risorse formative

ENRICO NARDELLI

la programmazione informatica per la scuola primaria

Breve guida introduttiva con "Programma il Futuro"

Programma il Futuro

EDIZIONI THEMIS

Enrico NARDELLI

La programmazione informatica per la scuola primaria

Breve guida introduttiva con

EDIZIONI THEMIS

Si ringrazia Francesco Lacchia per la collaborazione

ISBN 9788896069394

Quest'opera è distribuita gratuitamente da
Themis s.r.l.
via Veturia 44 – 00181 Roma
themiscrime.com/it/edizioni-themis

Indice

1 Premessa

Questo testo è indirizzato agli educatori (siano essi insegnanti di scuola, genitori o formatori di altro tipo) che vogliono imparare i primi elementi di base della programmazione informatica e guidare i loro alunni della scuola primaria in questo apprendimento. Il volume è utile per una prima formazione sull'argomento, che a seguito dell'approvazione della Legge n.159 del 20 dicembre 2019 deve far parte della preparazione di tutti gli insegnanti di scuola primaria. Si suggerisce anche il manuale operativo curato dall'autore e pubblicato dalla casa editrice Lisciani: "Coding e oltre: l'informatica nella scuola", in cui si discutono nel primo capitolo i concetti fondamentali dell'informatica e le motivazioni culturali per il suo insegnamento nella scuola. Il capitolo presenta inoltre la proposta organica di curriculum formulata dal CINI (Consorzio Interuniversitario Nazionale per l'Informatica), riportata integralmente nell'appendice del manuale e disponibile online.

Il percorso proposto in questo volume utilizza il materiale didattico interattivo realizzato negli Stati Uniti dall'organizzazione no-profit Code.org (http://code.org), che è stato adattato per l'Italia nel corso del progetto MIUR-CINI Programma il Futuro (http://programmailfuturo.it). In tal modo il docente ha subito a disposizione quanto serve per il processo di apprendimento, utilizzando un ambiente che dal punto di vista tecnologico è molto semplice e lineare, quindi facilmente fruibile. Il testo guida alla comprensione dei concetti fondamentali indicando nel materiale didattico disponibile le lezioni e gli esercizi più rilevanti e descrivendone in dettaglio princìpi di base e meccanismi di funzionamento. L'insegnante potrà poi riutilizzare lo stesso materiale per guidare i suoi alunni. Gli esercizi discussi sono dei link attivi che portano direttamente alla pagina del sito di Code.org contenente l'esercizio stesso. Diventa così molto facile seguire la spiegazione e sperimentare subito la comprensione di quanto presentato. Inoltre, tutti i riferimenti interni a pagine e paragrafi (così come le voci dell'indice) sono dei link attivi che rendono il testo navigabile.

Il testo può essere usato anche da quei genitori che, in assenza di esplicite iniziative sull'informatica nelle scuole dei loro figli, vogliono avvicinarli a questo importante ed affascinante materia ma sentono di non avere una preparazione adeguata. Il percorso culturale sulla programmazione informatica che viene qui sviluppato non richiede nessuna conoscenza pregressa ma solo voglia di apprendere e curiosità per l'argomento trattato e costituisce una piccola parte di tutta l'informatica, disciplina scientifica essenziale per comprendere il mondo digitale.

(questa pagina è intenzionalmente vuota)

2 Introduzione

La proposta di introduzione della programmazione informatica nella scuola presentata in questo volume è strutturata in 3 moduli di difficoltà crescente e pensati per l'utilizzo in sequenza. Sono possibili differenti punti di partenza, a seconda del livello iniziale di maturazione ed esperienza degli studenti. "Maturazione" indica il generale livello di crescita culturale e scolastica dello studente, mentre "esperienza" si riferisce all'aver svolto qualche lezione di introduzione alla programmazione, indicativamente per una decina di ore complessive di attività didattica.

Noi suggeriamo come **modulo di inizio del percorso** quanto indicato nella seguente tabella.

precedente esperienza / *classe*	NO	SI
1° 2° primaria	Modulo 1S	Modulo 1
3° 4° 5° primaria	Modulo 1 o 2	Modulo 2 o 3

Il Modulo 1S è una versione semplificata del Modulo 1 che si basa su materiale fruibile anche da studenti che stanno imparando a leggere.

Ogni modulo è suddiviso in una serie di lezioni pensate per essere trattate nell'ambito di un'unica sessione di lavoro, della durata variabile tra 60 e-120 minuti a seconda della specifica situazione della classe, anche da un punto di vista logistico.

Il requisito tecnologico indispensabile è quello di avere la connessione alla rete Internet. La situazione ideale in termini di dispositivi è avere un tablet o PC ogni 1-2 studenti, anche se è possibile, a prezzo di una minore efficacia didattica e di un allungamento dei tempi, lavorare con più di 2 studenti per ogni dispositivo. La disponibilità della LIM è utile sia per discutere più agevolmente con l'intera classe che nei casi di strumentazione carente.

Ogni modulo può quindi essere svolto in un intervallo temporale tra le 10 e le 20 ore (a seconda della specifica situazione della classe) e presenta alcuni dei concetti fondamentali della programmazione informatica, sequenziati in un ordine di difficoltà crescente.

2.1 L'ambiente di Code.org

Come detto nella premessa, tutti i moduli fanno riferimento al materiale didattico realizzato negli Stati Uniti da Code.org, e nel resto del volume, quando si fa riferimento a "corso" si intendono i corsi disponibili sul sito di Code.org, di cui si riportano per comodità gli indirizzi nel seguito:

- Corso 1 (https://studio.code.org/s/course1/lang/it)[1]
- Corso 2 (https://studio.code.org/s/course2/lang/it)
- Corso 3 (https://studio.code.org/s/course3/lang/it)
- Corso 4 (https://studio.code.org/s/course4/lang/it)

Le guide didattiche per gli insegnanti per questi corsi si trovano a partire da qua: https://programmailfuturo.it/come/primaria/vecchie-lezioni-tecnologiche/introduzione

Successivamente alla stesura di questo volume Code.org ha riorganizzato il materiale di questi corsi suddividendolo in 6 corsi identificati da una lettera da A ad F. Sul sito di Programma il Futuro si trovano le guide didattiche per gli insegnanti relative a questa nuova organizzazione a partire da questa pagina: https://programmailfuturo.it/come/primaria/introduzione

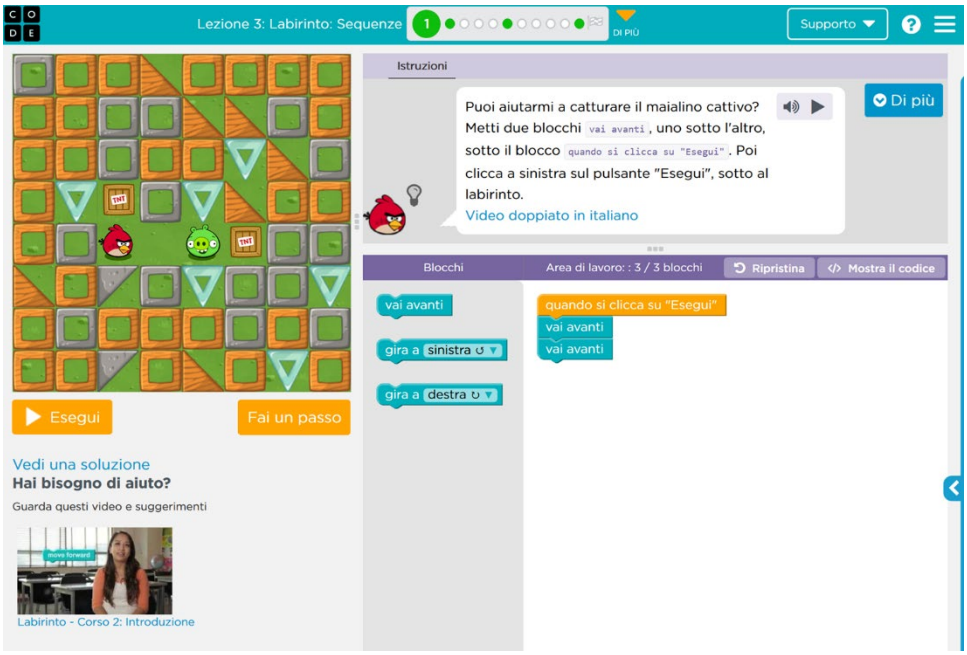
Il materiale didattico è di tipo interattivo, organizzato in "lezioni" costituite da un numero variabile (tipicamente, tra dieci e venti) di esercizi/problemi da risolvere. La corretta risoluzione degli esercizi indica l'acquisizione da parte dello studente dei temi affrontati nell'esercizio stesso.

Lo svolgimento delle lezioni richiede quindi che gli studenti possano usare strumentazione informatica (propria o della scuola). Come detto, la disponibilità di un PC o tablet ogni 1 o 2 studenti è la situazione migliore, ma il materiale può essere usato con successo anche se la classe è dotata solo di LIM collegata ad un PC connesso ad Internet.

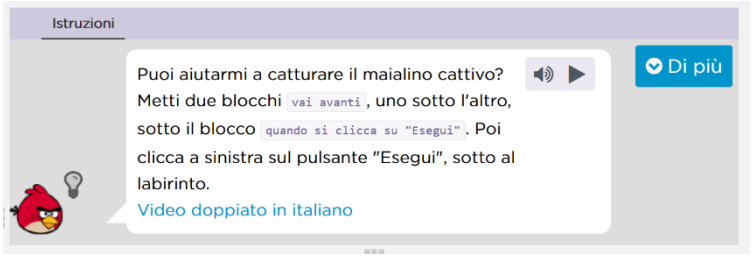
La soluzione di un esercizio consiste nella scrittura di un "programma informatico" che viene costruito assemblando nel modo opportuno dei "blocchetti", ognuno dei quali corrisponde ad una "istruzione" di tale programma.

[1] La parte finale dell'indirizzo /lang/it serve unicamente ad assicurare che la pagina venga visualizzata in italiano.

L'esercizio tipico si presenta come nella figura sottostante



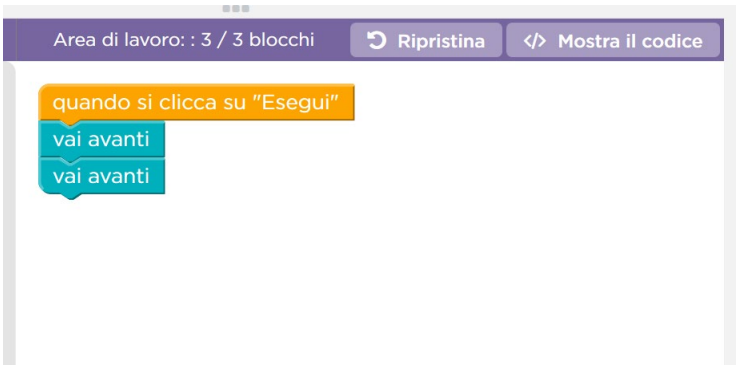
Il testo dell'esercizio viene visualizzato nella parte alta, da cui si accede anche alla versione doppiata in italiano del video di accompagnamento (per quelle lezioni che lo hanno). Cliccando sul pulsante che visualizza un altoparlante con vicino il triangolino si può attivare la lettura automatica (utile per i bambini che ancora non leggono). Il pulsante blu "Di più" a destra permette di visualizzare una versione più lunga del testo, con eventuali suggerimenti. In questa zona l'ambiente segnala anche eventuali errori del programma costruito o offre la possibilità di visualizzare suggerimenti (accessibili cliccando sulla lampadina).



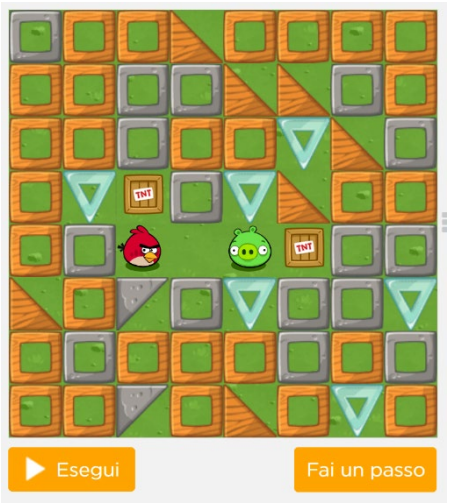
Le "istruzioni" sono visualizzate al centro mediante "blocchetti colorati" disponibili in una sezione "Blocchi" sotto il testo dell'esercizio/problema, che costituisce la "cassetta degli attrezzi" a disposizione per costruire la soluzione. Possono essere "prelevate" cliccandoci sopra col mouse, spostandole – mentre si tiene il pulsante del mouse premuto – nella zona a destra, che è l'area di lavoro in cui si costruisce il programma, e rilasciando il pulsante del mouse una volta che il blocchetto è nella posizione desiderata. Per impratichirsi con questa azione fondamentale di "tra-

scinare e rilasciare" (*drag and drop*, in inglese) si può svolgere la lezione 3 del corso 1 (vedere sezione 4.1 – Prerequisito: interazione con l'ambiente, a pag. 53).

La "area di lavoro" è nella parte destra. Disponendo in questa i blocchetti nella sequenza opportuna si costruisce il programma. I blocchetti trascinati e rilasciati dall'area centrale "Blocchi" vanno tutti piazzati sotto il blocco arancione "quando si clicca su Esegui" in modo che siano connessi tra loro (come si vede nella figura a fianco). L'area indica quando blocchetti sono stati piazzati e quanti ne servirebbero per risolvere l'esercizio in modo ottimale (si suggerisce di non preoccuparsi all'inizio di quest'ultima indicazione). Cliccando sul pulsante "Ripristina" si riporta la situazione dell'area di lavoro a quella iniziale. Cliccando su pulsante "Mostra il codice" si può vedere quali sono le effettive istruzioni che vengono eseguite dal reale linguaggio di programmazione (Javascript) su cui è basato l'ambiente.



A sinistra è visualizzato l'ambiente grafico con i personaggi che sono animati quando si esegue il programma cliccando sul pulsante "Esegui". Nei primi esercizi è disponibile anche un pulsante "Fai un passo" che permette di eseguire il programma un blocchetto alla volta, in modo da rendersi meglio conto dell'effetto di ognuno di essi. Al termine dell'esecuzione il pulsante "Esegui" in arancione diventa "Ricomincia" (in blu): cliccando su di esso si riporta la situazione dell'ambiente grafico nella situazione iniziale prima dell'esecuzione. Come detto, nel caso di errori, la segnalazione ed eventuali suggerimenti vengono visualizzati in alto, mentre se l'esercizio viene risolto correttamente si offre la possibilità di passare al successivo o di riprovarlo, segnalando contestualmente se sono stati usati più blocchetti di quanto strettamente necessario.



Sotto il pulsante "Esegui" è disponibile un link attivo "Vedi una soluzione" che è visualizzato solo quando si è entrati nell'ambiente di Code.org con un account da insegnanti. Gli studenti quindi non vedono tale link, mentre per il docente può essere un utile supporto. Ancora più in basso può essere presente un video di supporto/spiegazione: questa versione è quella in inglese con sottotitoli in italiano, mentre quella interamente doppiata in italiano è accessibile dall'area in alto con il testo dell'esercizio/problema.

Un suggerimento operativo che può abbreviare la predisposizione del codice in casi in cui una stessa istruzione viene ripetuta molte volte è, per chi ha dimestichezza manuale nell'uso della tastiera, quello di usare le scorciatoie da tastiera per fare "copia e incolla". In altre parole, dopo aver preso il blocco necessario dalla cassetta degli attrezzi, averlo spostato nell'area di lavoro, assicurandosi che il blocco sia selezionato (questa condizione è segnalata dal fatto che il bordo del blocco è evidenziato in giallo) attivare i comandi da tastiera per l'azione di "copia" e poi attivare i comandi da tastiera per l'azione di "incolla" per tante volte quante sono le copie da creare, che potranno poi essere sistemate come necessario. Nei sistemi operativi basati su Windows o Linux le scorciatoie da tastiera per i comandi "copia" e "incolla", sono rispettivamente CTRL+C (ovvero premere contemporaneamente il tasto con etichetta "CTRL" e quello con etichetta "C") e CTRL+V. Nei sistemi operativi dei dispositivi Apple le analoghe scorciatoie sono CMD+C e CMD+V.

2.2 Argomenti del percorso

Il piano generale del volume è il seguente.

Nel modulo 1 si introducono le prime astrazioni fondamentali della programmazione informatica: i concetti di **sequenza** di istruzioni e di **ripetizione** di istruzioni. Lo studente familiarizza con essi, imparando a identificare la corretta sequenza di istruzioni per risolvere il problema dato ed imparando quando usare il costrutto di ripetizione (detto anche **ciclo**). Esso viene introdotto in questo modulo nella forma più semplice in cui il numero delle ripetizioni è fissato a priori. La ripetizione permette anche di introdurre il concetto di **efficienza** del codice, ovvero di quantità di istruzioni usate per ottenere lo scopo desiderato. Viene infine presentata l'attività di **verifica della correttezza** dei programmi scritti con la conseguente correzione degli errori.

Per il modulo 1 è disponibile anche una versione semplificata, che presenta gli stessi concetti ma è fruibile anche dagli studenti che stanno imparando a leggere. In essa le istruzioni sono rappresentate da figure e il testo è ridotto al minimo indispensabile. Inoltre, i riferimenti direzionali sono espressi in modo assoluto rispetto all'esecutore, evitando quindi la necessità di doversi porre nei panni dell'esecutore per capire in che modo la specifica relativa del movimento agisca sull'esecutore stesso.

