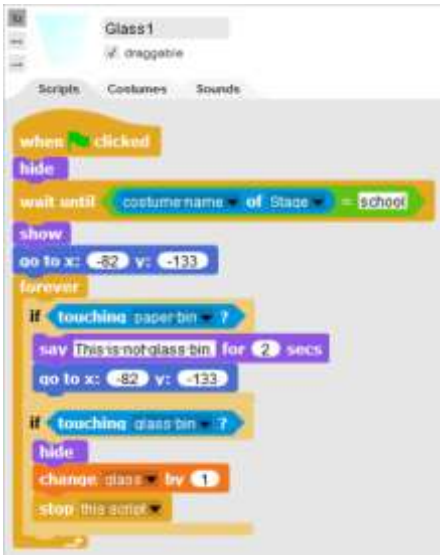
Dopo aver condiviso le possibilità di risolvere il problema, vengono discussi i codici per la scena e i personaggi.

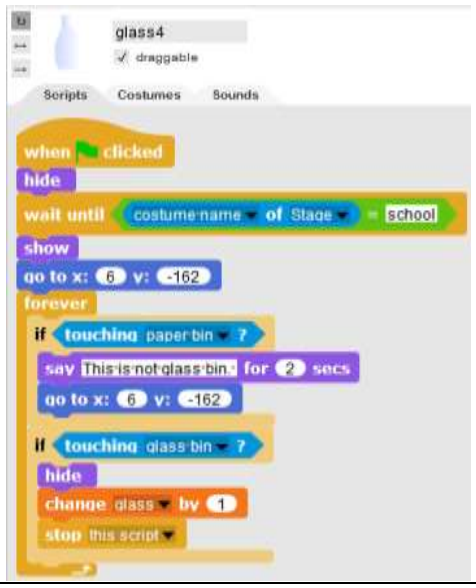
Il codice scena viene commentato ponendo enfasi su:

- 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.


paper1
draggable
Scripts
Costumes
Sounds
when clicked
hide
wait until costume name of Stage = school
show
go to x: 79 y: -141
forever
if touching glass bin ?
say This is not paper bin for 2 secs
go to x: 79 y: -141
if touching paper bin ?
hide
change paper by 1
stop this script


Glass1
draggable
Scripts
Costumes
Sounds
when clicked
hide
wait until costume name of Stage = school
show
go to x: -82 y: -133
forever
if touching paper bin ?
say This is not glass bin for 2 secs
go to x: -82 y: -133
if touching glass bin ?
hide
change glass by 1
stop this script


glass4
draggable
Scripts
Costumes
Sounds
when clicked
hide
wait until costume name of Stage = school
show
go to x: 6 y: -162
forever
if touching paper bin ?
say This is not glass bin for 2 secs
go to x: 6 y: -162
if touching glass bin ?
hide
change glass by 1
stop this script

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.

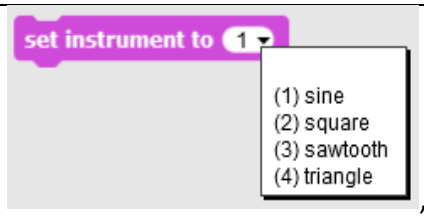
Strumenti e risorse per gli insegnanti	L'intera attività in Snap!: https://snap.berkeley.edu/project?user=ddureva&project=recycling L'intera attività in Scratch: ● Дурева Д., М. Касева, Г. Тупаров, Компютърно моделиране, 4. клас, Просвета, 2018, София (Dureva, D., M. Kaseva, G. Tuparov, Kompyutarno modelirane, 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 attivo (discussioni, esperimento con il gioco preparato precedentemente), apprendimento basato sulla

apprendimento e insegnamento	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	E b	E	F	F #	G	G #	A	B b	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.

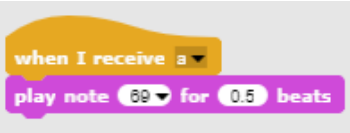
Deselezionato il pulsante "trascinabile", quindi gli sprite dei tasti non possono essere spostati durante la riproduzione.

[Fase 2]

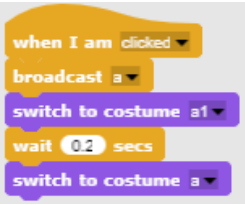
Abilita la riproduzione dei toni premendo gli sprite. Per il tasto "A", aggiungi il cappello "Quando faccio clic" e consenti di trasmettere il messaggio "a".

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.

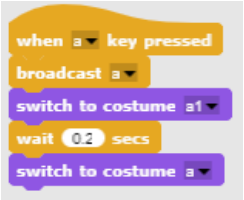


[Fase 3]

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]

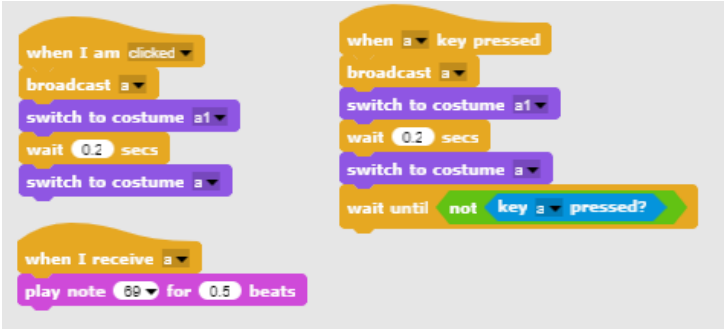
Crea un nuovo pulsante X che ripristinerà il progetto (senza le note).

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

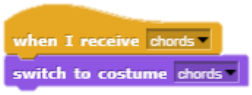
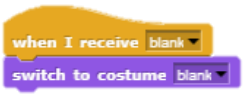


Chiave di violino

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

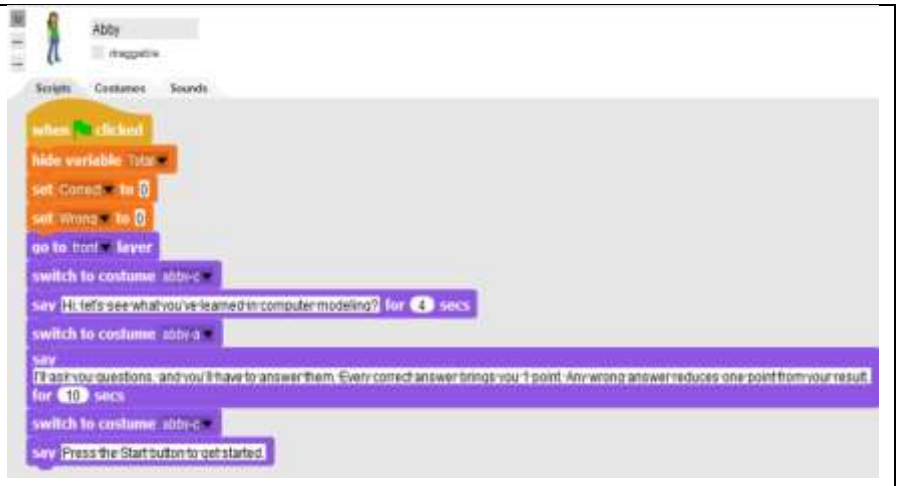
domanda;

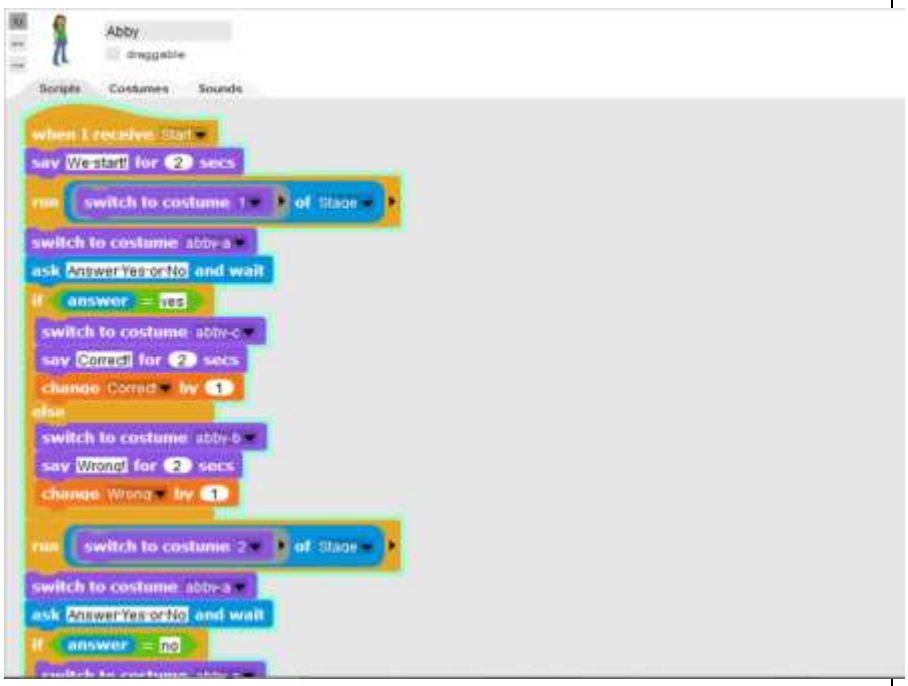
- Abby dice – Risposta Sì o No;
- il giocatore inserisce una risposta - Sì o No;

• 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.