Dal modulo 2 in avanti, ogni modulo contiene sempre una ripetizione dei concetti appresi nei moduli precedenti, in modo da consentire una facile ripresa delle attività anche nel caso in cui tra lo svolgimento di un modulo e il successivo sia passato un anno scolastico.

Nel modulo 2, dopo aver ripreso i concetti di sequenza e di ripetizione introdotti nel modulo precedente, si estende l'attività di **verifica della correttezza**: in senso lato la correttezza è legata all'efficienza, nel senso che se vi sono istruzioni in più rispetto a quanto strettamente necessario il programma è parzialmente scorretto anche se raggiunge il suo scopo.

Si introduce poi il concetto di **istruzione condizionale**. Mentre nel modulo

1 tutti i programmi conducono all'esecuzione di una stessa sequenza lineare di istruzioni, in questo modulo gli studenti imparano a scrivere programmi che prendono decisioni e possono quindi eseguire differenti sequenze di istruzioni.

Nel modulo 3 si impara che i costrutti di ripetizione possono contenere al loro interno ulteriori costrutti di ripetizione, dando luogo al concetto di **ciclo annidato** e permettendo così di realizzare schemi operativi più complessi. Si introduce poi il concetto di **evento**, cioè del meccanismo che permette al programma di reagire durante la sua esecuzione ad azioni dell'utente o ad altri avvenimenti rilevanti.

Viene definito a tal scopo il costrutto **gestore dell'evento**, costituito da una sequenza di istruzioni che vengono eseguite solo quando l'evento si verifica, insieme ad una prima variante del costrutto di ripetizione, denominata **ripeti per sempre**, utile per la realizzazione di programmi interattivi, cioè basati su eventi.

3 Modulo 1: sequenza e ripetizione

Il Modulo 1 introduce i concetti di base della programmazione informatica, in modo che gli studenti sviluppino le loro capacità di risolvere problemi ed imparino a concentrarsi su problemi che richiedono il loro impegno. Lo svolgimento delle attività in collaborazione tra studenti è una possibilità aggiuntiva.

La discussione in questo capitolo assume che gli studenti abbiano già sviluppato una certa capacità di lettura, quale normalmente si consegue tra la seconda e la terza elementare. Il capitolo successivo presenta gli stessi concetti in riferimento a materiale didattico che può essere fruito anche da studenti che stanno imparando a leggere.

I concetti fondamentali coperti dal modulo 1 sono quello di **sequenza** di istruzioni e di **ripetizione** di istruzioni. Lo studente familiarizza con essi, imparando a identificare la corretta sequenza di istruzioni per risolvere il problema dato ed imparando quando usare il costrutto di ripetizione (detto anche **ciclo**). Esso viene introdotto in questo modulo nella forma più semplice in cui il numero delle ripetizioni è fissato a priori. Lo studente inoltre apprende che la ripetizione di istruzioni è essa stessa un'istruzione che può essere combinata in sequenza con altre istruzioni.

Si introduce infine lo studente all'attività di **verifica** della correttezza del programma esponendo una serie di programmi che non funzionano e chiedendo di correggere gli errori.

Qualora gli studenti non siano familiari con l'azione fondamentale di "trascinare e rilasciare" (*drag and drop*, in inglese) è necessario svolgere preliminarmente la lezione 3 del corso 1 (vedere sezione 4.1 – Prerequisito: interazione con l'ambiente, a pag. 53).

3.1 La sequenza

Per questo argomento si usano la lezione 7 del Corso 1 e la lezione 3 del Corso 2. Gli studenti sviluppano programmi[2] costituiti da **sequenze** di istruzioni che vanno opportunamente definite in modo che l'esecutore raggiunga il suo obiettivo dato il contesto in cui si trova. Individuare la corretta successione delle istruzioni è quindi il concetto fondamentale che acquisisce lo studente mediante queste attività.

Nella lezione 7 del Corso 1 l'esecutore è un'Ape che, muovendosi su un prato, deve raccogliere il nettare dai fiori e produrre miele nei favi.

Nella lezione 3 del Corso 2 l'esecutore è un uccellino del noto gioco Angry Birds[3] il cui obiettivo è raggiungere il maialino verde muovendosi in un labirinto.

[2] In casi come questi in cui l'esecutore è un computer si usa preferibilmente il termine "programma" al posto di quello di "procedura".

[3] Angry Birds ® e © 2009-2020 di Rovio Entertainment Ltd. Tutti i diritti riservati.

Come primissima lezione da svolgere dell'intero corso, si è scelta una lezione presa dal corso 1 di Code.org, che presenta un contesto operativo particolarmente semplice. Se l'insegnante ritiene la classe abbastanza matura, può senza problemi saltare questa lezione ed iniziare direttamente con la successiva Corso 2: Lezione 3 (Sequenza – Labirinto), paragrafo 3.1.2 a pag. 18.

3.1.1 Corso 1: Lezione 7 (Sequenza – Ape)

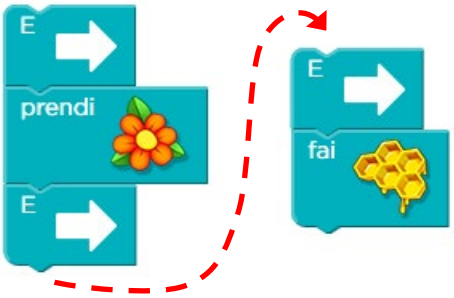
Il contesto della lezione è quello di un prato nel quale l'esecutore (un Ape) può spostarsi mediante azioni che la fanno muovere nella quattro direzioni corrispondenti ai quattro punti cardinali. Col suo movimento deve raggiungere fiori da cui raccogliere il nettare e favi nei quali produrre il miele. Le azioni a sua disposizione includono quindi, oltre a quelle di movimento, due azioni che realizzano le attività, appunto, di raccolta del nettare e produzione del miele.

L'**esercizio 1** ha lo scopo primario di presentare allo studente questa ambientazione, mostrando due sole delle possibili azioni di movimento e le due attività "prendi nettare" e "fai miele".

Osserviamo che queste ultime due azioni ottengono il loro risultato soltanto nel contesto opportuno. Più precisamente, quando l'Ape si trova su di un fiore che ha del nettare (le dosi disponibili sono segnalate da un numero maggiore di 0 in basso a destra nella casella che contiene il fiore) l'azione "prendi il nettare" ha il risultato di prelevare dal fiore una dose di nettare (decrementando quindi di 1 il relativo numero).

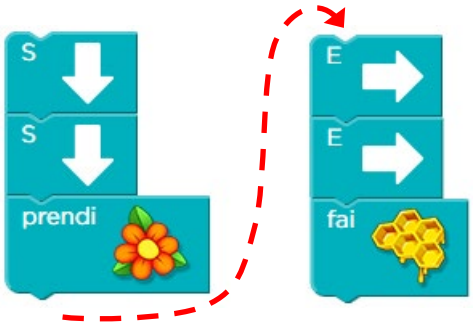
Analogamente, quando l'Ape si trova su di un favo che deve essere riempito di miele (le dosi da produrre sono segnalate come per il caso del nettare del fiore) l'azione "fai il miele" ha il risultato di depositare una dose di miele nel favo (decrementando quindi di 1 il relativo numero).

La sequenza corretta di azioni che deve eseguire l'ape in questo esercizio è mostrata qui a fianco (rappresentata su due colonne per motivi di spazio).



Osserviamo che in questa lezione, oltre al pulsante "Esegui" vi è il pulsante "Fai un passo" che permette di eseguire il programma "un blocco alla volta". È molto utile, soprattutto nei primi esercizi, usare questa modalità di esecuzione, detta anche "passo a passo", per comprendere con sicurezza cosa viene realizzato da ogni blocco del programma e se quanto sta accadendo è in linea con quanto ci si aspetta.

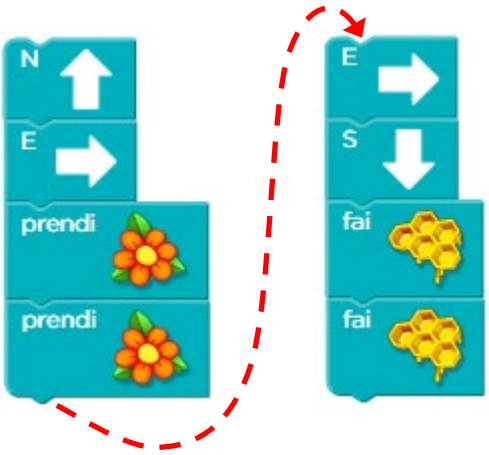
L'**esercizio 2** (sotto a sinistra) è concettualmente analogo al precedente – dal momento che si tratta sempre di mettere nella sequenza corretta le operazioni, ma la sua soluzione (sotto a destra) richiede di far cambiare direzione all'ape a metà del percorso.



Il successivo esercizio 3 è dello stesso tipo, e pertanto non lo trattiamo, mentre nell'**esercizio 4** (raffigurato a fianco) viene introdotta l'ulteriore variazione di più dosi di nettare da raccogliere e più dosi di miele da produrre (si notino i numeri 2 accanto al fiore e al favo). Osserviamo che ad un esame non approfondito dell'ambientazione può sfuggire il fatto che vi siano due dosi di nettare da prelevare e due dosi di miele da produrre. Oppure, si potrebbe assumere che il comando "prendi nettare" faccia prelevare tutto il nettare disponibile sul fiore e non soltanto una dose.

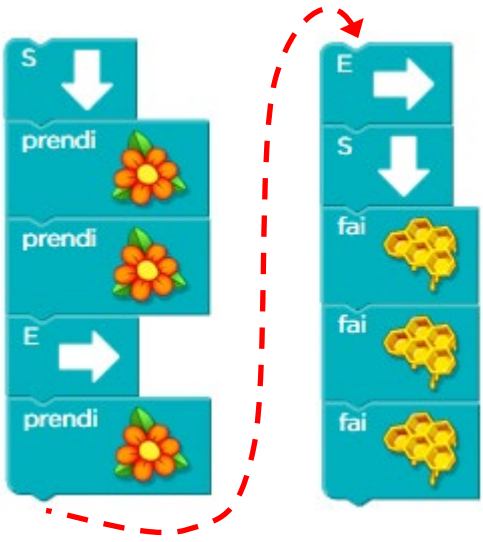
In entrambi i casi il docente può reindirizzare lo studente perplesso che non riesce a correggersi da solo sulla base dei messaggi di errore che il sistema fornisce (che non possono ovviamente avere la chiarezza e precisione dell'intervento di un essere umano).

La soluzione dunque è quella sotto mostrata (rappresentata su due colonne per motivi di spazio). Si noti anche che per muovere l'ape non è necessario che questa sia orientata verso la direzione in cui si deve spostare. Nello scenario sopra raffigurato essa è orientata verso Est ma compie il primo passo verso Nord.

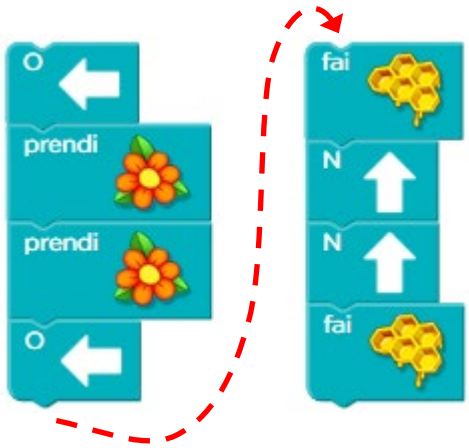


Per maggior chiarezza del comportamento delle varie istruzioni si può usare l'esecuzione "passo a passo", cliccando sul pulsante "Fai un passo": si osserverà che dopo ogni "prendi nettare" (o "fai miele") i numeri posti accanto al fiore o al favo diminuiscono di 1.

Un ulteriore variazione di tipo operativo introdotta dal successivo **esercizio 5** (sotto a sinistra) è che la raccolta del nettare deve essere fatta in momenti successivi, spostandosi quindi tra una raccolta e la successiva. La soluzione è quindi quella mostrata sotto a destra.

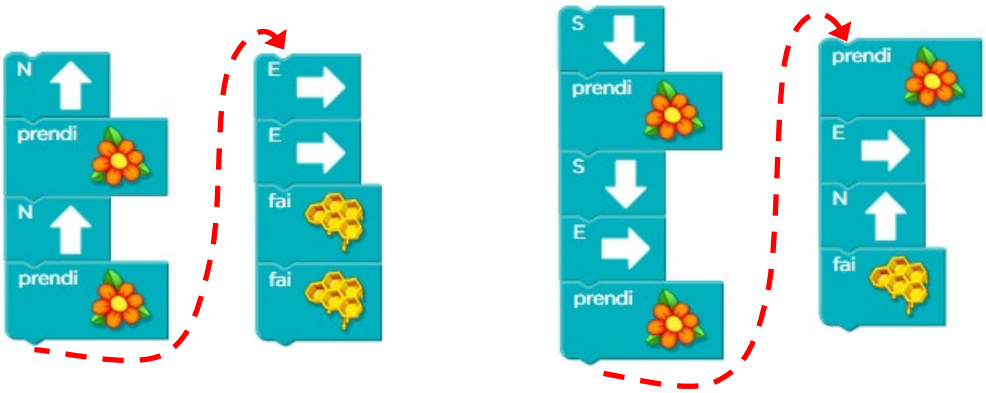


Nell'**esercizio 6** (sotto a sinistra) c'è ancora lo stesso tipo di situazione, tranne che per il fatto che questa volta il nettare viene preso da un unico fiore mentre è il miele a dover essere prodotto in due favi. La soluzione è quindi quella mostrata sotto a destra.



I successivi esercizi (dal 7 al 13) lavorano su combinazioni delle situazioni incontrate fino a questo momento. Una novità di un qualche rilievo si incontra solo nell'**esercizio 8** (sotto a sinistra) in cui vi è per l'ape la possibilità di muoversi in uno spazio più largo di quello avuto sinora a disposizione, costituito sostanzialmente da un corridoio, e nell'**esercizio 11** (sotto a destra) in cui l'ape può muoversi in due direzioni lungo il percorso circolare a sua disposizione. In quest'ultimo esercizio, inoltre, la quantità di miele da produrre è diversa dalla quantità totale di nettare raccolto.

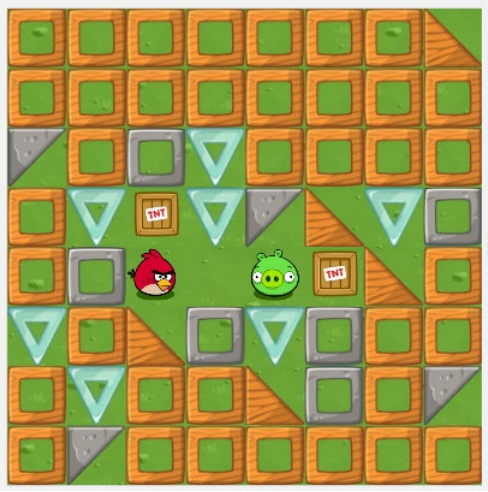
Le soluzioni per questi casi sono presentate, corrispondentemente, qui sotto, a sinistra per l'esercizio 8 e a destra per l'esercizio 11.



Osserviamo che, a causa del maggiore spazio in cui l'ape può muoversi qualche studente potrebbe produrre soluzioni non completamente corrette, nel senso che raggiungono comunque l'obiettivo specificato, ma usano più istruzioni di quante sono strettamente necessarie. In casi di questo genere va appunto chiarito che ciò che non è corretto è il non aver ottenuto una soluzione col minor numero possibile di blocchi.

Gli ultimi due esercizi di questa lezione (14 e 15) sono indicati come esercizi di verifica, ma non sono reali verifiche rispetto ai concetti affrontati dalla lezione dal momento che investigano, in modo peraltro assai semplice, le capacità aritmetiche degli studenti. Possono quindi essere trascurati.

3.1.2 Corso 2: Lezione 3 (Sequenza – Labirinto)



Il contesto della lezione è quello di un labirinto (vedi a fianco) nel quale l'esecutore (un uccellino di Angry Birds) ha solo tre azioni a disposizione: lo spostamento in avanti e la possibilità di girare a destra o a sinistra (vedi sotto).

In questa lezione (così come in tutte le successive) le istruzioni sono specificate in forma soltanto testuale senza far ricorso a simboli. Inoltre, le azioni per girare a destra o a sinistra hanno l'effetto di far ruotare l'uccellino su sé stesso rimanendo fer-

mo nella stessa casella. In altre parole, esse fanno riferimento all'esecutore stesso e non si intendono in senso assoluto.

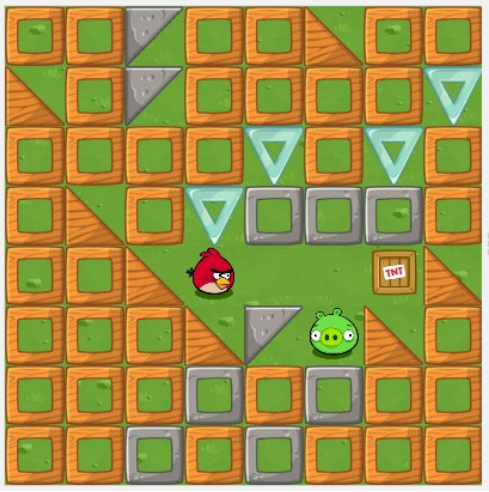
Per comprenderne l'effetto sull'esecutore, lo studente deve quindi mentalmente immaginarsi al posto dell'esecutore stesso. Poiché questa operazione non è concettualmente semplice, nel capitolo successivo ("Versione semplificata del Modulo 1", a pag. 53) viene sviluppata una versione di questo modulo adatta anche ad allievi più piccoli, in cui il movimento viene invece specificato in senso assoluto, con lo stesso approccio usato nella precedente "Corso 1: Lezione 7 (Sequenza – Ape)".

Nell'**esercizio 1**, quello sopra illustrato, lo studente potrebbe pensare che la soluzione sia quella mostrata a fianco, dal momento che può, giustamente, interpretare il blocchetto con l'azione "vai avanti" come "mettiti in moto e procedi in avanti".

Invece, la sua interpretazione corretta è "fai un solo passo in avanti" (come si può constatare da soli cliccando sul pulsante "Fai un passo" presente accanto a "Esegui" sotto la scena del labirinto) e pertanto la soluzione corretta del primo esercizio è mostrata a fianco.

L'eventuale presenza di tale incertezza negli studenti può essere sfruttata positivamente a scopo educativo. Essa infatti consente l'opportunità di riflettere come la procedura da costruire per l'esecutore (cioè il programma) deve specificare azioni che siano completamente ed univocamente interpretabili dall'esecutore che le deve mettere in opera. Si tratta di un elemento essenziale per padroneggiare la programmazione informatica.

L'esercizio seguente (2) è analogo al primo – quindi non lo trattiamo, ma con l'**esercizio 3** (vedi sotto) si inizia a far pratica con l'azione "gira a …". Osserviamo preliminarmente che può non essere immediato per lo studente capire qual è il sistema di riferimento usato. Ovvero, lo studente può interpretare i termini "a sinistra" e "a destra" come riferiti a sé stesso e non all'uccellino e quindi, ad esempio, trovarsi disorientato in questo esercizio perché non c'è nessuna istruzione per andare verso il basso. Se lo studente non è in grado di risolvere da solo la difficoltà l'insegnante può intervenire, ma dal momento che l'ambiente consente di sperimentare senza alcuna difficoltà o penalità si suggerisce di far fare dei tentativi in autonomia allo studente.

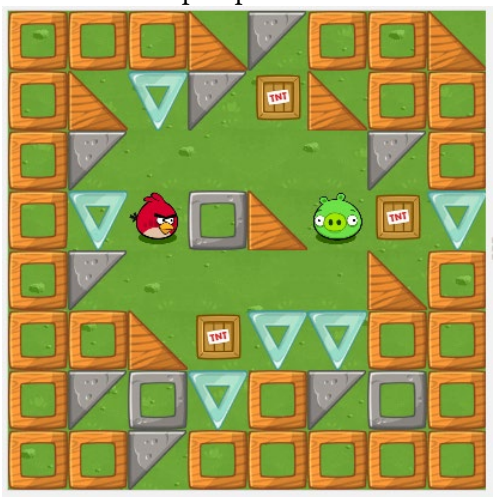


Ricordiamo che è possibile cliccare su "Esegui" anche se si è consapevoli di non aver completato il programma, semplicemente per verificare che cosa accade con le istruzioni scritte fino a quel momento. Analogamente, è possibile cliccare su "Fai un passo" per eseguire il programma un passo alla volta e poter rendersi meglio conto dell'effetto di ogni istruzione.

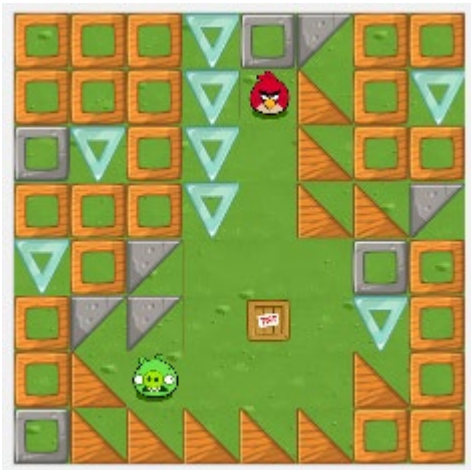
Una volta compreso che la direzione è quella relativa all'uccellino, potrebbe presentarsi un'incertezza come quella dell'esercizio 1: lo studente può pensare che la soluzione corretta sia quella a fianco, dal momento che può interpretare l'azione "gira a destra" come "continua a muoverti verso destra".

Invece, la sua interpretazione corretta è "gira a destra rimanendo nel punto in cui ti trovi" e pertanto la soluzione dell'esercizio 3 è quella mostrata qui a fianco.

Poiché il successivo esercizio 4 fa ancora pratica sui cambiamenti di direzione non lo discutiamo, mentre nell'**esercizio 5**, la cui ambientazione è mostrata qui sotto, sono possibili due soluzioni, dal momento che l'uccellino può passare sia in alto che in basso per raggiungere il maialino. Anche questo può essere lo spunto per evidenziare che possono esistere modi alternativi ed equivalenti di risolvere un problema.



Le due possibili soluzioni (simmetriche) sono dunque le seguenti.

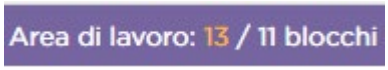


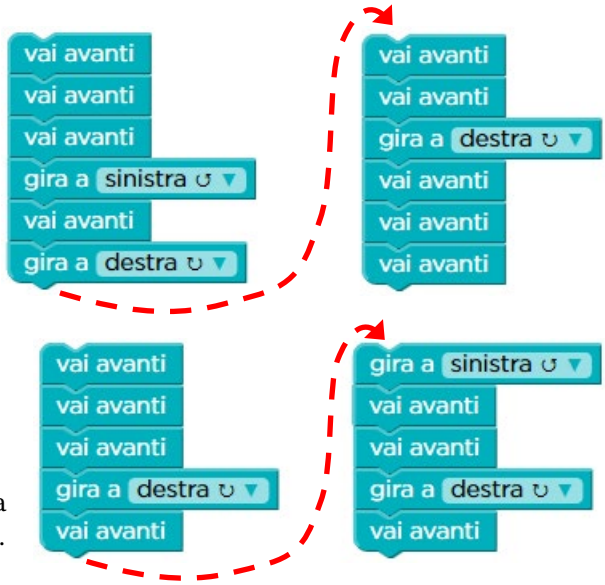
Nei due esercizi seguenti (6 e 7) si affrontano percorsi lievemente più complessi ma sostanzialmente equivalenti e non sono quindi discussi.

Anche nell'**esercizio 8**, qui a fianco, l'uccellino ha due strade possibili per raggiungere il maialino, come nel precedente esercizio 6, ma in questo caso una delle due è più lunga dell'altra.

Infatti, se il percorso dell'uccellino passa accanto alla dinamite (TNT) avendola sulla sua destra la soluzione (mostrata qui a fianco) consiste di 13 blocchi, quando invece ne

basterebbero 11, come segnalato nella zona viola dell'area di lavoro.

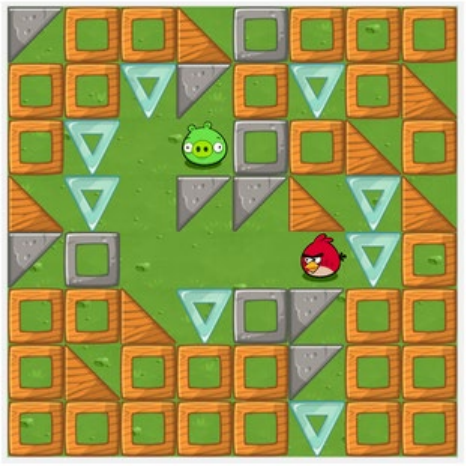




Non è opportuno considerare questa soluzione come sbagliata, dal momento che l'esecutore raggiunge il suo obiettivo (l'uccellino arriva al maialino). Si tratta solo di una soluzione meno efficiente della soluzione più corretta, qui a fianco.

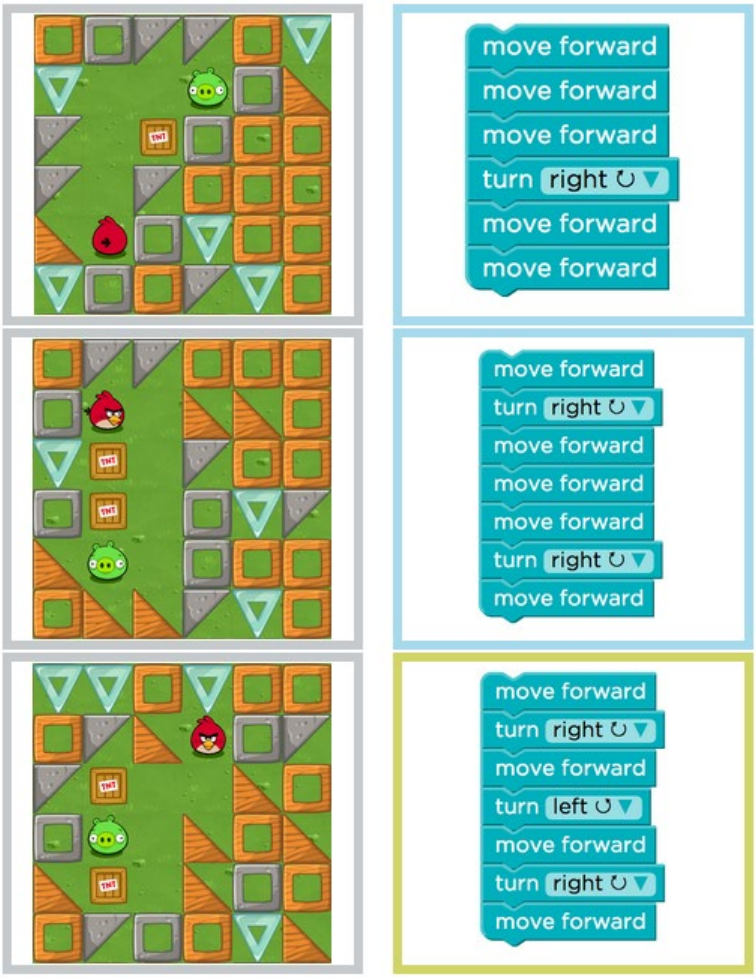
L'esercizio successivo (9) è ancora un percorso lineare con cambiamenti di direzione – e non viene quindi discusso, mentre gli ultimi due esercizi sono esercizi di verifica.

Nel primo di questi (**esercizio 10** – Risposte Multiple) si tratta di scegliere la sequenza di blocchi corretta per il contesto in cui si trova l'uccellino, che viene mostrato qui sotto a sinistra. Le possibili risposte sono in inglese. La corrispondenza tra le istruzioni in inglese e quelle in italiano è mostrata sotto a destra insieme alla soluzione.



move forward	=	vai avanti
turn right	=	gira a destra
turn left	=	gira a sinistra

Il secondo esercizio di verifica (**esercizio 11** – Associazioni) richiede di associare ad ogni contesto la corretta sequenza di blocchi. Anche in questo caso le possibili risposte sono programmi con le istruzioni in inglese e la soluzione è qui a fianco.



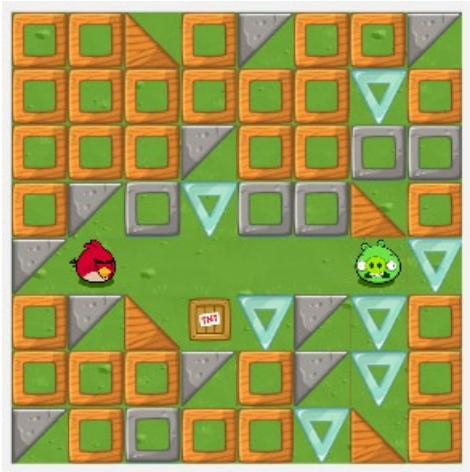
3.2 La ripetizione

Il concetto fondamentale che viene introdotto è quello della **ripetizione**, cioè di un'istruzione che, in modo sintetico, permette di far ripetere all'esecutore una o più azioni per un certo numero di volte. L'istruzione di ripetizione viene anche detta **ciclo**. Per lavorare su questo concetto si usano le lezioni 6 e 8 del corso 2.

Gli studenti imparano ad usare i cicli in modo via via più complesso per guidare nella lezione 6 del corso 2 l'uccellino attraverso il Labirinto verso la sua mèta e nella lezione 8 del corso 2 per guidare l'Ape nella raccolta del miele e nella produzione del nettare.

3.2.1 Corso 2: Lezione 6 (Ripetizione – Labirinto)

In questa lezione dapprima si ripetono singole azioni (ripetizione semplice – esercizio 2), poi si mettono in sequenza più ripetizioni (esercizio 4), infine si ripetono sequenze di azioni (ripetizione composta – esercizi 6, 7, 8 e 9). Le prime due tipologie sono introdotte con una ripetizione esplicita e successivamente lo studente deve determinare la ripetizione appropriata.



Mentre l'esercizio 1 propone allo studente una situazione già incontrata nel caso di sequenze di istruzioni (quindi non lo discutiamo), nell'**esercizio 2**, a sinistra, si chiede allo studente come sostituire la soluzione che trova già pronta nell'area di lavoro, costituita dalla ripetizione per cinque volte del blocco "vai avanti" (vedi a destra), con una soluzione costituita da due soli blocchi.

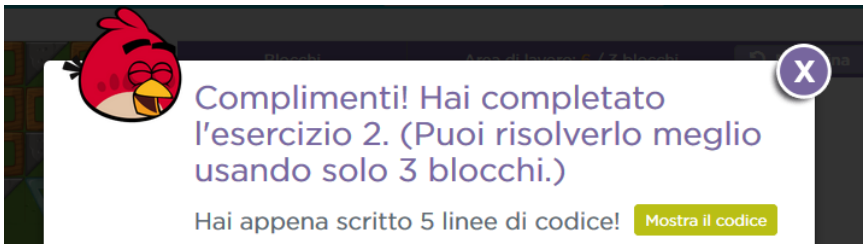
Adesso è disponibile un'ulteriore istruzione per guidare l'uccellino (cioè l'esecutore è in grado di realizzare una nuova azione), che fa sì che l'esecutore ripeta per 5 volte le istruzioni che verranno inserite al suo interno (vedi a fianco).

Lo studente dovrebbe quindi arrivare a comprendere che la soluzione dell'esercizio è quella consistente nel ripetere per 5 volte il blocco "vai avanti", come a fianco mostrato.

Ricordiamo che entrambi i programmi (sia quello con le 5 ripetizioni esplicite del blocco "vai avanti" che quello col blocco "ripeti 5 volte") sono corretti rispetto all'obiettivo di far sì che l'uccellino di Angry Birds raggiunga il maialino, ma solo il secondo è corretto rispetto all'obiettivo di farlo col minor nume-

ro possibile di blocchi. Osserviamo anche che il numero di istruzioni di cui è composto il secondo programma è inferiore al numero di passi che il programma stesso compie durante la sua esecuzione.

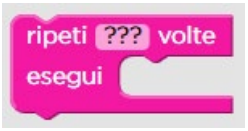
È importante notare che la presenza dell'insegnante è fondamentale, in casi di questo genere, per assicurarsi che lo studente comprenda qual è l'obiettivo dell'esercizio e lavori su di esso. Se infatti in questo esercizio lo studente si limita a cliccare sul pulsante "Esegui" sulla soluzione che trova già pronta, questa lo conduce comunque a risolvere l'esercizio, anche se in modo imperfetto, vedi sotto



C'è dunque il rischio che, in un'ottica da videogioco, lo studente consideri di aver "completato il livello" e proceda avanti ignaro. L'intervento dell'educatore è in tali casi necessario affinché l'attività formativa sia efficace.

Mentre il successivo esercizio 3 è concettualmente analogo a quello appena trattato, con l'**esercizio 4** si combina il concetto di sequenza e quello di ripetizione. Nello scenario che si presenta allo studente, raffigurato a fianco, è infatti necessario mettere in sequenza la ripetizione che permette all'uccellino di muoversi verso destra e la ripetizione per muoversi verso il basso

In questo caso il blocco con l'istruzione di ripetizione presenta dei punti interrogativi al posto del numero 5 dei precedenti esercizi (vedi sotto a sinistra). Per inserire il numero di ripetizioni desiderato basta cliccare sui punti interrogativi, scrivere il valore desiderato e premere il tasto "Invio" ("Return" o "Enter" sulle tastiere inglesi). La soluzione corretta è pertanto quella mostrata sotto a destra)



Osserviamo che uno studente che non presta molta attenzione alla specifica dell'esercizio potrebbe in questo definire una soluzione (corretta ma inefficiente perché usa più istruzioni del necessario) del tipo di quella mostrata a fianco. Come precedentemente notato, l'intervento dell'insegnante è fondamentale per assicurare in tali casi efficacia all'attività didattica.

L'esercizio seguente (5) ripete lo stesso concetto, ma l'**esercizio 6** (vedi sotto) presenta un nuovo modo di combinare sequenza e ripetizione. Viene infatti proposto allo studente una situazione per la quale una sequenza di blocchi è già pronta, e si tratta di capire come ripeterla per raggiungere l'obiettivo desiderato.

In questo caso l'ambientazione è quella di un altro gioco molto noto, Plants vs Zombies[4], in cui uno Zombie ha l'obiettivo di raggiungere il girasole per sfamarsi, evitando al tempo stesso le piante carnivore. i blocchi già pronti sono i seguenti

[4] Plant vs Zombies ® e © 2020 di Electronic Arts Inc. Tutti i diritti riservati.