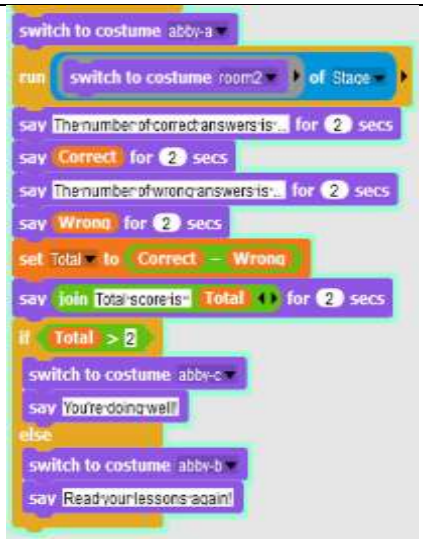
   Vengono commentati i comandi per avviare un dialogo con l'utente, per modificare la scena e il costume del personaggio e i comandi condizionali. Vengono esaminati i codici di ciascun personaggio. Si commenta la creazione di una variabile.





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;

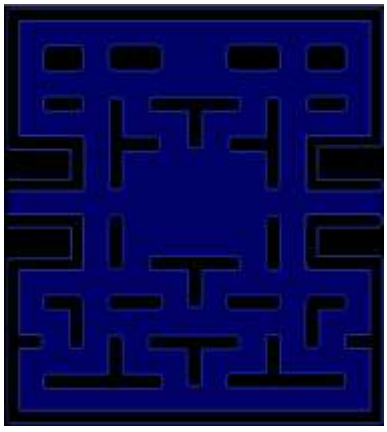
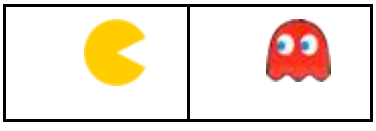
4. viene fornito agli studenti un file non completo di costumi per la scena e gli *sprite*. Se lo desiderano, possono anche creare un file tutto loro. Il lavoro viene svolto per analogia con il test del modello.

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

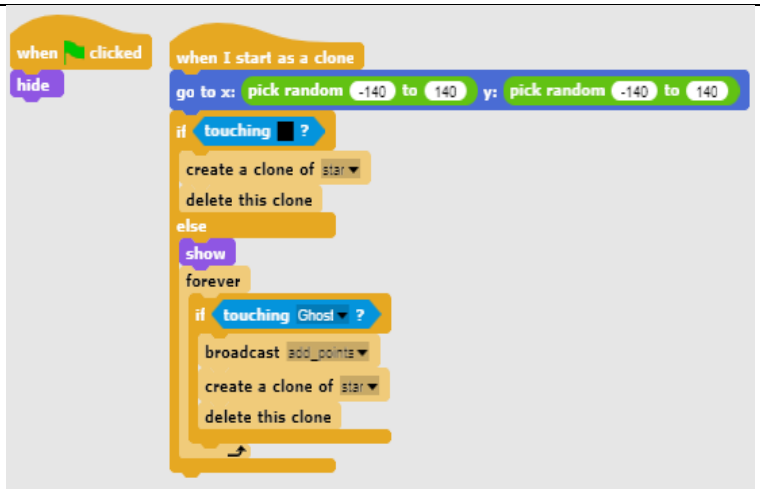
Riepilogo dell'insegnamento	(Motivazione-introduzione, attuazione, riflessione e valutazione) Il giocatore sta raccogliendo stelle posizionate casualmente mentre viene inseguito da un fantasma rosso. Se un giocatore e un fantasma si scontrano, il gioco è finito. Se un giocatore raccoglie le 20 stelle, 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.



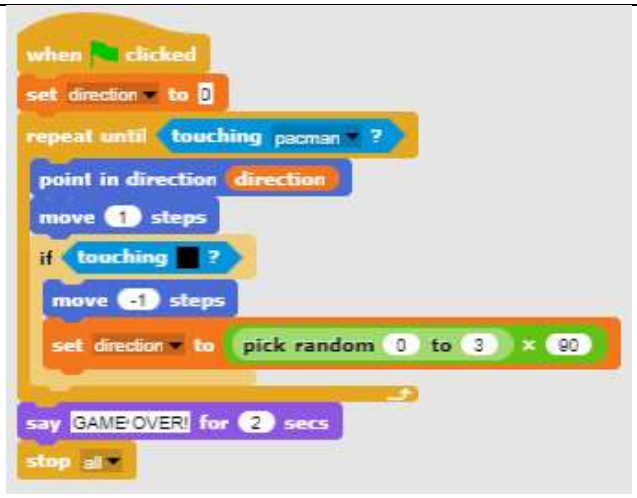
[Fase 5]

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.

<table>
<tr>
<td></td>
<td>


[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.

</td>
</tr>
<tr>
<td>Strumenti e risorse per gli insegnanti</td>
<td>

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.

</td>
</tr>
</table>

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)

APPENDICE 2. CODICI DEGLI SCENARI DI APPRENDIMENTO

SCENARI DI APPRENDIMENTO DI BASE

Scenari in INGLESE

N.	TITOLO	CODICE
1	**Introduction to Snap! Interface** Students add a new sprite, add a costume to the sprite, edits the costume, and deletes one of them. Student creates a new background to the stage, edits it, and deletes unwanted ones.	**starting**
2	**Discover Snap! : move a spite** Help the students to discover the Snap! interface and code their first sprite so that it moves and speaks.	**dispubeng**
3	**Moving around the stage** Students learn how to move the sprite in x and y direction on the stage, code an easy program to solve the tasks given, learn how to turn the sprite in a different direction.	**monkey**
4	**Changing costumes and turning** Students learn how to change the sprite's costume to make an animation by changing between different types of rotation.	**Dancer**
5	**Sounds of the farm** Students learn how to program a simple game in which player can recognize the sounds of animals by pressing certain keys.	**farm**
6	**Chameleon's summer vacation** Program a simple game in which an object will change its costume based on the color of the background	**chameleoneng**
7	**Helping Prince and Princess to find their animals** A girl has to help the Princess find her cat and the Prince find his dog by avoiding that the animals can meet each other during their path.	**finding**
8	**Drawing with a chalk** Students will be introduced into drawing different shapes with a code. They will learn to use loop repeat for shorten the code and to change a background.	**chalk**
9	**Picking up trash and cleaning the park** Students will learn how to use variables and how to duplicate a block of code or even a whole sprite.	**cleaning**
10	**Feeding the cats** Students will be introduced to the concept of multiple variable random value assignment inside a loop and how it is different from when we do it outside a loop. They will also learn about	**feeding**

	how to get, test and count correct player inputs.	
11	**Guessing the number of cats in a shelter** Students will be introduced to "repeat until" loop and how to set the condition to implicitly track the condition that stops the game. They will also learn how to use variable in a different situations: to set a random value, as a counter or to get the players input.	**shelter**

Scenari in SLOVENIO

N.	TITOLO	CODICE
1	**Uvod v Snap!** Učenec doda nov lik, doda obleko, uredi obleko in izbriše obleko/lik. Učenec ustvari novo ozadje, ga uredi, zbriše odvečna ozadja.	**uvod_slo**
2	**Razišči Snap!: premikaj lik** To uro boš spoznal/a, kako z bloki likom naročimo naj se premikajo po odru in govorijo.	**razisci_slo**
3	**Premikanje po odru** Učenec se nauči, kako premikati lik v x in y smeri, kako napisati preprost program ter kako obrniti lik v drugo smer	**premikanje_slo**
4	**Menjava obleke in obrat** Učenec se nauči, kako liku zamenjati obleko in kako narediti animacijo z obračanji lika	**obleka_slo**
5	**Zvoki na kmetiji** Učenec se nauči programiranja preproste igre, v kateri lahko igralec s pritiskom na določene tipke prepozna zvoke živali	**kmetija_slo**
6	**Kameleon na počitnicah** Izdelaj preprosto igro, v kateri bo glavni objekt spreminjal svojo obleko glede na barvo ozadja na njegovi trenutni poziciji.	**kameleon_slo**
7	**Pomagaj princu in princeski najti svoje živali** Dekle mora pomagati princesi poiskati svojo mačko in princu svojega psa. Pri tem se mora izogniti srečanju med živalmi.	**iskanje_zivali_slo**
8	**Risanje s kredo** Učenec se seznani z risanjem različnih oblik s kodo. Nauči se uporabljati zanko ponovi za krajšanje kode ter kako zamenjati ozadje.	**kreda_slo**
9	**Pobiranje smeti in čiščenje parka** Učenec se nauči uporabljati spremenljivke in kako podvojiti blok kode ali celotno kodo lika.	**ciscenje_parka_slo**
10	**Nahrani mucke** Učenec se seznani s konceptom dodeljevanja več spremenljivk z naključno vrednostjo znotraj zanke in kakšna je razlika, če vrednost dodelimo zunaj zanke. Spoznali bodo tudi, kako pridobiti in preveriti igralčev odgovor ter kako šteti pravilne odgovore.	**mucke_slo**
11	**Mačje zavetišče** Učenec se seznani z zanko "ponavljaj dokler" in kako ustvariti pogoj za implicitno sledeje pogoju, ki konča z igro. Nauči se tudi, kako uporabljati spremenljivko v različnih situacijah: nastaviti naključno vrednost, kot števec ali kot igralčev vnos.	**zavetisce_slo**

Scenari in ITALIANO

N.	TITOLO	CODICE
1	**Snap!: introduzione** Lo studente impara ad aggiungere un nuovo sprite, a modifica il costume e ad eliminarne uno di essi. Impara a crea un nuovo sfondo per il palco, lo modifica e cancella quelli indesiderati.	**Primipassi**
2	**Scoprire Snap!** Imparare a programmare un gioco semplice dove l'oggetto cambia il proprio colore secondo il colore dello sfondo	**Snap!**
3	**Muoversi sullo "stage"** Il discente impara a spostare il suo Sprite in direzione x e y sullo "stage"; a preparare un semplice programma per risolvere i compiti assegnati; a far girare il suo Sprite in diverse direzioni.	**stage**
4	**Cambiare il costume e creare rotazioni** Lo studente impara a cambiare il costume dello sprite per realizzare un'animazione.	**ballerina**
5	**I suoni di una fattoria** Gli studenti imparano come programmare un gioco semplice in cui il giocatore può riconoscere i suoni degli animali premendo determinati tasti.	**fattoria**
6	**Vacanze estive di un Camaleonte** Imparare a programmare un gioco semplice dove l'oggetto cambia il proprio colore secondo il colore dello sfondo	**camaleonte**
7	**Alla ricerca degli animali del Principe e della Principessa!** La ragazza deve aiutare la principessa a trovare il suo gatto e il principe a trovare il suo cane evitando che gli animali possano incontrarsi durante il loro.	**labirinto**
8	**Disegnare con un gesso** Gli studenti impareranno a disegnare forme diverse con un codice e ad usare la ripetizione ciclica per abbreviare il codice e cambiare lo sfondo.	**gesso**
9	**Raccogliere la spazzatura e pulire il parco** Gli studenti impareranno ad usare le variabili e a duplicare un blocco di codice o anche un intero Sprite.	**spazzatura**
10	**Dare da mangiare ai gatti** Gli studenti impareranno il concetto di assegnazione di più valori casuali, variabili all'interno e al di fuori di un ciclo. Impareranno anche come ottenere, testare e contare gli input corretti immessi dal giocatore.	**gatto**
11	**Indovinare il numero di gatti in un rifugio** Gli studenti impareranno ad utilizzare il ciclo "ripeti fino a" e ad impostare la condizione per interrompere il gioco. Impareranno anche ad usare la variabile in situazioni diverse.	**rifugio**

Scenari in CROATO

N.	TITOLO	CODICE
1	**Uvod u sučelje alata Snap!** Učenik dodaje nove objekte, dodaje kostime objektima, uređuje kostime te ih briše. Učenik stvara nove pozadine na pozornici, uređuje ju, te neželjene briše.	**start_hr**
2	**Otkrijte Snap!: kretanje sprite** Učenici otkriva gdje pronaći programske blokove i povezati ih u niz. Učenici će naučiti kako micati objekt i omogućiti da objekt govori.	**disc_cro**
3	**Kretanje po pozornici** Učenici će vidjeti kako izraditi program u kojem će se objekt kretati u smjeru x, te izraditi program u kojem će se objekt kretati u smjeru y.	**kretanje**
4	**Mijenjanje kostima i okretanje** Učenik uči kako promijeniti kostim objekta te kako napraviti animaciju. Također uči kako se između njih može mijenjati različite vrste rotacije objekta.	**ples**
5	**Zvukovi s farme** Učenici će se upoznati kako programirati jednostavnu igru u kojoj igrač može prepoznati zvukove životinja pritiskom na određene tipke.	**farma**
6	**Kameleonov ljetni odmor** Učenici će biti upoznati s blokom naredbi osjeta boja i kako ga koristiti u logičkim izrazima da bi razlikovali dinamično promjenjiva stanja igre i dali prave odgovore.	**chameleon_cr o**
7	**Pomaganje princu i princezi u pronalasku njihovih životinja** Učenici će se upoznati s crtanjem pokretom tipke. Uz to će naučiti kako koristiti uvjete kojima će spriječiti da se objekt kreće po cijelome ekranu.	**finding_hr**
8	**Crtanje s kredom** Učenici će se upoznati s crtanjem različitih oblika pomoću kôda. Naučit će koristiti petlju ponavljaj kako bi smanjili veličinu kôda i promijeniti pozadinu.	**kreda**
9	**Skupljanje otpada i čišćenje parka** Učenici će se upoznati kako koristiti varijable i kako kopirati blok kôda ili čak cijeli objekt.	**park**
10	**Hranjenje mačaka** Učenici će biti upoznati sa konceptom dodjele slučajne vrijednosti varijabli unutar petlje te će razlikovati kada navedeno napravimo van petlje. Naučiti će kako dobiti, testirati i izbrojati ispravne unose igrača.	**hranjenje**

11	**Pogađanje broja mačaka u skloništu** Učenici će se upoznati s petljom ponavljaj dok i kako postaviti uvjet koji zaustavlja igru. Također će naučiti kako koristiti varijable u različitim situacijama: za pohranjivanje nasumične vrijednosti, kao brojač ili za pohranjivanje vrijednosti koju upiše igrač.	**pogodi**