Prima di tutto osserviamo che lo Zombie è girato verso sinistra e se proviamo ad eseguire il codice così come proposto arriviamo nella situazione a fianco raffigurata, nella quale possiamo notare che si ripresenta lo stesso tipo di situazione iniziale. Lo Zombie deve infatti girare a destra per imboccare correttamente la strada che deve compiere, che è costituita di due caselle. Si tratta quindi di ripetere una seconda volta la stessa sequenza di istruzioni e poi ripetendola una terza volta si conduce lo Zombie sano e salvo fino al girasole.

La soluzione così ottenuta (mostrata qui sotto su più colonne per motivi di spazio) raggiunge l'obiettivo ma non è completamente corretta, dal momento che non usa il blocco "ripeti ... volte".

Osservare la soluzione così costruita è comunque utile per poter individuare l'appropriata sequenza di istruzione da ripetere ed arrivare a definire la soluzione corretta, qui sotto, che usa il blocco "ripeti ... volte".

Anche in questo caso l'insegnante deve fare attenzione allo studente che trascura di leggere con attenzione il testo dell'esercizio e semplicemente ripete in modo esplicito tutte le istruzioni, come più sopra illustrato. Procedendo in tal modo si risolve l'esercizio ma non si apprende il concetto della ripetizione. Il sistema segnala infatti che l'esercizio potrebbe essere risolto meglio con meno blocchi ma consente comunque il passaggio all'esercizio successivo.

La stessa tipologia di combinazione fra sequenza e ripetizione si incontra nell'**esercizio 7**. Il percorso che deve fare lo Zombie (vedi a fianco) è più complesso, ma la sequenza di azioni di base è già pronta nell'area di lavoro (vedi sotto)

Si tratta quindi di determinare quante volte la sequenza deve essere ripetuta. Anche in questo caso, eseguirla così come inizialmente proposta può aiutare a capire sia il suo funzionamento sia il numero totale di ripetizioni necessarie. Ad una prima analisi dell'esercizio, poiché si vede nell'immagine che lo Zombie deve scendere cinque "gradini" per raggiungere il girasole, sembrerebbe che la soluzione corretta sia quella a fianco.

In effetti, così facendo, l'esercizio viene risolto e si passa all'esercizio successivo. Osservando però la fascia viola sopra l'area di lavoro si nota che sono stati usati meno blocchi del previsto (6 invece di 9).

Questo accade perché nella soluzione sopra mostrata con cinque ripetizioni vi è un'istruzione che non serve. Nella quinta ed ultima ripetizione delle istruzioni all'interno del blocco "ripeti" non è infatti necessario eseguire l'ultima istruzione "gira a sinistra", dal momento che lo Zombie è già arrivato al girasole. La soluzione più corretta di questo esercizio è pertanto quella qui a fianco, in cui si ripetono le istruzioni all'intero del blocco solo 4 volte e poi vi sono 3 istruzioni aggiuntive per completare l'esercizio.

Sembra un'osservazione minore, ma il tema dell'individuare in modo corretto le istruzioni da ripetere nei cicli è importante, dal momento che il computer, diversamente dagli esseri umani, non ha in genere flessibilità (a meno che – come in questo caso – non sia stato appositamente programmato per considerare l'esercizio risolto anche con le cinque ripetizioni) e deve ricevere istruzioni esatte fino all'ultimo dettaglio.

Osserviamo che mentre quest'ultima soluzione riportata è meno efficiente da un punto di vista di numero di istruzioni usate (perché è costituita da 8 blocchi rispetto ai 5 della precedente) è però caratterizzata da una maggiore efficienza da un punto di vista di tempo.

Se analizziamo i passi che sono effettivamente eseguiti da questa soluzione, vediamo che per 4 volte si eseguono 4 istruzioni (i 4 blocchi all'interno del "ripeti 4 volte") e poi si eseguono 3 istruzioni (i 3 blocchi finali). Il conteggio totale del numero di passi compiuti dal programma è quindi fornito dall'espressione (4 x 4) + 3 che fornisce 19 in totale.

Il programma della precedente soluzione compie invece un numero di passi dato dall'espressione 5 x 4 cioè 20, ed è quindi peggiore per quanto riguarda l'**efficienza di tempo**.

Bisogna altresì osservare che nelle lezioni di Code.org non sempre viene indicato in modo esatto il numero di blocchi delle soluzioni che sono esatte nel senso che non contengono ripetizioni inutili. E se ne vedrà un esempio proprio nel successivo esercizio.

L'**esercizio 8** (vedi a sinistra) presenta una situazione simile a quello dell'esercizio 6 ma in questo caso lo Zombie ha due possibili strade per raggiungere il girasole e sono quindi possibili due soluzioni.

Mostriamo qui a destra solo quella più efficiente per numero di istruzioni usate dal momento che per questa soluzione uno studente particolarmente maturo potrebbe individuare un ulteriore miglioramento.

Prima di discuterla, osservate come nella terza ripetizione non sia necessario effettuare l'ultima istruzione "gira a destra". Cioè, per questo esercizio, la soluzione (più precisa) che non ripete inutilmente istruzioni (e quindi più efficiente dal punto di vista di tempo) sarebbe quella a fianco. Questa soluzione però usa complessivamente 8 blocchi rispetto ai 6 che costituiscono l'obiettivo dell'esercizio[5]. In questo caso il sistema non la considera del tutto corretta e invita a risolvere l'esercizio con meno blocchi (ovve-

[5] L'efficienza di tempo di questa soluzione è (2 x 4) + (3 x 1), cioè 11 passi, migliore dell'efficienza di tempo della precedente soluzione, che richiede 3 x 4 = 12 passi.

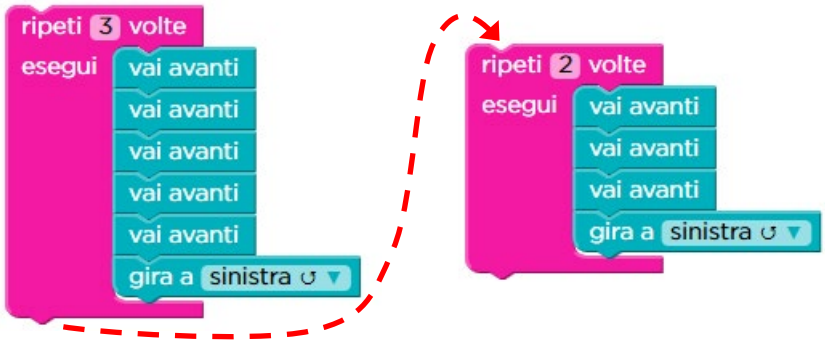
ro usando la soluzione– meno precisa – prima mostrata).

Assumiamo quindi di considerare come corretta la soluzione con le tre ripetizioni. Per questa, uno studente particolarmente maturo, osservando che all'interno del blocco di ripetizione vi è a sua volta la ripetizione per tre volte del blocco "vai avanti", potrebbe trovare la soluzione mostrata a fianco, che risolve l'esercizio usando un blocco di meno.

Questo tipo di soluzione, che inserisce un ciclo all'interno di un altro ciclo, è al di là del livello medio di sviluppo raggiunto generalmente dagli studenti a questo stadio del percorso e verrà discussa più avanti (vedi sezione 6.2 "Il ciclo annidato", a pag. 109). La segnaliamo qui per completezza di discussione[6].

Il successivo **esercizio 9** presenta, in una situazione più complessa, una configurazione analoga, mostrata qui a fianco.

La soluzione canonica che il sistema si aspetta è raffigurata più sotto, su due colonne per motivi di spazio.



[6] Anche a questa soluzione con il ciclo annidato soffre dell'imprecisione dovuta all'inutile ripetizione di "gira a destra". Sulla base di quanto visto sinora, il lettore dovrebbe essere in grado di rimuovere tale imprecisione. Si ottiene in tal caso una soluzione con un blocco in più dell'obiettivo di risolvere l'esercizio con 6 blocchi. L'efficienza di tempo della soluzione come sopra raffigurata è data dall'espressione {3 x [(2 x 1) + 1]}, cioè 9 passi, mentre l'efficienza di tempo della soluzione più precisa è caratterizzata da 1 passo in meno.

Anche in questo caso, infatti, tale soluzione canonica potrebbe essere migliorata da uno studente particolarmente maturo. Osservando infatti la ripetizione dei blocchi "vai avanti" all'interno dei blocchi di ripetizione lo studente potrebbe infatti produrre la soluzione mostrata a fianco (più efficiente come numero di istruzioni perché usa solo 9 blocchi in totale rispetto ai 13 attesi per la soluzione canonica).

Anche in questo caso tale soluzione è al di là del livello medio di sviluppo degli studenti e viene segnalata solo per completezza[7].

In questo caso è inoltre possibile produrre una soluzione senza l'imprecisione dovuta all'inutile ripetizione di "gira a sinistra" e rispettando l'obiettivo di risolvere l'esercizio con 13 blocchi complessivi. Sono possibili almeno le due soluzioni mostrate nel seguito, rispettivamente con 10 e 11 blocchi complessivi.

Anche queste sono al di là del livello medio di sviluppo degli studenti e segnalate solo per completezza[8].

I successivi due esercizi (10 e 11) sono ancora dedicati alla ripetizione di sequenze di istruzioni. Una volta individuata la successione di istruzioni che risolve l'esercizio, identificare e ripetere la sequenza appropriata il giusto numero di volte (come fatto nel precedente esercizio 6, pag. 25) non dovrebbe a questo punto presentare grandi

[7] Sempre per completezza, si segnala che l'efficienza di tempo di questa soluzione è costituita da {3 x [5 x (1) + 1]} + {2 x [3 x (1) + 1]} passi, ovvero 26 in totale.

[8] L'efficienza di tempo della due soluzioni più precise è la stessa ed è fornita per la prima soluzione dall'espressione {3 x [5 x (1) + 1]} + [3 x (1)] + 1 + [3 x (1)] e per la seconda dall'espressione {2 x [5 x (1) + 1]} + [5 x (1)] + {2 x [1 + 3 x (1)]}, che in entrambi i casi risultano in 25 passi in totale.

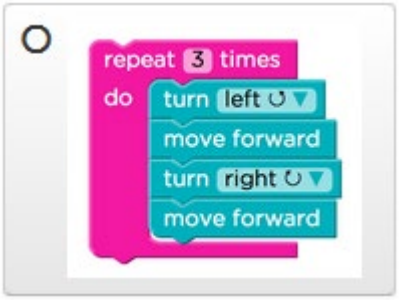
difficoltà per lo studente. Ricordiamo che usare l'esecuzione "passo a passo" è una tecnica utile per capire bene se si sta risolvendo correttamente l'esercizio.

Dopo un ulteriore esercizio di riepilogo (12), gli ultimi due esercizi sono di verifica. Nel primo (**esercizio 13** – Risposte Multiple) si tratta di scegliere la sequenza di blocchi (fra le quattro possibili) che nel contesto presentato permette allo Zombie di raggiungere sano e salvo il girasole.

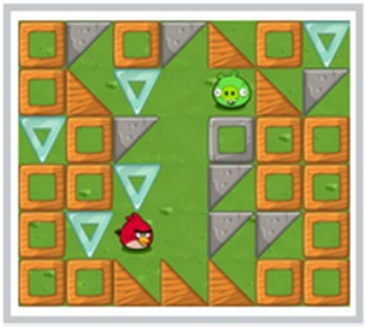
Le possibili risposte sono in inglese. La corrispondenza tra le istruzioni in inglese e quelle in italiano è la seguente

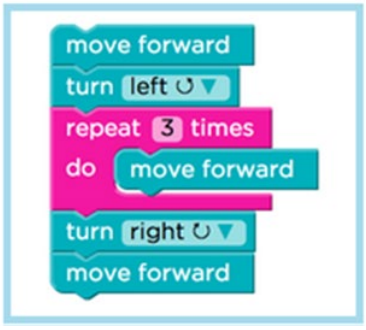
repeat ... times	=	ripeti ... volte
do	=	esegui
move forward	=	vai avanti
move backward	=	vai indietro
turn right	=	gira a destra
turn left	=	gira a sinistra
degrees	=	gradi

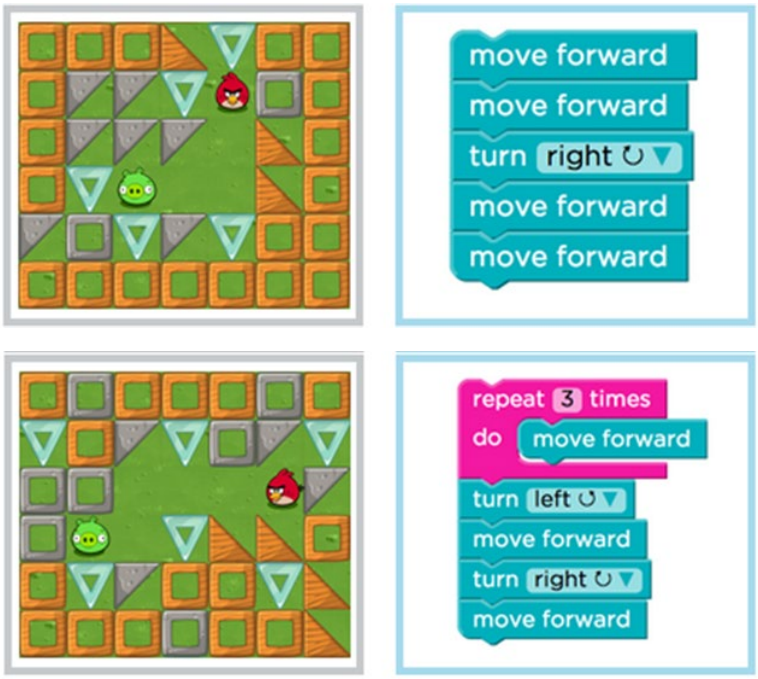
Pertanto la soluzione è quella raffigurata a fianco.



Il secondo esercizio di verifica (**esercizio 14** – Associazioni) richiede di associare ad ogni contesto la corretta sequenza di blocchi. Anche in questo caso le possibili risposte sono programmi con le istruzioni in inglese e la soluzione è la seguente







3.2.2 Corso 2: Lezione 8 (Ripetizione – Ape)

Anche in questa lezione si ripetono sia sequenze di azioni (esercizio 2) che singole azioni (esercizio 3), poi si mettono in sequenza più ripetizioni (esercizio 4) e si impara a ripetere sequenze date (esercizio 5). Sono successivamente presentati vari modi di usare i cicli per una più efficiente risoluzione degli esercizi, ma senza introdurre concetti nuovi.

L'**esercizio 1** ha il solo scopo di presentare allo studente la nuova ambientazione (vedi figura sotto), costituita da un'Ape che si muove su un prato per raggiungere fiori da cui raccogliere il nettare e favi nei quali produrre il miele. Le azioni a sua disposizione, infatti, oltre a quelle già incontrate con la precedente lezione sul Labirinto, includono due nuove azioni, appunto, per raccogliere il nettare ("prendi il nettare) e per produrre il miele ("fai il miele").

Si tratta però di azioni che ottengono il loro risultato soltanto nel contesto opportuno. Più precisamente, quando l'Ape si trova su di un fiore che ha del nettare (le dosi disponibili sono segnalate da un numero in basso a destra nella casella che contiene il fiore) l'azione "prendi il nettare" ha il risultato di prelevare dal fiore una dose di nettare (decrementando quindi di 1 il relativo numero).

Analogamente, quando l'Ape si trova su di un favo che deve essere riempito di miele (le dosi da produrre sono segnalate come per il caso del nettare del fiore) l'azione "fai il miele" ha il risultato di depositare una dose di miele nel favo (decrementando quindi di 1 il relativo numero).

Questo esercizio viene quindi risolto senza usare il meccanismo di ripetizione, dal momento che non è disponibile il blocco "ripeti ... volte", ma sequenziando correttamente tutte le istruzioni necessarie (vedi codice a fianco).

Nell'**esercizio 2** lo studente si trova esattamente nella stessa ambientazione ma ha a disposizione anche l'istruzione "ripeti ... volte" – già incontrata nella precedente lezione "Corso 2: Lezione 6 (Ripetizione – Labirinto)", e gli viene chiesto di usarla per scrivere un programma con meno blocchi (cioè più efficiente dal punto di vista di tempo).

Non dovrebbe questo punto essere difficile, dopo aver svolto la precedente lezione sul concetto di ripetizione (paragrafo 3.2.1 a pag. 23), individuare come soluzione quella mostrata a fianco.

Nell'**esercizio 3** (vedi sotto) lo studente viene ancora esortato ad usare un ciclo, ma questa volta per raccogliere tutto il nettare (si tratta di 4 dosi).

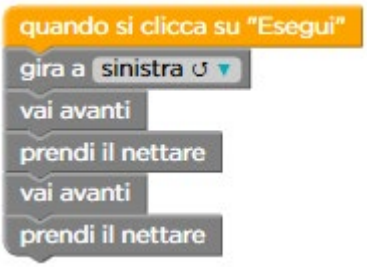
Non ha infatti senso usare il ciclo per i due passi di spostamenti in avanti, dal momento che usare un ciclo quando un blocco semplice[9] viene ripetuto solo due volte conduce ad usare sempre due blocchi. La soluzione è quindi quella mostrata sotto.

[9] Un blocco è "semplice" se non contiene al suo interno altri blocchi: il blocco "vai avanti" è semplice, il blocco "ripeti" non è semplice.

Nell'**esercizio 4** (mostrato a fianco) il ciclo dovrà essere usato non solo per la raccolta del nettare ma anche per la produzione del miele (soluzione sotto).

L'**esercizio 5** (qui sotto) introduce una complicazione concettuale perché richiede la ripetizione di una sequenza di istruzioni, e non più di una singola istruzione.

Lo studente trova già pronta la seguente sequenza di istruzioni (raffigurata in grigio e che quindi non può essere cancellata) e gli viene chiesto di ripeterla per risolvere l'esercizio.



Procedendo come nell'esercizio 6 (vedi pag. 25) della lezione "Corso 2: Lezione 6 (Ripetizione – Labirinto)" clicchiamo su "Esegui" per osservare cosa succede eseguendo il codice così come proposto. Si arriva in questa situazione mostrata a fianco (notare che nell'angolo in alto a sinistra è raffigurato cosa è stato raccolto/prodotto)

Si può notare che l'ape si trova in una situazione analoga quella iniziale, dal momento che deve girare a sinistra per imboccare cor-

rettamente la strada che deve compiere, che è costituita di due caselle. Si tratta quindi di ripetere una seconda volta la stessa sequenza di istruzioni e poi ripetendola una terza volta si conduce l'ape a completare il suo obiettivo.

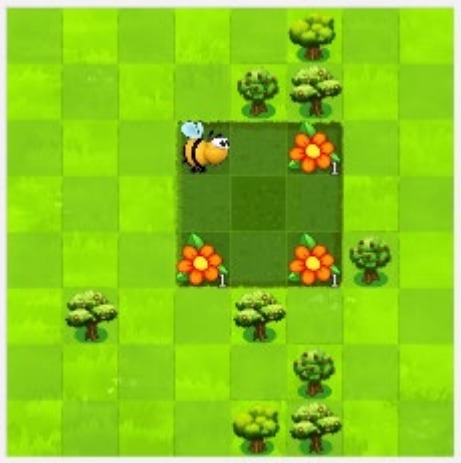
Data l'esperienza sviluppata con gli esercizi della precedente lezione "Corso 2: Lezione 6 (Ripetizione – Labirinto)", vedi paragrafo 3.2.1 a pag. 23, dovrebbe essere chiaro che la sequenza di blocchi che si trova già pronta deve essere inserita all'intero di un blocco "ripeti … volte" per ottenere la soluzione desiderata, come visualizzato qui a fianco.

Anche in questo caso l'insegnante deve fare attenzione allo studente che trascura di leggere con attenzione il testo dell'esercizio e semplicemente ripete in modo esplicito per 3 volte il blocco di istruzioni proposto, come sotto illustrato. La soluzione così ottenuta (mostrata su più colonne per motivi di spazio) raggiunge l'obiettivo ma non è completamente corretta, dal momento che non usa il blocco "ripeti … volte".

Procedendo in tal modo si risolve l'esercizio ma non si apprende il concetto della ripetizione. Il sistema segnala infatti che l'esercizio potrebbe essere risolto meglio con meno blocchi ma consente comunque il passaggio all'esercizio successivo.

L'esercizio seguente (6) propone lo stesso tipo di problema (ripetere una sequenza già pronta[10]).

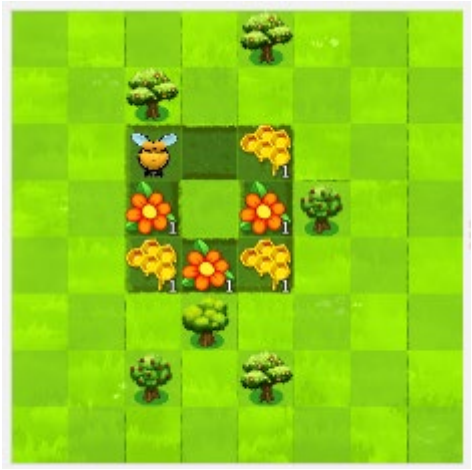
[10] Nella sequenza che si trova già pronta per l' esercizio 6 è presente a sua volta un blocco "ripeti … volte" per cui si ottiene un primo esempio di un concetto quello del "ciclo annidato" che verrà affrontato successivamente (vedi sezione 6.2 "Il ciclo annidato", a pag. 91).



Nell'**esercizio 7** (mostrato qui a sinistra) è lo studente a dover individuare la sequenza che poi dovrà essere ripetuta. La soluzione più efficiente si ottiene se l'ape procede in senso anti-orario (vedi sotto).

Se si sceglie di far procedere l'ape in senso orario si può ottenere:

- (vedi sotto a sinistra) una soluzione che, rispetto a quella sopra raffigurata, pur essendo altrettanto efficiente in termini di numero di istruzioni usate è meno efficiente in tempo (perché l'ultima istruzione che viene eseguita è inutile),
- oppure (vedi sotto a destra) una soluzione che rispetto a quella sopra raffigurata è più efficiente in tempo (vengono eseguiti 11 passi invece di 12) ma meno efficiente in numero di istruzioni usate.



I successivi due esercizi (8 e 9) sono ancora sulla ripetizione di sequenze. Lo stesso tipo di problematica si ritrova nell'**esercizio 10**, la cui soluzione canonica presenta però le caratteristiche già discusse per l'esercizio 8 della lezione "Corso 2: Lezione 6 (Ripetizione – Labirinto)" (vedi pag. 28), cioè la ripetizione inutile dell'ultima azione contenuta all'interno del blocco "ripeti … volte" (vedi sotto a destra).

La soluzione più precisa del suddetto esercizio è quindi quella a fianco (raffigurata su due colonne per motivi di spazio), che però contiene quattro blocchi in più rispetto al numero che il sistema si attende.

La stessa problematica (ripetizione inutile dell'ultimo blocco contenuto nel ciclo) si ritrova nei due esercizi successivi (11 e 12). L'esercizio ancora seguente (13) riepiloga uno scenario già visto,

mentre l'ultimo, **esercizio 14** – Risposte Multiple, è di verifica. Si tratta di scegliere la sequenza corretta di blocchi per lo scenario in cui si trova l'ape (vedi fianco). Le possibili risposte sono in inglese. La corrispondenza tra le istruzioni in inglese e quelle in italiano è quella mostrata a pag. 31 e la soluzione è quindi quella mostrata qui a fianco.

3.3 La verifica

La verifica, con conseguente correzione degli errori, è un elemento fondamentale dell'apprendimento della programmazione dal momento che lo studente è condotto a riflettere su cosa compie ogni singola istruzione e sul fatto che la loro sequenza conduca o meno al risultato desiderato. A questo scopo lo studente si mette, in un certo senso, nei panni del computer, e questo esercizio lo conduce ad una migliore comprensione dei concetti.

Nelle lezioni dedicate a questo concetto gli studenti si trovano di fronte esercizi che sono stati svolti in modo non corretto. Devono quindi esaminare il codice esistente per identificare gli errori, che possono essere costituiti da:

- blocchi mancanti,
- blocchi superflui,
- blocchi collocati nell'ordine sbagliato,
- blocchi sbagliati rispetto all'obiettivo dell'esercizio.

Per lavorare su questo aspetto si inizia con la lezione 5 del Corso 1, per iniziare a prendere confidenza con questo particolare contesto didattico. Se gli studenti sono particolarmente maturi si potrebbe anche iniziare direttamente con la lezione 10 del Corso 2, anche se può valer la pena svolgere comunque la lezione 5 del Corso 1 per entrare meglio in questa modalità di lavoro.

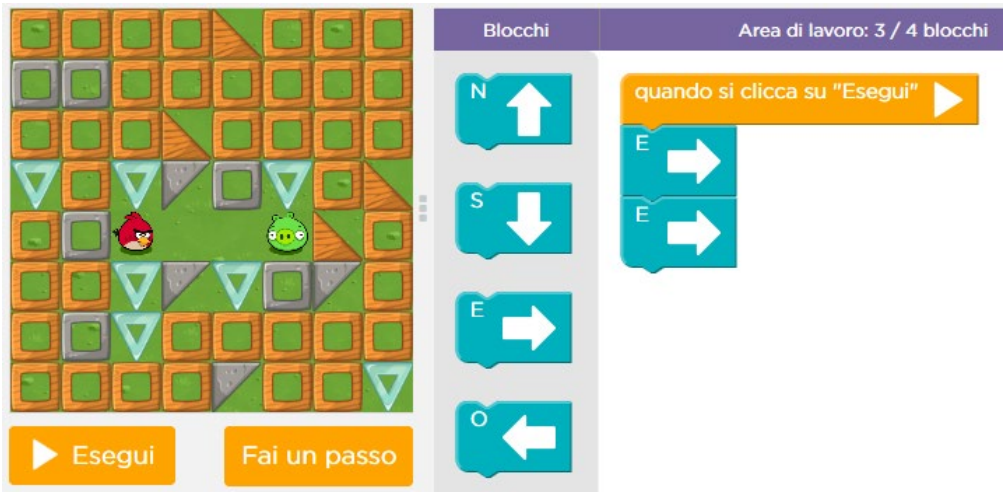
Nelle lezioni che lavorano su questa attività di verifica si osserverà che oltre al solito pulsante "Esegui" vi è il pulsante "Fai un passo" che permette di eseguire il programma un blocco alla volta.

Si raccomanda di insistere in modo particolare, soprattutto nei primi esercizi, sulla necessità che lo studente usi la modalità di esecuzione "un blocco alla volta", detta anche "passo a passo", per comprendere con sicurezza il punto in cui il programma esegue un'istruzione che non porta al risultato desiderato.

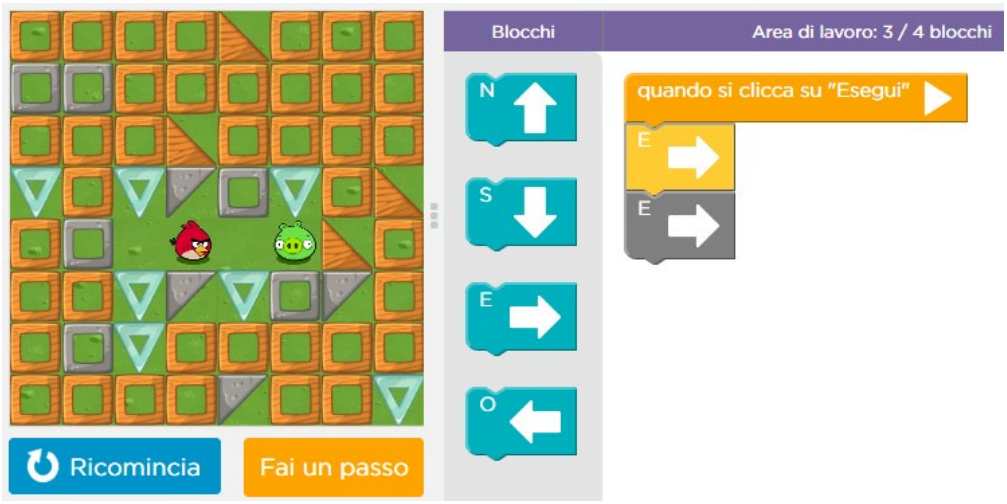
Si noti anche che, mentre il sistema è in questa modalità di esecuzione passo a passo non è possibile apportare correzioni al codice in esecuzione. Bisogna interromperla (cliccando su "Ricomincia), eseguire le modifiche desiderate, e poi ricominciarla con la modalità desiderata.

3.3.1 Corso 1: Lezione 5 (Verifica – Labirinto)

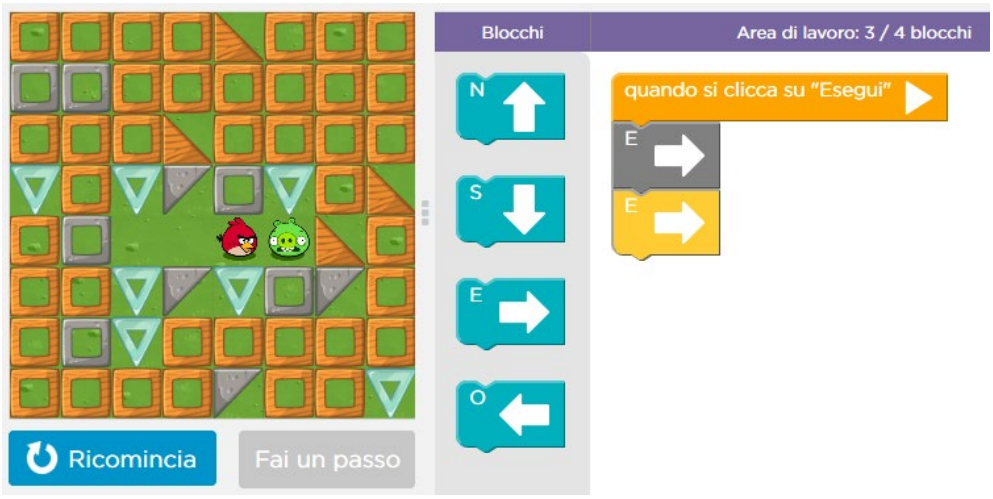
In questa lezione lo studente si trova nell'**esercizio 1** con la solita ambientazione di Angry Birds e il labirinto, ma le istruzioni sono più semplici, dal momento che sono costituite soltanto dai blocchi che permettono di fare un passo verso Nord, Sud, Esto, Ovest.



Dopo aver cliccato una volta su "Fai un passo" il sistema si trova in questa configurazione



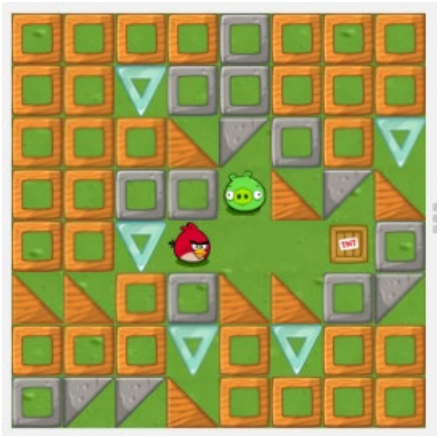
L'uccellino si è spostato di una casella verso Est, eseguendo il blocco evidenziato in giallo nell'area di lavoro. Cliccando un'ulteriore volta su "Fai un passo" si ottiene la situazione sotto raffigurata, in cui le istruzioni sono state tutte eseguite ma l'uccellino ancora non ha raggiunto il maialino, il che rende abbastanza evidente come sia necessario aggiungere un ulteriore blocco al programma. L'errore quindi consiste in un'**istruzione mancante** dal programma.



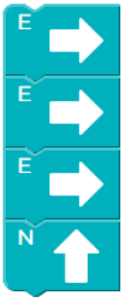
Questo meccanismo dell'esecuzione "passo a passo" del programma è fondamentale per l'attività di verifica. Mediante tale meccanismo, chi ha scritto il programma è forzato a calarsi nel ruolo del computer e riesce meglio a comprendere cosa effettivamente il programma sta facendo fare al computer.

Spesso infatti accade che, mentre con la nostra intelligenza abbiamo ben chiaro cosa il computer dovrebbe fare, non riusciamo ad essere fedeli nel tradurre questo in istruzioni rigorosamente complete e corrette per il computer. L'assoluta mancanza di intelligenza del computer porta quindi ad un risultato diverso da quello che vorremmo, laddove un esecutore umano potrebbe, almeno in alcuni casi, usare invece la sua intelligenza per correggere queste imprecisioni delle nostre istruzioni.

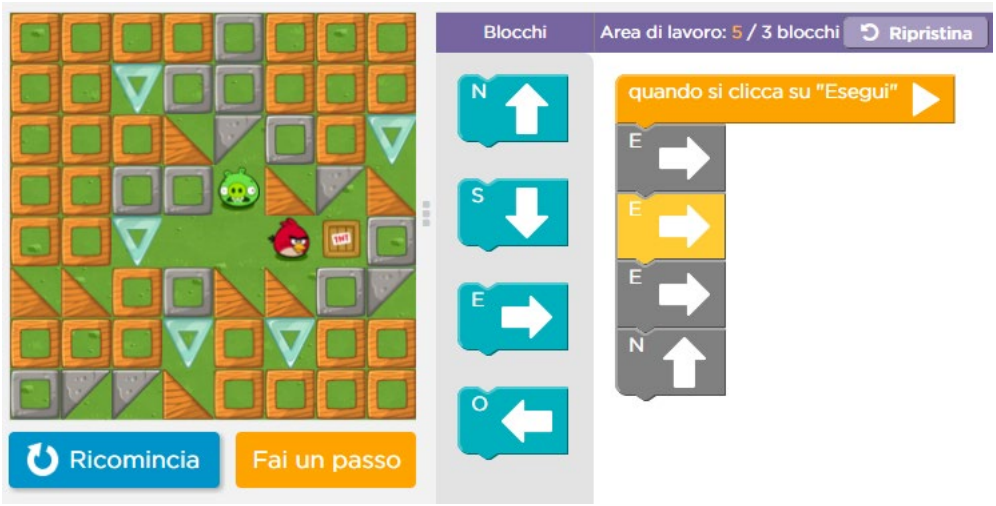
Un beneficio culturale dello studio della programmazione informatica è proprio questo allenamento alla riflessione e all'attenzione sulla corrispondenza tra ciò che vorremmo fosse eseguito ed il modo in cui chiediamo di eseguirlo.



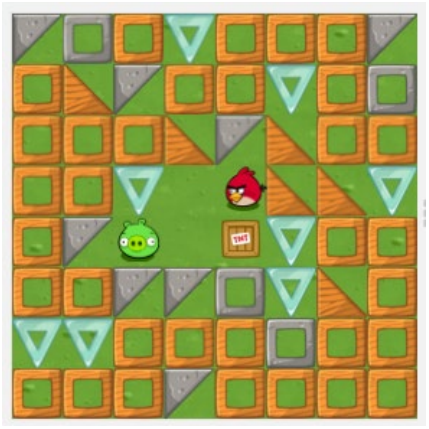
Mentre il successivo esercizio 2 è analogo al precedente e non viene quindi discusso, l'**esercizio 3** (a sinistra) presenta un diverso tipo di errore, dal momento che si chiede di eliminare dei blocchi dal programma (a destra) che si trova già pronto nell'area di lavoro.