Scenari in BULGARO

N.	TITOLO	CODICE
1	**Увод в Snap!** Ученикът добавя нов спрайт, добавя костюм към спрайта, редактира костюма и изтрива едино от двете. Ученикът създава нов фон на сцената, редактира го и го изтрива.	**start_bg**
2	**Да разучим Snap!: Движещ се спрайт** Помогнете на учениците да разучат интерфейса на Snap! и да съставят код за първия им спрайт, така че той да се движи и говори.	**dissnap_bg**
3	**Движейки се по сцената** ченикът се научава как да движи спрайта в x и y посока на сцената, създава лесна програма за решаване на задачите и се научава как да завърти спрайт в различна посока.	**monkey_bg**
4	**Смяна на костюми и завъртане** Ученикът се учи как да промени костюма на спрайт при завъртане, за да направи анимация.	**dancer_bg**
5	**Звуци от фермата** Учениците научават как да програмират игра, в която играчът може да разпознава звуците на животните чрез натискане на определени клавиши.	**sounds_bg**
6	**Лятната ваканция на хамелеона** Програмирайте проста игра, в която обект да променя костюма си в зависимост от цвета на фона.	**cham_bg**
7	**Помогнете на принца и принцесата да намерят своите домашни любимци** Момиче трябва да помогне на принцесата да намери котката си, а на принцът да намери кучето си, като трябва да избегне възможността животните да се срещнат по пътя.	**find_bg**
8	**Рисуване с тебешир** Учениците ще бъдат запознати с рисуването на различни фигури(форми) използвайки код. Те ще се научат да използват цикъл с повторение за намаляне обема на кода и за промяна на фона.	**draw_bg**
9	**Събиране на боклук и почистване на парка** Учениците ще научат как да използват променливи и как да копират блок с код или дори цял спрайт.	**clean_bg**
10	**Нахранете котките** Учениците ще бъдат въведени в концепцията за присвояване на случайно число на няколко променливи в цикъл, както и каква е разликата при използването им вътре и извън цикъл. Те също така ще научат как за да	**cats_bg**

	зададат, тестват и изброят правилно въведеното от играча.	
11	**Познайте броя на котките в приюта** Учениците ще бъдат запознати с цикъл "repeat until" и как да зададат условието, при което спира играта. Те също така ще се научат как да използват променливи в различни ситуации: за задаване на произволна стойност, като брояч или като стойност въведена от играча.	**shelter_bg**

Scenari in GRECO

N.	TITOLO	CODICE
1	**Εισαγωγή στο Snap!** Ο μαθητής προσθέτει ένα νέο στοιχείο, προσθέτει ένα κοστούμι στο στοιχείο, επεξεργάζεται το κοστούμι και διαγράφει ένα από αυτά. Ο μαθητής δημιουργεί ένα νέο φόντο στη σκηνή, το επεξεργάζεται και διαγράφει τα ανεπιθύμητα.	--
2	**Ανακαλύψτε το Snap! : μετακινήσε μία εικόνα** Βοήθησε τους μαθητές να ανακαλύψουν την πλατφόρμα προγραμματισμού Snap! και να κωδικοποιήσουν την πρώτη τους εικόνα ώστε να κινέιται και να μιλάει.	**SampleCourse1Greek**
3	**Ας μετακινηθούμε γύρω από τη σκηνή** Ο μαθητής μαθαίνει πώς να μετακινεί ένα στοιχείο σε κατεύθυνση x και y στη σκηνή, δημιουργεί ένα εύκολο πρόγραμμα για την επίλυση των καθηκόντων του, μαθαίνει πώς να γυρίζει το στοιχείο σε διαφορετική κατεύθυνση.	--
4	**Αλλαγή κοστουμιών και στροφή** Ο μαθητής μαθαίνει πώς να αλλάζει το κοστούμι του στοιχείου για να κάνει ένα κινούμενο σχέδιο αλλάζοντας μεταξύ διαφορετικών τύπων περιστροφής.	--
5	**Ήχοι φάρμας** Οι μαθητές μαθαίνουν πώς να προγραμματίζουν ένα απλό παιχνίδι στο οποίο ο παίκτης μπορεί να αναγνωρίζει τους ήχους των ζώων πατώντας συγκεκριμένα κουμπιά.	--
6	**Καλοκαιρινές διακοπές του Χαμαιλέοντα** Προγραμματίστε ένα απλό παιχνίδι στο οποίο ένα αντικείμενο θα αλλάξει το κοστούμι του με βάση το χρώμα του φόντου	**SampleCourse2Greek**
7	**Βοηθώντας τον Πρίγκιπα και την Πριγκίπισσα να βρουν τα ζώα τους** ο κορίτσι πρέπει να βοηθήσει την Πριγκίπισσα να βρει τη γάτα της και τον Πρίγκιπα να βρει τον σκύλο του, αποφεύγοντας τα υπόλοιπα ζώα που μπορεί να συναντήσουν κατά τη διάρκεια της πορείας τους.	--
8	**Ζωγραφίζοντας με κιμωλία** Οι μαθητές θα μάθουν να ζωγραφίζουν διαφορετικά σχήματα με κώδικα. Θα μάθουν να χρησιμοποιούν βρόχους επανάληψης για να συντομεύσουν τον κώδικα και να αλλάξουν φόντο.	--
9	**Μαζεύοντας σκουπίδια και καθαρίζοντας το πάρκο**	--

	Οι μαθητές θα μάθουν πώς να χρησιμοποιούν μεταβλητές και πώς να αντιγράφουν ένα κομμάτι κώδικα ή ακόμα και ένα ολόκληρο στοιχείο.	
10	**Ταΐζοντας τις γάτες** Οι μαθητές θα μάθουν την έννοια της πολλαπλής ανάθεσης τυχαίας τιμής σε μεταβλητές μέσα σε ένα βρόχο και πώς αυτό είναι διαφορετικό αν το κάνουμε εκτός βρόχου. Θα μάθουν επίσης για τον τρόπο λήψης, δοκιμής και μέτρησης των σωστών εισόδων από τον παίκτη.	--
11	**Μαντέψτε τον αριθμό των γατιών στο καταφύγιο** Οι μαθητές θα μάθουν το βρόχο «επανάληψη έως» και πώς να ρυθμίσουν τη συνθήκη ώστε να παρακολουθούν τη συνθήκη που σταματά το παιχνίδι. Θα μάθουν επίσης πώς να χρησιμοποιούν τη μεταβλητή σε διαφορετικές καταστάσεις: για να ορίσουν μια τυχαία τιμή, ως μετρητή ή για να πάρουν είσοδο από τον παίκτη.	--

Scenari in PORTOGHESE

N.	TITOLO	CODICE
1	**Introdução à interface Snap!** Os alunos adicionam um novo sprite, adicionam um traje ao sprite, editam o traje e excluem um deles. O aluno cria um novo plano de fundo para o palco, edita-o e exclui os indesejados**.**	**starting_PT**
2	**Descubra o Snap! : mover um Sprite** Os alunos descobrem a interface do Snap! e codificam o seu primeiro sprite para que ele se mova e fale.	**dispubeng_PT**
3	**Movendo-se pelo palco** Os alunos aprendem como mover o sprite nas direções x e y do palco, codificam um programa fácil para resolver as tarefas dadas e aprendem a virar o sprite em uma direção diferente.	**monkey_PT**
4	**Mudando de traje e girando** Os alunos aprendem a mudar o traje do sprite para fazer uma animação e alternam entre os diferentes tipos de rotação.	**dancer_PT**
5	**Sons da quinta** Os alunos aprendem a programar um jogo simples no qual o jogador pode ouvir os sons dos animais pressionando certas teclas.	**farm_PT**
6	**Férias de verão do camaleão** Os alunos programam um jogo simples em que um objeto muda de traje de acordo com a cor do fundo	**chameleoneng_PT**
7	**Ajudando o príncipe e a princesa a encontrar seus animais** Uma menina tem que ajudar a Princesa a encontrar seu gato e o Príncipe a encontrar seu cachorro, evitando que os animais se cruzem durante o caminho.	**finding_PT**
8	**Desenho com giz** Os alunos aprendem a desenhar formas diferentes com um código. Aprendem a usar ciclos para encurtar o código e alterar um plano de fundo.	**chalk_PT**
9	**Recolher o lixo e limpar o parque** Os alunos aprendem a usar variáveis, a duplicar um bloco de código e um sprite.	**cleaning_PT**
10	**Alimentando os gatos** Os alunos aprendem o conceito de atribuição de valores aleatórios a múltiplas variáveis dentro de um loop e como é diferente de quando o fazemos fora de um loop. Eles também aprendem a obter, testar e contar as entradas corretas do jogador.	**feeding_PT**
11	**Adivinhando o número de gatos em um abrigo**	**shelter_PT**

	Os alunos aprendem o ciclo "repetir até" e como definir a condição para rastrear a condição que interrompe o jogo. Aprendem ainda a usar variáveis em diferentes situações: definir um valor aleatório, como um contador ou para obter dados dos jogadores.	