Anche in questo caso, eseguendolo un passo alla volta ci si trova, dopo il secondo passo, in questa situazione



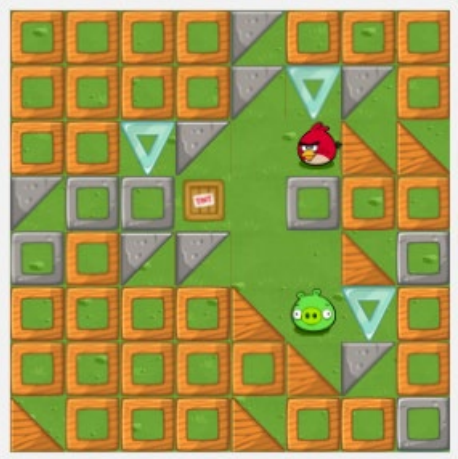
nella quale è chiaro che un ulteriore passo porterà il povero uccellino a saltare per aria! L'errore quindi consiste in questo caso in **istruzioni superflue** presenti nel programma e la soluzione corretta in questo caso consiste nel rimuovere due blocchi di movimento verso Est.

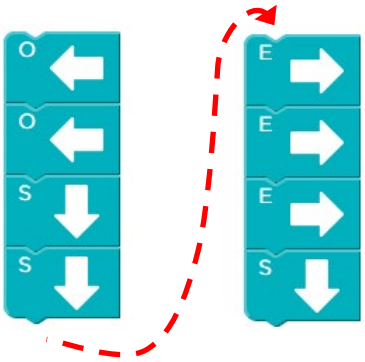


Con il programma (a destra) già pronto per l'**esercizio 4** (mostrato a sinistra) ci troviamo ancora in un'altra tipologia di errore, dal momento che effettivamente l'uccellino, per raggiungere il maialino, deve fare due passi verso Ovest, ma tra l'uno e l'altro deve spostarsi di un passo verso Sud.

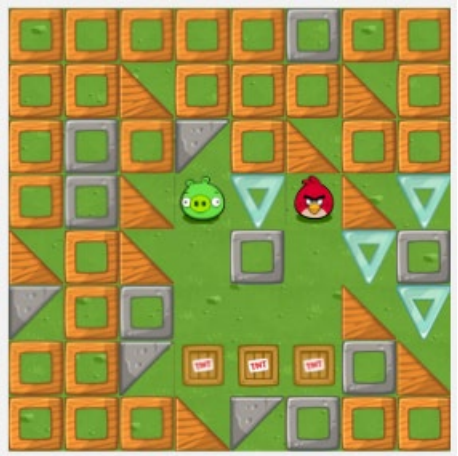
In questo caso quindi si tratta di **istruzioni in ordine sbagliato**, e la soluzione corretta è quella mostrata qui a destra:

Negli esercizi 5 e 6 abbiamo ancora degli errori dovuti a singole istruzioni superflue, ma con l'**esercizio 7** (sotto a sinistra) gli errori si moltiplicano. In questo caso non si tratta di un solo blocco superfluo nella soluzione che si trova già pronta nell'area di lavoro (qui raffigurata sotto a destra su due colonne per motivi di spazio), ma ce ne sono ben tre, in diversi punti del programma.

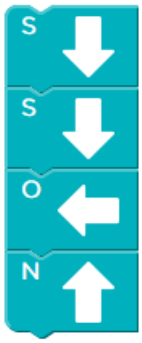




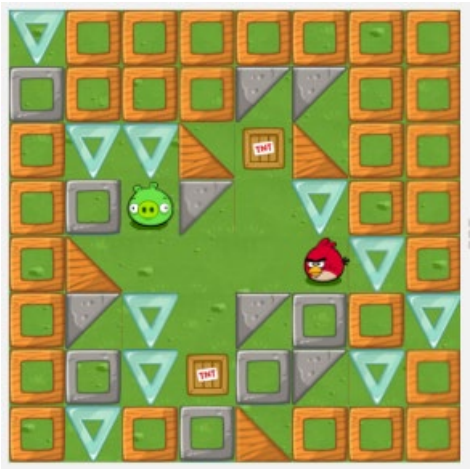
Per la corretta soluzione, che può essere individuata aiutandosi con l'esecuzione "passo a passo", bisogna quindi rimuovere il blocco verso Ovest di troppo e i due blocchi verso Est in eccesso.



La stessa tipologia di errore multiplo si verifica nell'**esercizio 8**, (ambientazione a sinistra e programma a destra) ma per il caso di **istruzioni mancanti**.



In questo caso è necessario aggiungere un blocco per andare verso Ovest ed un blocco per andare verso Nord. Usare l'esecuzione "passo a passo" aiuta a capire il punto in cui vanno inseriti.

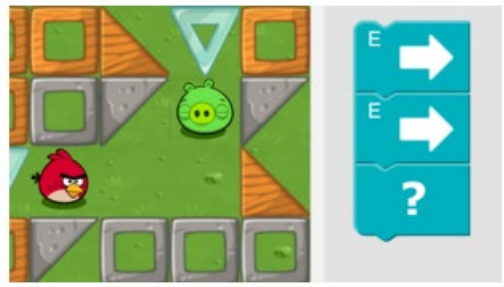


Con l'**esercizio 9** (ambientazione a sinistra e programma a destra) la situazione degli errori si complica ulteriormente, perché si verifica la presenza contemporanea di più tipologie di errori. In questo caso, oltre al caso di istruzioni mancanti vediamo anche la presenza di **istruzioni sbagliate**.

Per la soluzione, oltre ad aggiungere l'istruzione mancante (verso Ovest), vanno rimosse le istruzioni sbagliate (verso Sud) al cui posto va aggiunta l'altra istruzione mancante (verso Nord). Anche in questo caso l'esecuzione passo a passo aiuta a capire dove intervenire

Nell'**esercizio 10** abbiamo ancora sorgenti multiple di errori, costituite da più istruzioni mancanti e da più istruzioni superflue.

Per ottenere la soluzione corretta, infatti, va rimosso il blocco superfluo verso Nord, vanno aggiunti due blocchi verso Est, e va rimosso il blocco superfluo verso Sud. Come al solito, l'esecuzione "passo a passo" è utile per capire i punti in cui intervenire.



Il successivo esercizio 11 è ancora un caso di un'istruzione superflua da eliminare, mentre l'**esercizio 12** (quello finale) è una semplice verifica a risposte multiple. Si tratta di individuare quale blocco aggiungere alla sequenza parziale già mostrata per far arrivare l'uccellino di Angry Birds al maialino.

La soluzione è costituita dal blocco verso Nord.

Nel caso sia necessario costruire ulteriori esercizi per far allenare gli studenti a trovare gli errori, si possono usare a questo scopo gli esercizi della lezione 4 del Corso 1. In tutti questi esercizi, infatti, si può predisporre una soluzione scorretta nell'area di lavoro e chiedere agli studenti di trovare l'errore. Ovviamente, un'attività di questo genere richiede che, esercizio per esercizio, l'insegnante predisponga la soluzione sbagliata e poi esorti gli studenti alla riflessione. È quindi praticabile nel caso di un'organizzazione didattica in cui la classe lavora tutta insieme (per esempio con una LIM) sui vari esercizi.

3.3.2 Corso 2: Lezione 10 (Verifica – Ape)

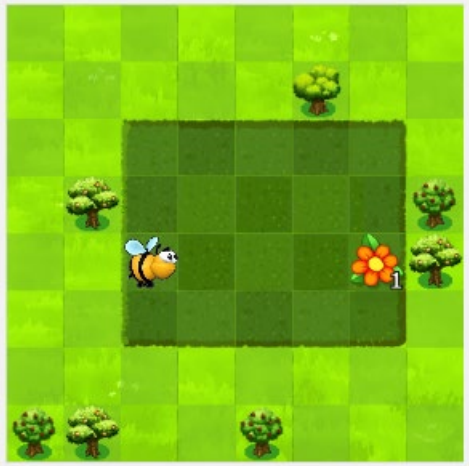
In questa lezione lo studente si trova con l'ambientazione di un'ape che deve raccogliere il nettare dai fiori e produrre il miele nei favi (la quantità di nettare o miele è indicata da un numeretto vicino al fiore o al favo).

Poiché a questo livello lo studente ha già acquisito il concetto di ripetizione ed ha quindi lavorato con meccanismi più complessi del semplice sequenziamento di istruzioni, la varietà di errori che si possono incontrare aumenta.

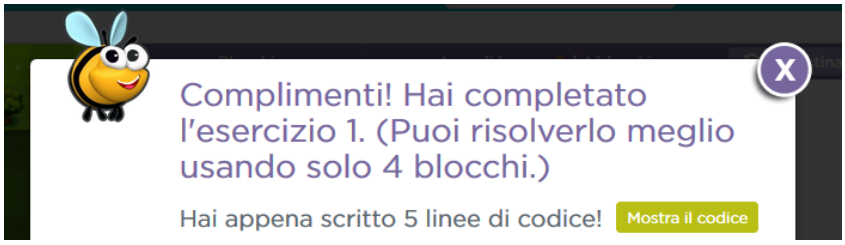
In particolare, il caso di istruzioni sbagliate acquisisce una nuova variante: l'**istruzione con parametro sbagliato**. È questo il caso in cui l'istruzione usata è corretta ma è sbagliato il suo "parametro", cioè la parte variabile ad essa associata. Il caso tipico è quello di un errore nel numero di ripetizioni di un ciclo, ma – con le istruzioni sinora viste – può essere sbagliato anche il verso in cui far girare l'esecutore (ape o uccellino). Lo si vedrà nell'esercizio 9.

Inoltre, anche il caso di istruzioni in ordine sbagliato acquisisce una nuova variante, cioè il caso di un'**istruzione nel posto sbagliato**. Questo si verifica, ad esempio, quando un'istruzione, invece di essere dentro a un ciclo è fuori o, viceversa, invece di essere fuori si trova dentro. Lo si vedrà nell'esercizio 10.

Nell'**esercizio 1**, il cui scenario è sotto raffigurato insieme alla soluzione (scorretta) già disponibile nell'area di lavoro, sembrerebbe che manchi un blocco "vai avanti".



Lo studente non attento potrebbe semplicemente aggiungere il blocco "vai avanti" e poiché il sistema riconosce l'esercizio come risolto (anche se con un numero di blocchi maggiore del necessario, vedi immagine seguente) lo studente potrebbe passare all'esercizio successivo senza aver realmente lavorato sull'obiettivo dell'esercizio stesso. Si tratta di un altro di quei casi in cui la guida dell'insegnante è fondamentale per assicurare efficacia didattica.



La soluzione (più efficiente per numero di istruzioni usate) infatti richiede l'introduzione del blocco di ripetizione. Si raccomanda comunque di non considerare scorretta la soluzione che usa i 4 blocchi "vai avanti" e senza il ciclo. Si tratta solo di una soluzione meno efficiente dal punto di vista di tempo[11].

Si osservi anche che in questo esercizio, e in quelli successivi, lo studente è forzato ad eseguire il codice un passo alla volta. È infatti presente il solo pulsante "Fai un passo". Ciò costituisce senz'altro un'utile disciplina per apprendere il metodo di correzione degli errori.

[11] Dal punto di vista dell'efficienza di tempo entrambe le soluzioni sono equivalenti.

Nell'**esercizio 2** ci si trova in un caso, già visto, di **istruzioni superflue**, come segnalato anche dal numero evidenziato in arancione nella striscia superiore viola.

La soluzione corretta è quella sotto a sinistra, ma si ricorda che uno studente più maturo potrebbe proporre anche quella sotto a destra (più efficiente per numero di istruzioni usate).

Nei due esercizi successivi diventa quasi indispensabile eseguire il programma "passo a passo" per capire in che punto è presente l'errore, dal momento che si tratta di situazioni più complesse delle precedenti, e può non essere agevole eseguire il programma interamente nella propria testa.

Nel caso dell'**esercizio 3** diventa chiaro, alla pressione per la quinta volta del pulsante "Fai un passo", che il blocco corrente (evidenziato in giallo nella figura seguente) è quello con l'azione sbagliata dal momento che fa girare l'ape verso destra invece che verso sinistra. Si vede infatti nella figura sotto che l'ape adesso è girata verso l'alto dove non c'è il fiore.

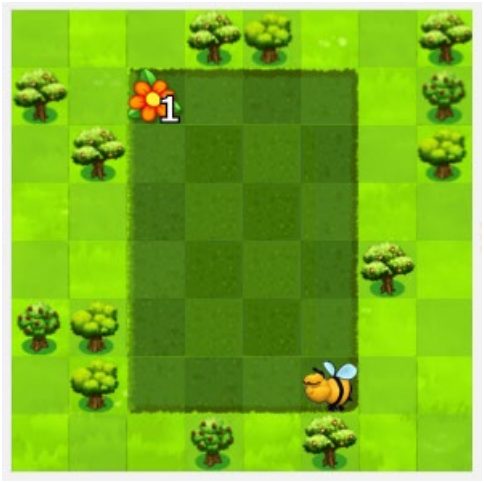
Notiamo che in questo caso l'istruzione non è del tutto sbagliata, ma è sbagliato il "parametro" (cioè la parte variabile) dell'istruzione stessa.

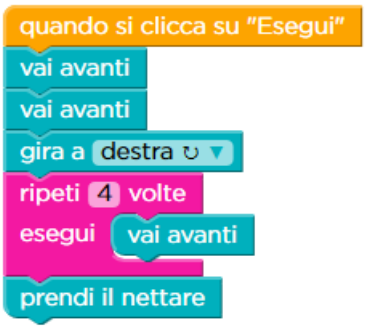
L'**esercizio 4** presenta una situazione un po' più complessa, dal momento che vi sono più errori da correggere. Anche in questo caso diventa chiaro cosa fare alla pressione per la sesta volta del pulsante "Fai un passo", quando si ottiene la configurazione mostrata sotto a sinistra,

dal momento che ci si accorge che vi è sia un'**istruzione mancante** (prendi il nettare) che un'**istruzione con parametro sbagliato** ("sinistra", che dovrebbe invece essere "destra").

Con l'**esercizio 5** viene reintrodotto il pulsante "Esegui" per poter mandare in esecuzione il programma tutto di seguito. Poiché in questo, e negli esercizi seguenti, vi sono cicli con numerose ripetizioni, mandare in esecuzione le istruzioni tutte insieme può condurre più velocemente a capire dove sia l'errore.

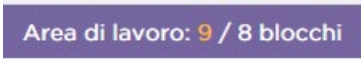
Nell'**esercizio 6** (sotto a sinistra) è necessario fare attenzione al fatto che lo studente potrebbe produrre soluzioni meno efficienti di quella canonica se, dopo aver osservato l'esecuzione del codice che trova già pronto (vedi sotto a destra),





che fa arrivare l'ape nella situazione mostrata sotto, valuta che sia necessario, oltre a correggere il parametro del ciclo, anche aggiungere dei blocchi per arrivare correttamente al fiore. Una soluzione di questo tipo è mostrata sotto a destra e, seppur corretta, non è la più efficiente per numero di istruzioni.

Il contatore dei blocchi utilizzati mostra infatti che è stato usato un blocco in più della soluzione canonica.

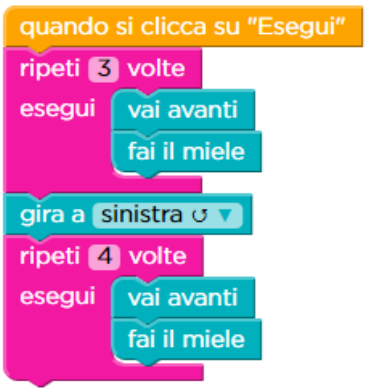


La soluzione canonica dell'esercizio è visualizzata sotto a sinistra. Segnaliamo anche che uno studente più maturo potrebbe correggere i due errori presenti nell'esercizio producendo la soluzione mostrata sotto a destra, più efficiente in termini di numero di istruzioni di quella canonica.

Il successivo esercizio 7 non presenta particolari difficoltà (sono sbagliati i parametri delle due istruzioni "ripeti … volte"), mentre nell'**esercizio 8** bisogna fare attenzione a non farsi confondersi dalla necessità di andare avanti per tre volte. Sulla base di questa necessità si potrebbe infatti essere tentati di lasciare i tre blocchi "vai avanti" presenti nella soluzione già pronta nell'area di lavoro, mentre invece il tipo di errore in questo caso è quello di **istruzioni superflue**, come segnalato anche dal numero evidenziato in arancione nella striscia superiore viola.

Basta quindi rimuovere due dei tre blocchi "vai avanti" per ottenere la soluzione corretta.

Nell'**esercizio 9** vi è un'ulteriore difficoltà costituita dalla contemporanea presenza di un'istruzione del tutto sbagliata e di un parametro sbagliato in un'istruzione per altri versi corretta (**istruzione con parametro sbagliato**).



Il parametro del secondo ciclo dovrebbe essere infatti 3 invece di 4, mentre nel primo ciclo al posto dell'istruzione "fai il miele" ci dovrebbe essere "prendi il nettare". Quest'ultimo è un tipico esempio di errori che sono dovuti al fatto che chi scrive il programma, nonostante abbia chiaro in mente cosa debba essere fatto, si trova di fatto (per molti motivi) ad aver indicato un'istruzione scorretta.

Mentre nell'interazione tra persone questo tipo di errori può essere colto facilmente dall'intelligenza umana e dal contesto (per cui le persone capiscono ciò che vogliamo dire nonostante l'errore fattuale nel ciò che abbiamo detto) nel fornire istruzioni ad un computer non possiamo contare su questa rete di sicurezza. La lezione da ricavarne non è ovviamente quella di comportarsi meccanicamente, da calcolatori, ma quella di prestare una particolare attenzione, anche nella comunicazione tra persone finalizzata all'esecuzione di attività, alla precisione e comprensibilità di ciò che si dice.

L'**esercizio 10** presenta una situazione in cui l'ape deve realizzare molte attività. La semplice ispezione del programma che si trova già pronto nell'area di lavoro può non rendere subito evidente dove si annidi l'errore, mentre la sua esecuzione tutto di seguito lo mostrerà chiaramente.

Si tratta di un'**istruzione nel posto sbagliato**, dal momento che è necessario inserire il blocco "fai il miele" all'interno dell'ultimo ciclo con 5 ripetizioni.

L'**esercizio 11** (quello finale) non presenta nuove difficoltà poiché contiene due **istruzioni con parametro sbagliato** (il parametro del primo ciclo e il parametro dell'istruzione "gira a ...").

(questa pagina è intenzionalmente vuota)

4 Versione semplificata del Modulo 1

Questa versione semplificata del Modulo 1 è adatta a studenti che stanno imparando a leggere ed introduce i concetti di base della programmazione informatica. Gli studenti sviluppano le loro capacità di risolvere problemi ed imparano a concentrarsi su problemi che richiedono il loro impegno.

Le attività didattiche del modulo fanno riferimento al materiale del Corso 1 di Code.org, che è stato realizzato proprio per gli studenti che non ancora padroneggiano la lettura Esso è infatti adatto anche agli studenti che stanno imparando a leggere dal momento che i blocchi usati per la costruzione dei programmi sono corredati da simboli e immagini con uso minimale di testo.

Può addirittura essere usato, con l'assistenza del docente, nella scuola dell'infanzia. In esso si usano blocchi direzionali con le frecce e non con i nomi delle direzioni. I blocchi di spostamento indicano il movimento in modo assoluto e non in relazione all'orientamento del soggetto.

Data l'età particolarmente giovane degli studenti cui sono destinate le lezioni di questo modulo, si raccomanda particolare attenzione da parte dell'insegnante nel seguire lo svolgimento delle attività. Poiché infatti gli allievi a questo livello di maturazione non hanno in genere una sufficiente abilità nella lettura, tenderanno ad operare per analogia e non riusciranno quindi a cogliere appropriatamente i passi di astrazione che il percorso presenta. Potrebbe quindi capitare che risolvono comunque gli esercizi, ma non ne acquisiscono i pieni benefici. Si consiglia dunque, se possibile, un approccio basato sulla discussione e svolgimento collettivo degli esercizi.

I concetti fondamentali presentati (sequenza, ripetizione e verifica) sono gli stessi analizzati nel precedente capitolo 3, cui si rimanda per una discussione (vedi pag. 13).

4.1 Prerequisito: interazione con l'ambiente

Come attività preliminare a questo modulo è necessario assicurarsi che tutti gli studenti siano in grado di realizzare l'azione fondamentale di "trascinare e rilasciare" (*drag and drop*, in inglese) ed abbiamo familiarità con gli oggetti ed i comandi dell'interfaccia.

La lezione 3 del corso 1 di Code.org serve a questo scopo. In essa si impara ad usare il mouse per interagire con l'interfaccia dell'ambiente.

Gli studenti iniziano spostando immagini sullo schermo, dalla posizione di partenza a quella di arrivo, e continuando mettendo nell'ordine giusto i blocchi che corrispondono ai pezzi di un puzzle. Negli esercizi finali imparano anche a combinare blocchi inserendoli all'interno di altri blocchi.

Il sequenziamento e la combinazione dei blocchi sono le due operazioni fondamentali con le quali si costruiscono i programmi nell'ambiente utilizzato. La forma dei blocchi fornisce indicazioni su come si possono sequenziare e combinare.

4.2 La sequenza

Per questo argomento si usano le lezioni 4 e 8 del corso 1. Gli studenti sviluppano programmi[12] costituiti da **sequenze** di istruzioni che vanno opportunamente definite in modo che l'esecutore raggiunga il suo obiettivo dato il contesto in cui si trova. Individuare la corretta successione delle istruzioni è quindi il concetto fondamentale che acquisisce lo studente mediante queste attività

Nella lezione 4 l'esecutore è un uccellino del noto gioco Angry Birds[13] il cui obiettivo è raggiungere il maialino verde muovendosi in un labirinto, mentre nella lezione 8 è un Artista il cui scopo è disegnare delle forme geometriche.

Qualora necessario, gli studenti possono svolgere ulteriori esercizi con le lezioni 10 e 11 del corso 1 (vedi paragrafo 4.3.3 a pag.71).

4.2.1 Corso 1: Lezione 4 (Sequenza – Labirinto)

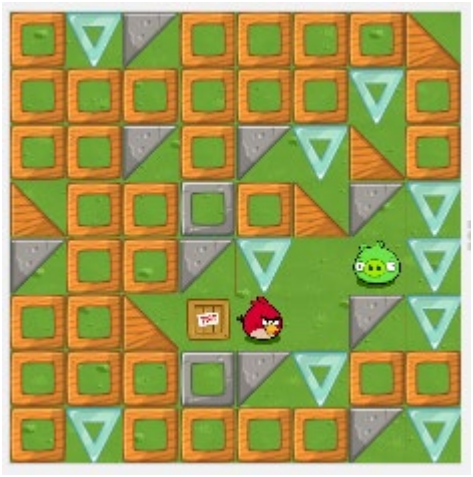
In questa lezione l'esecutore (un uccellino di Angry Birds) ha quattro azioni a disposizione per spostarsi nelle quattro direzioni corrispondenti ai quattro punti cardinali per raggiungere il suo obiettivo di raggiungere il maialino. La specifica del movimento, formulata mediante frecce chiaramente interpretabili ed usando un riferimento assoluto, quindi non relativamente alla posizione dell'esecutore, rende questa lezione adatta anche ai bambini più piccoli.

[12] In casi come questi in cui l'esecutore è un computer si usa preferibilmente il termine "programma" al posto di quello di "procedura".
[13] Angry Birds ® e © 2009-2020 di Rovio Entertainment Ltd. Tutti i diritti riservati.

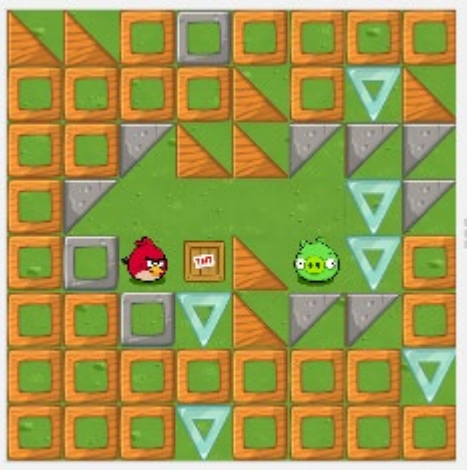
La soluzione dell'**esercizio 1**, quello sopra illustrato, consiste nel semplice programma mostrato qui a fianco e non presenta la possibile ambiguità descritta nel precedente capitolo 3 a pag. 19.

I successivi quattro esercizi (dal 2 al 5) sono spostamenti sempre in una stessa direzione, mentre con l'**esercizio 6** si richiede di cambiare direzione, combinando dunque spostamenti in due direzioni (vedi sotto a destra la soluzione).

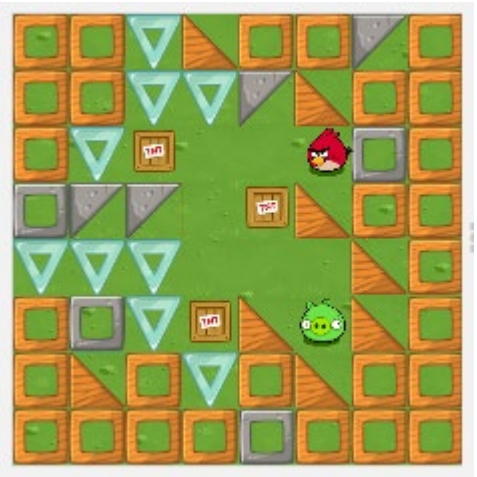
Il successivo esercizio 7 è analogo, mentre nell'**esercizio 8** si cambia direzione due volte (sotto a destra la soluzione).



L'esercizio successivo (9) è analogo al precedente, mentre l'**esercizio 10** richiede di usare tre direzioni differenti (sotto a destra la soluzione).



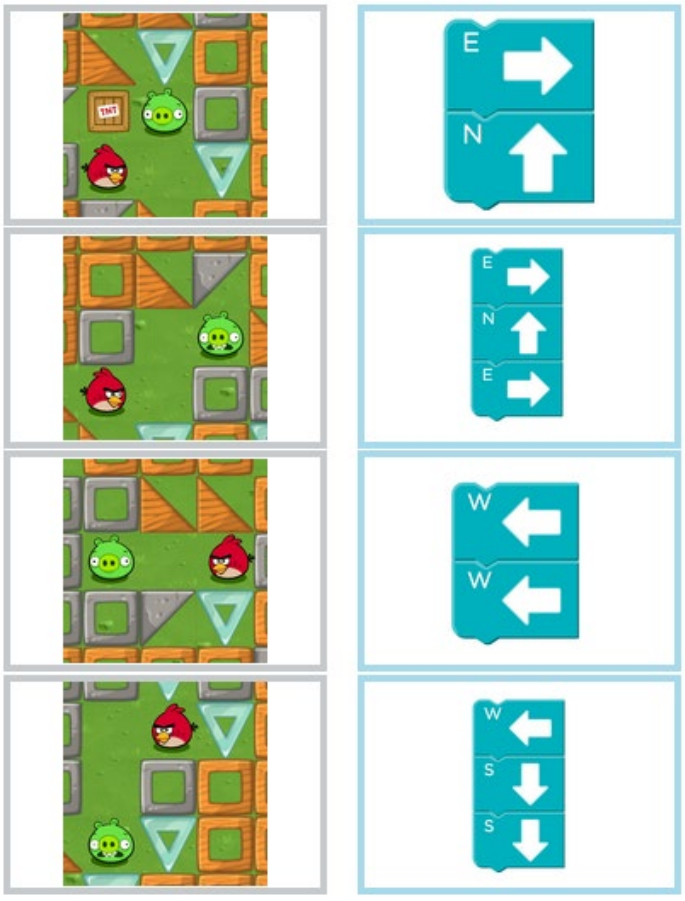
Anche nell'**esercizio 11** si usano tre direzioni differenti, ma si cambia di direzione per tre volte (sotto a destra la soluzione).



I successivi due esercizi (12 e 13) non introducono scenari di tipo differente rispetto a quelli sinora svolti, mentre gli ultimi due esercizi sono esercizi di verifica.

Nell'**esercizio 14** (Risposte Multiple) si tratta di scegliere la sequenza di blocchi corretta (fra le quattro proposte) per il contesto in cui si trova l'uccellino. In questo caso le direzioni sono rappresentate dalle iniziali delle parole inglesi (**N**orth=Nord, **W**est=Ovest, **E**ast=Est, **S**outh=Sud) per cui la soluzione è quella raffigurata sotto a destra.

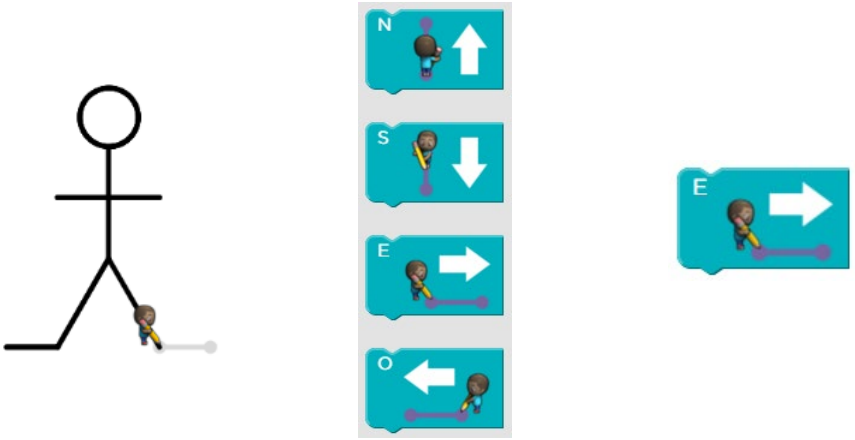
Il secondo esercizio di verifica (**esercizio 15** – Associazioni) richiede di associare ad ogni contesto la corretta sequenza di blocchi che fa raggiungere il maialino. Anche in questo caso le possibili risposte sono programmi con le direzioni rappresentante dalle iniziali dei termini inglesi e la soluzione corretta è la seguente.



4.2.2 Corso 1: Lezione 8 (Sequenza – Artista)

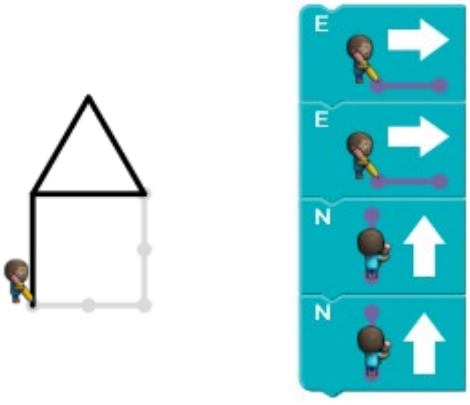
L'ambientazione di questa lezione è quella del disegno di forme geometriche da parte di un artista, che disegna mentre si sposta nelle quattro direzioni corrispondenti ai quattro punti cardinali. Anche in questo caso la specifica del movimento mediante frecce ed usando un riferimento assoluto rende la lezione adatta anche ai bambini più piccoli.

L'**esercizio 1** propone una figura che va completata disegnandone il piede (vedi sotto a sinistra), usando le istruzioni disponibili (sotto al centro). La soluzione è quella mostrata sotto a destra.

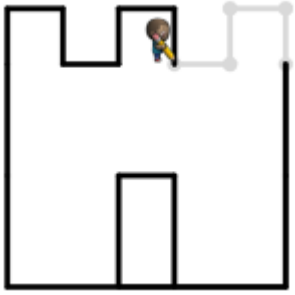


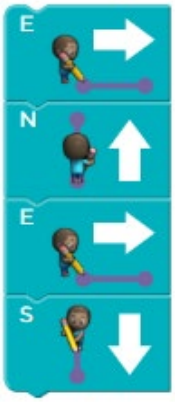
Si noti che, diversamente dalle lezioni con l'artista del corso 2 (e di quelli successivi), nelle lezione del corso 1 che usano questo esecutore non vengono utilizzati i gradi: esse sono pertanto pienamente fruibili anche da parte dei bambini più piccoli.

Il successivo esercizio 2 prevede ancora spostamenti sempre nella stessa direzione, mentre nell'**esercizio 3** per completare il disegno della casa che viene proposto vanno combinati movimenti in due direzioni (sotto a destra la soluzione).

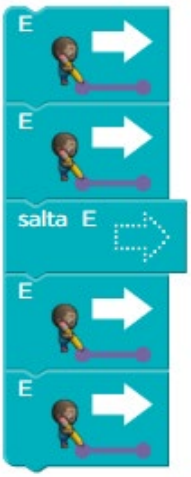


Nell'**esercizio 4** il completamento della figura richiede l'uso di tre direzioni diverse e di cambiare direzione tre volte (sotto a destra la soluzione).





Mentre il successivo esercizio 5 ripete un contesto già incontrato, con l'**esercizio 6** il repertorio delle istruzioni disponibili per l'esecutore (l'artista) si estende con due nuove azioni, che gli permettono di saltare verso Est o verso Ovest (vedi sotto al centro) e di completare il suo disegno come mostrato sotto a destra.

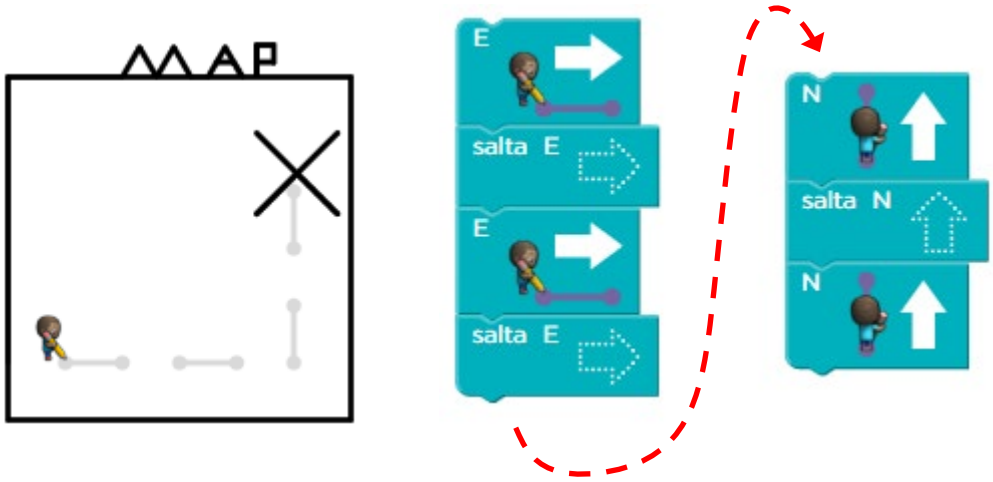


Questo esercizio offre l'occasione per un'interessante riflessione di natura più generale su come la "potenza operativa" dell'esecutore sia strettamente dipendente dalle azioni che è in grado di compiere. Se l'artista non avesse a disposizione l'istruzione per saltare non sarebbe in grado di completare il disegno in modo del tutto corretto rispetto a quanto richiesto[14]. Questo è appunto una caratteristica di natura generale di tutti gli "esecutori automatici". Il loro "potere computazionale" è legato all'insieme di istruzioni che hanno a disposizione.