Scenari in TURCO

N.	TITOLO	CODICE
1	**Snap! ara yüzünü tanıyalım** Öğrenciler yeni bir karakter (sprite/kukla) ekler, karaktere bir kostüm ekler, kostümü düzenler ve bunlardan birini siler. Öğrenciler sahne için yeni bir arka plan oluşturur, düzenler ve istenmeyenleri siler.	**başla**
2	**Snap!'i keşfet - Karakteri hareket ettirme** Öğrencilerin Snap! arayüzünü keşfetmelerine ve ilk kodlarını oluşturarak hareket eden ve konuşan bir karakter oluşturmalarına yardımcı olun.	**keşfet**
3	**Sahnede hareket etme** Öğrenciler sahnede karakteri x ve y yönünde hareket ettirmeyi öğrenir, verilen görevleri çözmek için basit bir program programlar, karakteri farklı yöne çevirmeyi öğrenir.	**hareket**
4	**Kostüm değiştirme ve dönüşler** Öğrenci, farklı rotasyon türleri arasında geçiş yaparak bir animasyon yapmak için karakerin kostümünü nasıl değiştireceğini öğrenir.	**dans**
5	**Çiftlikteki sesler** Öğrenciler, oyuncunun belirli tuşlara basarak hayvanların seslerini tanıyabileceği basit bir oyun programlamayı öğrenirler.	**çiftlik**
6	**Bukalemenun yaz tatili** Bir nesnenin kostümünü arka planın rengine göre değiştireceği basit bir oyun programlayın	**bukalemun**
7	**Prens ve Prensese evcil hayvanlarını bulmaları için yardım et** Kız, seyahatleri sırasında hayvanların birbirleriyle karşılaşmasını engelleyerek Prenses'in kedisini bulmasına ve Prens'in köpeğini bulmasına yardım etmelidir.	**prenses**
8	**Tebeşir ile çizim yapmak** Öğrencilere bir kodla farklı şekiller çizme tanıtılacaktır. Kodu kısaltmak ve arka planı değiştirmek için döngü tekrarını kullanmayı öğrenecekler.	**çizim**
9	**Çöpleri toplamak ve parkı temizlemek** Öğrenciler değişkenleri nasıl kullanacaklarını ve bir kod bloğunu veya hatta bütün bir hareketli grafiği nasıl kopyalayacaklarını öğrenecekler.	**çöpler**
10	**Kedileri besleme** Öğrencilere, bir döngü içinde çok değişkenli rastgele değer	**kediler**

	atama kavramı ve bir döngü dışında yaptığımızdan nasıl farklı olduğu anlatılacaktır. Ayrıca doğru oyuncu girdilerinin nasıl alınacağını, test edileceğini ve sayılacağını da öğrenecekler.	
11	**Barınaktaki Kedi Sayısını Tahmin Etmek** Öğrencilere "----e kadar tekrar et" döngüsü ve oyunu durduran koşulu örtük olarak izlemek için koşulun nasıl ayarlanacağı tanıtılacaktır. Ayrıca değişkeni; rastgele bir değer belirlemek, sayaç olarak veya oyuncuların girdisini almak gibi farklı durumlarda nasıl kullnacaklarını da öğrencekler.	**barınak**

SCENARI AVANZATI DI APPRENDIMENTO

Scenari in INGLESE

N.	TITOLO	CODICE
12	**Catching healthy food** Students will learn how to randomly move for X steps and choose a position and also how to use variables and conditionals for preventing other event.	**food**
13	**Storytelling** Students will learn how to plan storytelling, how to use broadcast messages for synchronisation of the activities of sprites and stage changes.	**storytelling**
14	**Drawing** Student will learn to draw in Snap!, to change the colour and the pen thickness, and how to use variables and conditionals that cause a new event.	**drawing**
15	**Catch the mouse** Student will be introduced to the concept of multiple variable random value assignment. They will learn how to use the Operators/pick random[x]to[y] block.	**mouse**
16	**Buying food for a picnic** Students will learn how to work with variables: setting different starting values, using conditionals to compare variables' value, changing variables' value, using variables for counting (un)healthy food. In addition, they will repeat adding text, joining texts and if statement.	**picnic**
17	**Operations** Students will improve their knowledge on variables, random numbers, loops, broadcast.	**operation**
18	**Recycling** Students will improve their knowledge and will extend the game scenario with new objects, code and changing code with respect to new sprites. They will be trained in code refactoring.	**recycling**
19.1	**Play a piano_1** Students will learn about melodies coding and playing and will improve their knowledge about variables, loops, conditional, broadcast and other events.	**music**
19.2	**Play a piano_2** Students will learn how to play music and change costume by clicking on a sprite.	**piano**
20	**Test** Students will improve their knowledge and will extend the	**test**

	game scenario with new background, code and changing code with respect to new stages.	
21	**Simplified PACMAN game** Students will review their knowledge about movement inside a labyrinth with the use of sense color block. They will learn the concept of cloning the object with position restrictions and how to create a very simple non player character with its own random movement.	**pacman**

Scenari in SLOVENIO

N.	TITOLO	CODICE
12	**Lovljenje zdrave hrane** Učenec se nauči naključnega premikanja za X korakov in naključne izbire pozicije ter uporabo spremenljivk in pogojnih stavkov za preprečevanje drugih dogodkov.	**lovljenje_hrane_slo**
13	**Sestavi zgodbo** Učenec se nauči načrtovanja pripovedovanja zgodb, uporabe pošiljanja obvestil za sinhronizacijo dejavnosti likov ter zamenjave ozadij.	**zgodba_slo**
14	**Onesnažen zrak** Učenec se nauči risanja, spreminjanja barve, debeline svinčnika ter uporabe spremenljivke in pogojev za izvedbo drugega dogodka.	**zrak_slo**
15	**Ulovi miš** Učenec se seznani s konceptom dodeljevanja naključnih vrednosti in se nauči, kako uporabiti operator naključno število od_[x]_do_[y].	**mis_slo**
16	**Kupovanje hrane za piknik**	**piknik_slo**
17	**Računanje** Učenec izboljša svoje znanje o spremenljivkah, naključnih številkah, zankah ter pošiljanju obvestil.	**racunanje_slo**
18	**Recikliranje**	**recikliranje_slo**
19.1	**Zaigraj na klavir 1** Učenec se nauči sestavljanja melodij ter izboljša svoje znanje o spremenljivkah, zankah, pogojih, pošiljanju obvestil ter drugih dogodkov.	**ples_slo**
19.2	**Zaigraj na klavir 2** Učenec se nauči predvajati glasbo in liku spremeniti obleko s klikom nanj.	**klavir_slo**
20	**Test** Učenec izboljša svoje znanje in igro razširi z novimi ozadji, doda in spremeni kodo.	**test_slo**
21	**Enostavni Pacman** Učenec ponovi s pomočjo zaznavanja barve liku omejiti gibanje, spozna koncept kloniranja z omejitvami gibanja in se nauči ustvarjanja lika, ki se samostojno naključno premika po labirintu.	**pacman_slo**

Scenari in ITALIANO

N.	TITOLO	CODICE
12	**Raccogliere i cibi salutari** Gli studenti impareranno a muoversi casualmente per X passi, a scegliere una posizione, ad usare variabili e condizioni per prevenire altri eventi.	**cibo**
13	**Alice nel Paese delle Meraviglie** Gli studenti impareranno a pianificare una storia, ad utilizzare i messaggi inviati e ricevuti per la sincronizzazione delle attività degli Sprite e per i cambiamenti di scena.	**alice**
14	**Disegno** Lo studente imparerà a disegnare con Snap!, a cambiare il colore e lo spessore della penna e ad usare le variabili e le condizioni che determinano nuovi eventi.	**disegno**
15	**Alla caccia di un topolino!** Gli studenti comprenderanno il concetto di assegnazione di più valori casuali variabili. Impareranno come usare gli Operatori / scegliere il blocco casuale da [x] a [y].	**topolino**
16	**Acquistare cibo per un picnic** Gli studenti impareranno a lavorare con le variabili: impostare diversi valori iniziali, usare i condizionali per confrontare il valore delle variabili, cambiare il valore delle variabili, usare le variabili per contare alimenti sani (e non). Inoltre, aggiungeranno il testo, useranno l'unione di testi e la condizione "if".	**picnic_1**
17	**Operazioni** **Gli studenti miglioreranno le loro conoscenze su variabili, numeri casuali, cicli e broadcast.**	**operazioni**
18	**Raccolta differenziata** Gli studenti miglioreranno le loro conoscenze per estendere lo scenario di gioco con nuovi oggetti, codice e cambiando codice rispetto ai nuovi Sprite.	**riciclo**
19.1	**Suonare il piano_1** Gli studenti impareranno a codificare le melodie e miglioreranno le loro conoscenze su variabili, loop, condizionale, trasmissione e altri eventi.	**music_1**
19.2	**Suonare il piano_2** Gli studenti impareranno a suonare e a cambiare costume facendo clic su uno Sprite	**piano_2**
20	**Test** Gli studenti miglioreranno le loro conoscenze ed estenderanno lo scenario del gioco con nuovi background, codici e sue modifiche.	**test_1**

21	**PACMAN** Gli studenti rivedranno le loro conoscenze sul movimento all'interno di un labirinto con l'uso dei blocchi dei sensori del colore. Impareranno il concetto di clonazione dell'oggetto con restrizioni di posizione e a creare un personaggio "non giocatore".	**pacman_1**

Scenari in CROATO

N.	TITOLO	CODICE
12	**Hvatanje zdrave hrane** Učenici će naučiti kako nasumično pomicati za X koraka i odabrati položaj i također kako koristiti varijable i uvjete za sprječavanje drugih događaja.	**hvatanje**
13	**Pričam ti priču** Učenici će naučiti kako ispričati priču, kako koristiti poruke i kako promijeniti pozadinu pozornice.	**prica**
14	**Crtanje** Učenici će naučiti kako se crta pomoću Snap!-a, promijeniti boju i debljinu olovke te kako koristiti varijable i uvjete koji uzrokuju novi događaj.	**crtanje**
15	**Uhvati miša** Učenik će se upoznati s konceptom dodjeljivanja više varijabli slučajnim vrijednostima. Naučit će kako koristiti blok Operatori / slučajni broj od [x] do [y].	**miš**
16	**Kupnja hrane za piknik** Učenici će naučiti kako raditi sa varijablama: postavljanje različitih početnih vrijednosti, korištenjem uvjeta za usporedbu vrijednosti varijabli, promjenom vrijednosti varijabli, korištenjem varijabli za brojanje (ne) zdrave hrane. Osim toga, ponovit će dodavanje i spajanje teksta te naredbu ako.	**piknik**
17	**Operacije** Učenici će poboljšati svoje znanje o varijablama, slučajnim brojevima, petljama, emitiranju.	**operacije**
18	**Recikliranje** Učenici će poboljšati prethodno stečeno znanje te će proširiti scenarij igre sa novim objektima, kodom i promjenom koda s obzirom na nove objekte. Učenici će moći restrukturirati kod.	**recikliranje**
19.1	**Sviranje klavira_1** Studenti će upoznati kodiranje i sviranje melodija te će poboljšati svoje prethodno stečeno znanje o varijablama, petlji, uvjetnim, emitiranim i ostalim događajima.	**glazba**
19.2	**Sviranje klavira_2** Učenici će naučiti svirati glazbu i mijenjati kostim klikom na sprite.	**klavir**
20	**Test** Učenici će poboljšati svoje znanje i proširiti scenarij igre s novom pozadinom, kodom i promjenom koda u odnosu na nove pozornice.	**test_cro**

Scenari in BULGARO

N.	TITOLO	CODICE
12	**Подбор на здравословна храна** Учениците ще се научат как да се придвижат на случаен принцип с X стъпки и да изберат позиция, както и как да използват променливи и условности за предотвратяване на друго събитие.	**food_bg**
13	**Разказвач** Учениците ще научат как да планират разказването на истории, как да използват излъчвани съобщения за синхронизиране на дейностите на спрайтове и промени на сцената.	**story_bg**
14	**Рисуване** Ученикът ще се научи да рисува в Snap !, да променя цвета и дебелината на писалката и да използва променливи и условия, които причиняват ново събитие.	**drawing_bg**
15	**Хванете мишката** Учениците ще бъде запознати с концепцията за присвояване на случвайни стойности на променливи. Ще се научат как да използват блока Operators/pick random[x]to[y] block.	**mouse_bg**
16	**Купуване на храна за пикник** Учениците ще се научат как да работят с променливи: задаване на различни начални стойности, използване на условия за сравняване на стойностите на променливите, промяна на стойностите на променливите, използване на променливи за броене на (не) здравословна храна. В допълнение, те ще разучат добавяне на текст, съединяване на текстове и if оператор.	**picnic_bg**
17	**Операции** Учениците ще подобрят знанията си за променливи, случайни числа, цикли, Broadcast.	**oper_bg**
18	**Рециклиране** Учениците ще подобрят знанията си и ще разширят сценария на играта с нови обекти, код и променлив код по отношение на новите спрайтове. Те ще бъдат обучени как да редактират кодове.	**recycling_bg**
19.1	**Посвири на пиано Версия 1** Учениците ще се запознаят с кодирането и свирането на мелодии и ще подобрят знанията си за променливи, цикли, условия, излъчване и други събития.	**music_bg**
19.2	**Посвири на пиано Версия 2**	**piano_bg**

	Учениците ще се научат как да свирят музика и да сменят костюма, като кликнат върху спрайт.	
20	**Тест** Учениците ще подобрят знанията си и ще разширят сценария на играта с нов фон, код и променлив код по отношение на новите сцени.	**test_bg**
21	**Опростена игра PACMAN** Учениците ще приложат своите знания за движение в лабиринт с помощта на сензорен цветен блок. Те ще научат концепцията за клониране на обект с ограничения за позицията и как да създадат много прост персонаж, който не е играч, с опция за произволно движение.	**pacman_bg**

Scenari in GRECO

N.	TITOLO	CODICE
12	**Πιάνοντας υγιεινά τρόφιμα** Οι μαθητές θα μάθουν πώς να κινούνται για Χ τυχαία βήματα και να επιλέγουν μια θέση και επίσης πώς να χρησιμοποιούν μεταβλητές και συνθήκες για την πρόληψη άλλου συμβάντος.	--
13	**Διήγηση ιστορίας** Οι μαθητές θα μάθουν πώς να σχεδιάζουν την αφήγηση μιας ιστορίας, πώς να χρησιμοποιούν τα μηνύματα μετάδοσης για συγχρονισμό των δραστηριοτήτων των στοιχείων με τις αλλαγές στη σκηνή.	--

Scenari in PORTOGHESE

N.	TITOLO	CODICE
12	**Comendo comida saudável** Os alunos aprendem a mover-se aleatoriamente X passos e a escolher uma posição. Aprendem a usar variáveis e instruções condicionais para prevenir um evento.	**food_PT**
13	**Narrativa** Os alunos aprendem a planejar a narração de histórias e como usar mensagens para sincronização das atividades dos sprites e mudanças de palco.	**storytelling_PT**
14	**Desenhando** Os alunos aprendem a desenhar no Snap!, a alterar a cor e a espessura da caneta e a usar variáveis e condicionais que geram um novo evento.	**drawing_PT**
15	**Apanhar o rato** Os alunos aprendem o conceito de atribuição de valores aleatórios de múltiplas variáveis. Aprendem ainda como usar o bloco de operadores de escolha aleatória [x] a [y].	**mouse_PT**
16	**Comprando comida para um piquenique** Os alunos aprendem a trabalhar com variáveis: valores iniciais, operadores condicionais para comparar o valor das variáveis, alterar o valor das variáveis, usar variáveis para contar alimentos saudáveis. Além disso, eles vão adicionar e juntar textos.	**picnic_PT**
17	**Operações** Os alunos vão melhorar os seus conhecimentos sobre variáveis, números aleatórios e ciclos.	**operation_PT**
18	**Reciclagem** Os alunos vão aumentar um cenário do jogo com novos objetos, código e código em relação a novos sprites. Aprendem ainda sobre refatoração de código.	**recycling_PT**
19.1	**Tocar piano_1** Os alunos aprendem sobre codificação e reprodução de melodias e melhoram os seus conhecimentos sobre variáveis, ciclos, instruções condicionais e outros eventos.	**music_PT**
19.2	**Tocar piano_2** Os alunos aprendem a tocar música e a mudar de traje clicando num sprite.	**piano_PT**
20	**Teste** Os alunos irão melhorar os seus conhecimentos alargando o cenário do jogo com um novo fundo, código e código mutável em relação a novos palcos.	**test_PT**

21	**Jogo PACMAN simplificado** Os alunos irão rever os seus conhecimentos sobre o movimento dentro de um labirinto com o uso do bloco de reconhecimento de cores. Aprendem o conceito de clonar o objeto com restrições de posição e como criar um personagem não-jogador muito simples com o seu próprio movimento aleatório.	**pacman_PT**

Scenari in TURCO

N.	TITOLO	CODICE
12	**Sağlıklı yiyeceği yakalamak** Öğrenciler, belirli bir adım atmak için rastgele hareket etmeyi, bir konum seçmeyi ve ayrıca diğer olayları önlemek için değişkenleri ve koşulluları nasıl kullanacaklarını öğreneceklerdir.	**yemek**
13	**Hikaye Anlatma** Öğrenciler hikaye anlatımını nasıl planlayacaklarını, karakterin aktiviteleri ile sahne değişikliklerinin senkronizasyonu için yayın mesajlarının nasıl kullanılacağını öğrenecekler	**hikaye**
14	**İklimi İyileştirmek-Çizim Yapma** Öğrenci, Snap!'te çizim yapmayı, rengi ve kalem kalınlığını değiştirmeyi ve yeni bir olaya neden olan değişkenler ile koşullu ifadeleri nasıl kullanacağını öğrenecek.	**iklim**
15	**Fareyi yakalamak** Öğrencilere çok değişkenli rastgele değer atama kavramı tanıtılacaktır. Operatörleri nasıl kullanacaklarını ve rastgele [x] ila [y] bloğu seçmeyi öğrenecekler.	**fare**
16	**Piknik için yemek almak** Öğrenciler değişkenlerle nasıl çalışılacağını öğrenecekler: farklı başlangıç değerleri belirleme, değişkenlerin değerini karşılaştırmak için koşullu ifadeler kullanma, değişkenlerin değerini değiştirme, sağlıklı yiyecekleri saymak (olmayan) için değişkenler kullanma. Ek olarak, metin eklemeyi, metinleri birleştirmeyi ve Eğer (if) ifadesini tekrarlayacaklar.	**piknik**
17	**İşlemler** Öğrenciler değişkenler, rastgele sayılar, döngüler, yayın hakkındaki bilgilerini geliştireceklerdir.	**işlem**
18	**Geri Dönüşüm** Öğrenciler bilgilerini geliştirecek ve oyun senaryosunu yeni karakterlere göre yeni nesneler, kodlar ve değişen kodlarla genişletecekler. Kod yeniden düzenleme konusunda eğitilecekler	**dönüşüm**
19.1	**Piyano Çalmak** Öğrenciler melodileri kodlama ve çalma hakkında bilgi edinecek ve değişkenler, döngüler, koşullu, yayın ve diğer	**Piyano1**

	olaylar hakkındaki bilgilerini geliştireceklerdir.	
19.2	**Piyano Çalmak** Öğrenciler bir karaktere tıklayarak müzik çalmayı ve kostümü değiştirmeyi öğrenecekler.	**piyano2**
20	**Test- Kostümleri değiştirmek ve Dönüşler** Farklı döndürme türleri arasında geçiş yaparak bir animasyon yapmak için karakterin kostümünü nasıl değiştireceğinizi öğrenin	**test**
21	**Basitleştirilmiş PACMAN oyunu** Öğrenciler duyu renk bloğu kullanarak bir labirent içindeki hareketin nasıl kodlanması gerektiği hakkındaki bilgilerini gözden geçireceklerdir. Nesneyi konum kısıtlamaları ile rastgele hareket edecek şekilde basitçe kodlayacakar ve karakterleri klonlamayı öğrenecekler.	**pacman**

BIBLIOGRAFIA

A project of the K-12 Initiative at Stanford University. (2009, 05 06). *Taking Design Thinking to School: Approaches to Integrating the Design Process in K-12 Teaching and Learning.* Retrieved 09 10, 2020, from Stanford Graduate School of Education: https://web.stanford.edu/dept/SUSE/taking-design/presentations/Taking-design-to-school.pdf

Alserri, S., Zin, N., & Wook, T. (2018). Alserri, S. A., Zin, N. A. M., & Wook, T. S. M. T.Gender-based Engagement Model for Serious Games. , , 8(4). . *International Journal on Advanced Science, Engineering and Information Technology, 8*(4), 1350-1357.

Baptista, R., & Vaz de Carvalho, C. (2010). Role Play Gaming and Learning. *Learning Technology, 12*(1).

Combéfis, S., Beresnevičius, G., & Dagiene, V. (2016). Learning Programming through Games and Contests: Overview, Characterisation and Discussion. *Olympiads in Informatics, 10*.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: A 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges, 15*(5), 107-116.

Doorley, S., Holcomb, S., Klebahn, P., Segovia, K., & Utley, J. (2018). *Design Thinking Bootleg.* Retrieved from https://static1.squarespace.com/static/57c6b79629687fde090a0fdd/t/5b19b2f2aa4a99e99b26b6bb/1528410876119/dschool_bootleg_deck_2018_final_sm+%282%29.pdf

Eurostat. (2020, 09 25). *ICT specialists in employment.* Retrieved 11 12, 2020, from eurostat Statistics Explined: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=ICT_specialists_in_employment#Number_of_ICT_specialists

Feldgen, M., & Clua, O. (2004). Games as a motivation for freshman students to learn programming. *34th ASEE/IEEE Frontiers in Education Conference*, (pp. 1079–1084).

Frankovic, I., Hoic-Bozic , N., & Nacinovic-Prskalo, L. (2018). Serious Games for Learning Programming Concepts. *8th Conference edition The Future of Education.* Florence, Italy.

Garris , R., Ahlers, R., & Driskell, J. E. (2002). Games, motivation and learning. Simulation & Gaming. *An Interdisciplinary Journal of Theory, Practice and Research, 33*(4), 43-56.

Gee, J. P. (2003). *What Video Games Have to Teach Us About Learning and Literacy.* New York: Palgrave/Macmillan.

Gee, J. P. (2007). Are video games good for learning? *Curriculum & Leadership Journal, 5*(1).

Geist, E. (2016). Robots, Programming and Coding, Oh My! *Childhood Education, 92*(4), 298-304. doi:10.1080/00094056.2016.1208008

Georgieva, R., & Tuparova, D. (2019). Educatiomnal Computer Game-Based Environments for Teaching Programming and Developmnt of an Algorithmic Thinking - Comparative AnalysisITHMIC THINKING – COMPARATIVE ANALYSIS. *Mathematics and Education in Mathematics, Forty-eighth Spring Conferenceof the Union of Bulgarian Mathematicians* (pp. 150-156). Union of Bulgarian Mathematicians. Retrieved from http://www.math.bas.bg/smb/2019_PK/tom/pdf/150-156.pdf

Georgieva, R., & Tuparova, D. (2020). Design Thinking - helps in school eucation. *Anniversary International Scientific Conference "Synergeticsand ReflectioninMathematics Education"*, (pp. 341-347). Pamporovo.

Grivokostopoulou, F., Perikos, I., & Hatzilygeroudis, I. (2016). An Educational Game for Teaching Search Algorithms. *Proceedings of the 8th International Conference on Computer Supported Education (CSEDU 2016)* (pp. 129–136). Setubal. Portugal: SCITEPRESS - Science and Technology Publications, Lda.

Gross, B. (2003). The impact of videogames in education. 8(7). *First Monday, 8*(7).

Hijon-Neira, R., Velazquez-Iturbide, A., Pizarro-Romero, C., & Carrico, C. (2015). Serious games for motivating into programming. *Frontiers in Education Confere.*

IDEO Design thinking. (2008, 09 7). *Definitions of design thinking*. Retrieved 11 20, 2020, from https://designthinking.ideo.com/blog/definitions-of-design-thinking

Innovation Starter. (2015, 06 04). *What is design thinking?* Retrieved 11 20, 2020, from http://innovationstarterbox.bg/resources/kakvo-e-dizain-mislene/

Jemmali , C. (2016). May's Journey: A serious game to teach middle and high school girls programming. Worcester Polytechnic Institute. Retrieved from https://digitalcommons.wpi.edu/etd-theses/455

Johnson, C., & all, a. (2016). Game Development for Computer Science Education. *the 2016 ITiCSE Working Group Reports.*

Johnston, R. T., & de Felix, W. (1993). Learning from video games. *Computer in the Schools* , 199-233.

Kafai, Y. (1995). Making game artifacts to facilitate rich and meaningful learning. *Annual Meeting of the American Educational Research Association*, (pp. 1–20). San Francisco.

Karakehayova, M. (2019). Opportunities for Using Design Thinking in School Education. *Education and Development, 3*.

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. *Procedia - Social and Behavioral Sciences, 47*, 1991-1999.

Kelleher , C., Pausch, R., & Kiesler, S. (2007). Storytelling Alice motivates middle school girls to learn computer programming. *Proc. CHI 2007.*, (pp. 1455-1464).

Luka, I. (2014). Design thinking in pedagogy. 2014, 5,. *The Journal of Education, Culture, and Society, 5*, 63-74.

Malliarakis, C., Satratzemi, M., & Xinogalos, S. (2014). CMX: Implementing an MMORPG for learning programming, 8th European Conference on Games Based Learning. *8th European Conference on Games Based Learning - ECGBL 2014.* Berlin.

Malone, T. (1981). What makes computer games fun? *Byte, 6*(12), 258-277.

Malone, T. W. (1981b). Toward a theory of intrinsically motivating instruction. *Cognitive Science, 4*.

Mathrani, A., Christian, S., & Ponder-Sutton, A. (2016). PlayIT: Game Based Learning Approach for Teaching Programming Concepts. *Educational Technology & Society, 19*(2), 5-17.

Meerbaum-Salant , O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with Scratch. *Computer Science Education, 23*(3), 239-264. doi:10.1080/08993408.2013.832022

Michael, D. R., & Chen, S. (2006). *Serious Games: Games That Educate, Train, and Inform.* Boston: Course Technology PTR.

Miljanovic, M., & Bradbury, J. (2016). Robot on!: a serious game for improving programming comprehension. *Proceedings of the 5th International Workshop on Games and Software Engineering.* Austin, Texas, USA.

Miljanovic, M., & Bradbury, J. (2018). A Review of Serious Games for Programming. *4th Joint International Conference on Serious Games.* Darmstadt, Germany.

Naiman, L. (2019, 06 10). *Design Thinking as a Strategy for Innovation*. Retrieved 11 21, 2020, from Creativity at work: https://www.creativityatwork.com/design-thinking-strategy-for-innovation/

Nančovska Šerbec, I., Kaučič, B., & Rugelj, J. (2008). Pair programming as a modern method of teaching computer science. *International journal on emerging technologies in learning, 3*, 45-49.

Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A., & Lahmine, S. (2015). Learning Basic Programming Concepts by Creating Games with Scratch Programming Environment. , 191, . *Procedia - Social and Behavioral Sciences, 191*, 1479-1482. doi:doi:10.1016/j.sbspro.2015.04.224

Paavola, S., & Hakkarainen, K. (2005). The Knowledge Creation Metaphor – An Emergent Epistemological Approach to Learning.14. *Sci Educ, 14*, 535-546.

Phan, M., Jardina, J., Hoyle, S., & Chaparro, B. (2012). Examining the role of gender in video game usage, preference, and behavior. *Human Factors and Ergonomics Society 51*, (pp. 1496–1500.).

Pivec, M., & Kearney, P. (2007). Games for Learning and Learning from Games. *Informatica, 31*, 419-423.

Pivec, M., Koubek, A., & Dondi, C. (2004). *Guidelines for Game-Based Learning. Lengerich.* Pabst Science Publishers.

Prensky , M. (2001). *Digital Game-Based Learning.* New York: Mc-Graw-Hill.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond , E., Brennan , K., . . . Kafai, Y. (2009). Scratch: Programming for All. *Communication of the ACM, 52*, 60-67.

Rieber , L. P., Smith , L., & Noah, D. (1998). The value of serious play. *Educational Technology, 38*(6), 29-37.

Rugelj, J. (2016). Serious computer games design for active learning in teacher education. (C. Carvalho , P. Escudeiro, & A. Coelho, Eds.) *Serious games, interaction, and simulation : revised selected papers. Lecture notes of the institute for computer sciences, social informatics and telecommunications engineering, 161*, 94-102.

Rugelj, J., & Lapina, M. (2019). Game design based learning of programming. In J. Rugelj, & M. Lapina (Ed.), *Proceedings of the International Scientific Conference Innovative Approaches to the Application of Digital Technologies in Education and Research (SLET-2019), May 20-23, CEUR workshop proceedings. 2494.* Stavropol-Dombay, Russia: Aachen: CEUR-WS.

Shabanah, S., Chen, J., Wechsler, H., Carr, D., & Wegman, E. (2010). Designing Computer Games to Teach Algorithms. *Seventh International Conference on Information Technology: New Generations, ITNG 2010*, (pp. 1119-1126). Las Vegas, NV.

Shute, V. J., & Ke, F. (2012). Games, Learning, and Assessment. In D. Ifenthaler, D. Eseryel, & X. Ge, *Assessment in Game-Based Learning: Foundations, Innovations, and Perspective.* New York, USA: Springer.

Shute, V. J., Rieber, L. P., & Van Eck, R. (2012). Games ... and ... Learning. In R. A. Reiser, & J. V. Dempsey, *Trends and issues in instructional design and technology (3rd ed.).* (pp. 321-332). Boston: Pearson Education.

Sykes, E. (2007). Determining the Effectiveness of the 3D Alice Programming Environment at the Computer Science I Level. *Journal of Educational Computing Research, 36*(2), 223-244. doi:10.2190/J175-Q735-1345-270M

Tuparova, D., Nikolova, E., & Tuparova, E. (2020). Integrated game-based learning in an informatics secondary course: Is there a difference between girls' and boys' achievements? *CSEDU 2020 - Proceedings of the 12th International Conference on Computer Supported Education*, *1*, pp. 702-709. Retrieved 10 12, 2020, from https://www.scitepress.org/Link.aspx?doi=10.5220/0009818407020709

Tuparova, D., Tuparov, G., & Veleva, V. (2019 b). Girls' and boys' viewpoint on educational computer games. *European Conference on Games-based Learning*, (pp. 757–766).

Vahldick, A. (2017). Aperfeiçoamento das Competências de Resolução de Problemas na Aprendizagem Intodutória de Programação de Computadores usando um jogo sério digital. Faculdade de Ciências e Tecnologia da Universidade de Coimbra. Retrieved from http://hdl.handle.net/10316/79528

van Eck, R. (2006). Digital Game-Based Learning: It's Not Just the Digital Natives Who Are Restless. *EDUCAUSE Review, 41*(2).

Vermeulen, L., Van Looy, J., Courtois, C., & De Grove, F. (2011). Girls will be girls: a study into differences in game design preferences across gender and player types. *Under the mask: perspectives on the gamer conference*, (pp. 1-21).

Weintrop, D., & Wilensky, U. (2015). To Block or not to Block, That is the Question: Students' Perceptions of Blocks-based Programming. *Interaction Design and Children*, (pp. 199-208).

Whitton, N. (2009). *Learning with Digital Games: A Practical Guide to Engaging Students in Higher Education.* New York: Routledge.

Whitton, N., & Moseley, A. (2012). *Using Games to Enhance Learning and Teaching: A Beginner's Guide.* New York: Routledge.

Wolz, U., Barnes, T., & Bayliss, J. (2009). Girls do like playing and creating games. *ACM SIGCSE Bulletin, 41*(1), 199–200.

Wolz, U., Maloney, J., & Pulimood, S. (2008). 'Scratch' Your Way to Introductory CS. *SIGCSE Bulletin, 40*, 298-299.

Zapušek, M., & Rugelj, J. (2013). Learning programming with serious games. *EAI Endorsed Trans. on Game-Based Learning, 13*, 1-12.