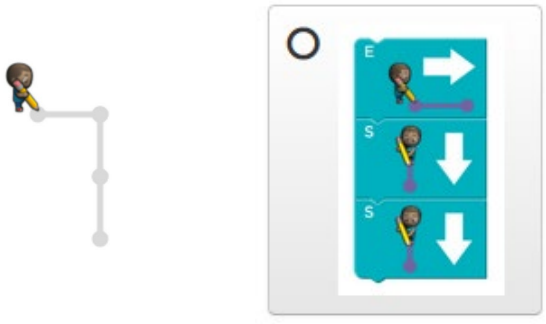
[14] È possibile fornire una soluzione "parzialmente corretta" semplicemente spostando l'artista per cinque volte verso Est.

Va anche chiarito sul piano teorico che tutti i computer esistenti hanno, ad un certo livello di astrazione, lo stesso potere computazionale e si differenziano soltanto per la maggiore o minore velocità di esecuzione delle istruzioni. Questo vuol dire che tutti i computer esistenti hanno la possibilità di risolvere gli stessi problemi. Il fatto che questo non sia vero in pratica deriva da limiti temporali (potrebbero essere necessari secoli) o tecnologici (impossibilità di memorizzare tutti i dati necessari).

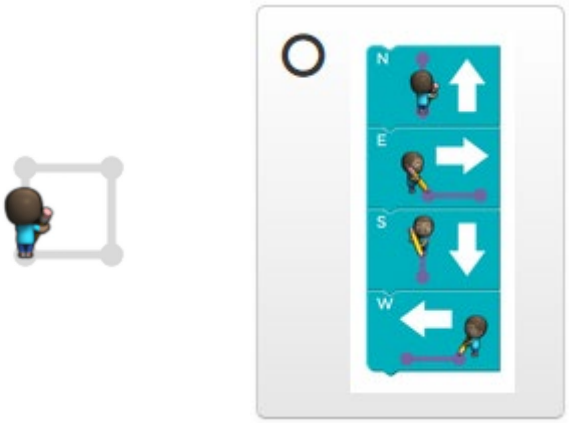
L'esercizio successivo (7) è ancora una combinazione di spostamenti e salti sempre nella stessa direzione, mentre nell'**esercizio 8** (in cui sono disponibili altre due istruzioni di salto, verso Nord e verso Sud) è necessario combinare movimenti e salti in due direzioni (sotto a destra la soluzione, raffigurata per motivi di spazio su due colonne).



Mentre i successivi due esercizi (9 e 10) presentano situazioni dei tipi già incontrati, gli ultimi due esercizi sono esercizi di verifica. Nel primo (**esercizio 11** – Risposte Multiple) si tratta di scegliere la sequenza di blocchi corretta per il disegno che deve completare l'artista. Anche in questo caso le direzioni sono rappresentate dalle iniziali delle parole inglesi (**N**orth=Nord, **W**est=Ovest, **E**ast=Est, **S**outh=Sud) per cui la soluzione è quella raffigurata sotto a destra.



Anche nel secondo esercizio di verifica (**esercizio 12** – Risposte Multiple) bisogna scegliere la sequenza di blocchi corretta per il disegno che deve completare l'artista. La soluzione è quella raffigurata sotto a destra.



4.2.3 Ulteriori lezioni sulla sequenza

Per svolgere esercizi aggiuntivi sul concetto di sequenza sono disponibili ulteriori lezioni del Corso 1.

Nella lezione 10 il protagonista è di nuovo l'Artista, che affronta la realizzazione di nuovi disegni. Nell'ultimo esercizio (10) di questa lezione lo studente è libero di disegnare qualunque cosa voglia.

Nella lezione 11 ritorna l'Ape, ma questa volta con un'ambientazione nella quale si muove su una griglia in cui ogni cella contiene una lettera. L'obiettivo di ogni esercizio è per trovare una dopo l'altra tutte le lettore che compongono delle parole date.

4.3 La ripetizione

Il concetto fondamentale che viene introdotto è quello della **ripetizione**, cioè di un'istruzione che, in modo sintetico, permette di far ripetere all'esecutore una o più azioni per un certo numero di volte. L'istruzione di ripetizione viene anche detta **ciclo**. Per lavorare su questo concetto si usano le lezioni 13 e 14 del Corso 1.

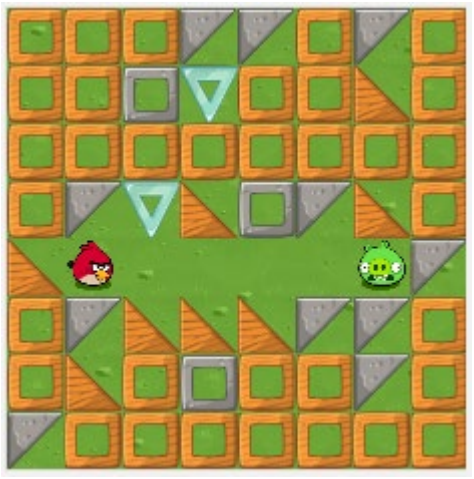
Gli studenti imparano ad usare i cicli in modo via via più complesso per guidare nella lezione 13 l'uccellino attraverso il Labirinto verso la sua mèta e nella lezione 14 per guidare l'Ape a raccogliere nettare dai fiori e produrre miele nei favi.

Qualora necessario, gli studenti possono svolgere ulteriori esercizi con la lezione 18 del corso 1 (vedi paragrafo 4.3.3 a pag. 71).

4.3.1 Corso 1: Lezione 13 (Ripetizione – Labirinto)

In questa lezione l'ambientazione è quella già incontrata dell'uccellino di Angry Birds che deve spostarsi nel Labirinto per raggiungere il suo obiettivo, costituito dal maialino.

Dapprima si ripetono singole azioni (ripetizione semplice – esercizi 2 e 3), poi si mettono in sequenza più ripetizioni (esercizi 4, 5, 6 e 7), infine si ripetono sequenze di azioni (ripetizione composta – esercizi 8 e 9, 10).

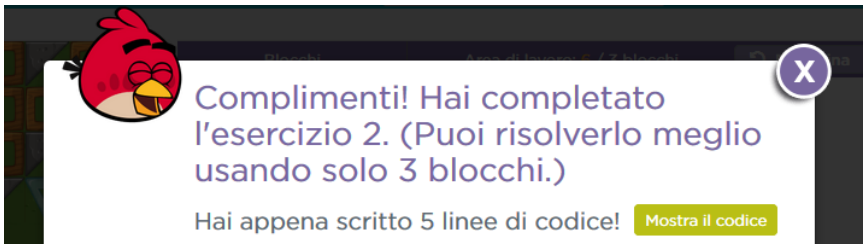


Mentre l'esercizio 1 propone allo studente una situazione già incontrata nel caso di sequenze di istruzioni, nell'**esercizio 2**, raffigurato a fianco, si chiede allo studente di usare un nuovo blocco per far arrivare l'uccellino fino al maialino.

Adesso è infatti disponibile un'ulteriore istruzione per guidare l'uccellino (cioè l'esecutore è in grado di realizzare una nuova azione) e tale istruzione (mostrata qui a fianco) permette di ripetere per 5 volte le istruzioni che verranno inserite al suo interno.

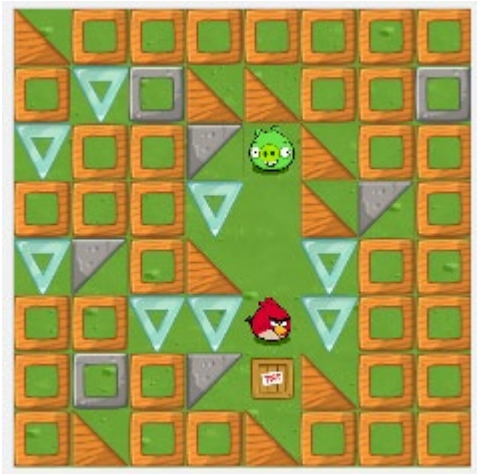
Lo studente dovrebbe quindi arrivare a comprendere che la soluzione all'esercizio è quella consistente nel ripetere per 5 volte il blocco di spostamento verso Est, come illustrato qui a fianco.

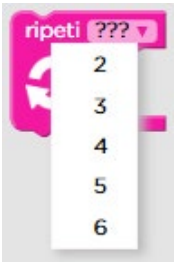
È importante notare che la presenza dell'insegnante è fondamentale, in casi di questo genere, per assicurarsi che lo studente comprenda qual è l'obiettivo dell'esercizio e lavori su di esso. Se infatti lo studente affronta questo esercizio come il precedente e quindi colloca nell'area di lavoro cinque blocchi di spostamento verso Est e clicca su "Esegui" sulla soluzione, ciò lo conduce comunque a risolvere l'esercizio, anche se in modo imperfetto, come viene segnalato dal sistema col messaggio sotto raffigurato.



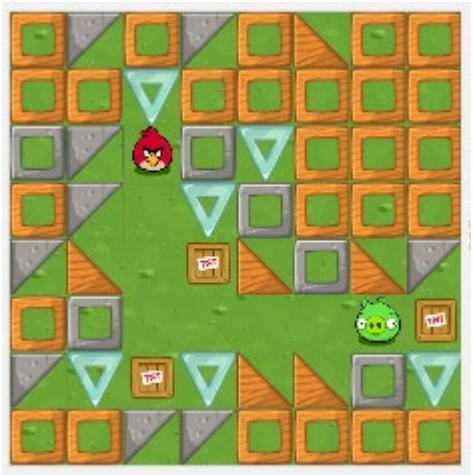
C'è dunque il rischio che, in un'ottica da videogioco, lo studente consideri di aver "completato il livello" e proceda avanti ignaro. L'intervento dell'educatore è in tali casi necessario affinché l'attività formativa sia efficace. Questo è vero, a maggior ragione, considerando la giovane età degli studenti che affrontano questa versione del modulo 1.

Nell'**esercizio 3** vi è una situazione analoga, tranne per il fatto che nel nuovo blocco di ripetizione non si trova già scritto il numero di volte che devono essere ripetute le istruzioni che verranno inserite al suo interno. Il numero deve essere scelto dallo studente, cliccando sul triangolino con la punta in basso accanto ai tre punti interrogativi, che apre una finestrella nella quale si può scegliere il numero desiderato (vedi sotto al centro). La soluzione allora è quella mostrata sotto a destra.

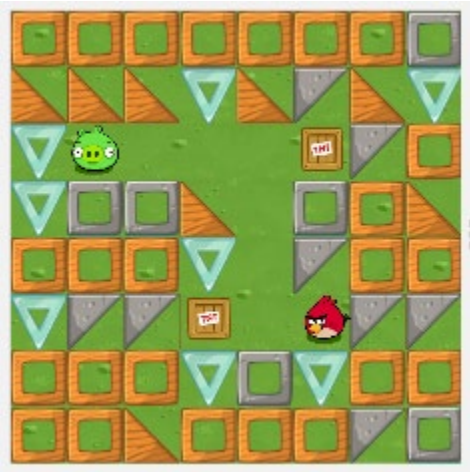




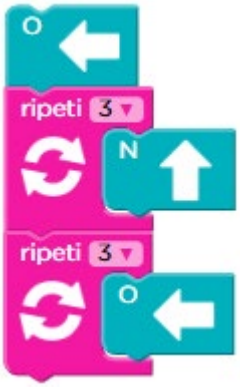
Nell'**esercizio 4** si combina il concetto di sequenza e quello di ripetizione. Nello scenario che si presenta allo studente è infatti necessario usare in sequenza la ripetizione che permette all'uccellino di muoversi verso il basso e la ripetizione per muoversi verso destra. Questo è già predisposto nell'area di lavoro (vedi in basso a destra).

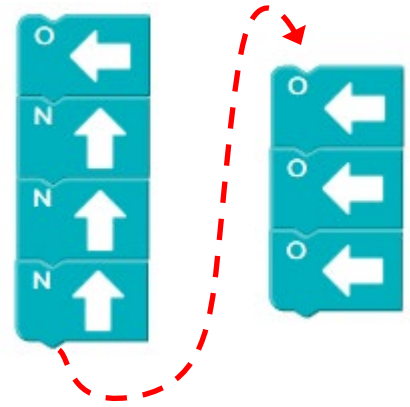


Lo studente deve soltanto inserire all'interno dei blocchi di ripetizione gli opportuni blocchi di spostamento: verso Sud per la ripetizione da 3 volte e verso Est per la ripetizione da 4 volte.



L'**esercizio 5** è concettualmente analogo al precedente, ma in questo caso (vedi qui a sinistra) non vi sono blocchi già pronti nell'area di lavoro e devono essere scelti dallo studente. La soluzione è dunque quella mostrata qui a destra.

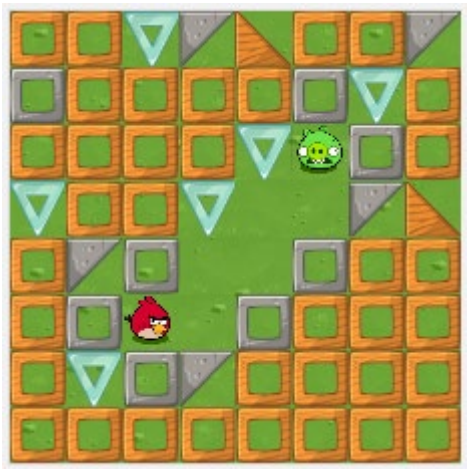


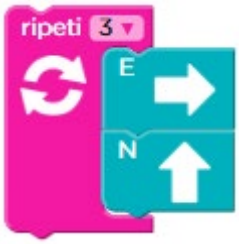


Osserviamo che uno studente che non presta molta attenzione alla specifica dell'esercizio potrebbe in questo definire una soluzione (corretta ma inefficiente come numero di istruzioni) del tipo mostrato a fianco (raffigurata per motivi di spazio su due colonne).

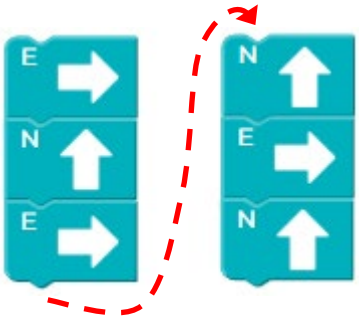
Come precedentemente notato, l'intervento dell'insegnante è fondamentale per assicurare in tali casi efficacia all'attività didattica.

Gli esercizi 6 e 7 insistono su scenari dello stesso tipo, mentre nell'**esercizio 8** (sotto a sinistra) viene affrontato un nuovo modo di combinare sequenza e ripetizione. L'esercizio che viene proposto è il seguente e la sua soluzione (mostrata sotto a destra) richiede di usare una ripetizione non più di singole azioni ma di sequenze di azioni.



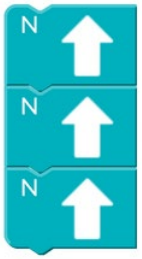


Il superamento di questo passaggio (dalla ripetizione di un'istruzione alla ripetizione di una sequenza di istruzioni) richiede un passo di astrazione che permette di considerare una sequenza di istruzioni niente altro che un'istruzione essa stessa. Non si tratta di qualcosa di così immediato per studenti quali quelli che affrontano questo modulo. Potrebbe essere quindi opportuno che l'insegnante proponga dapprima la soluzione (corretta ma imperfetta) consistente nel ripetere esplicitamente tutti i blocchi necessari per raggiungere il maialino, che è raffigurata a fianco, e poi stimoli i suoi allievi a cercare la soluzione che usa il blocco di ripetizione per risolvere l'esercizio usando complessivamente meno blocchi.



Gli esercizi 9 e 10 lavorano sempre sulla stessa problematica della ripetizione di sequenze di istruzioni. Gli esercizi 11 e 12 esercitano scenari già visti, mentre gli ultimi due esercizi sono di verifica.

Nel primo (**esercizio 13** – Risposte Multiple) bisogna scegliere la sequenza di blocchi che usa il blocco di ripetizione ed è equivalente alla sequenza di blocchi mostrata sotto a sinistra. La soluzione corretta è raffigurata sotto a destra.

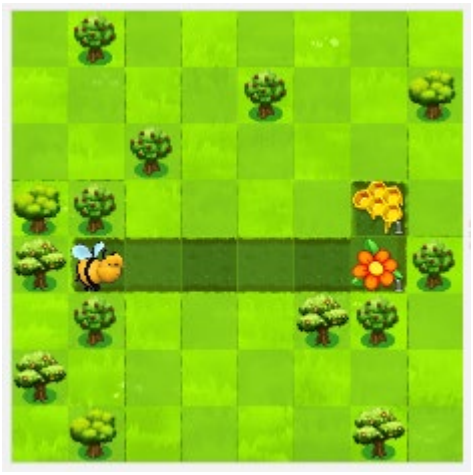


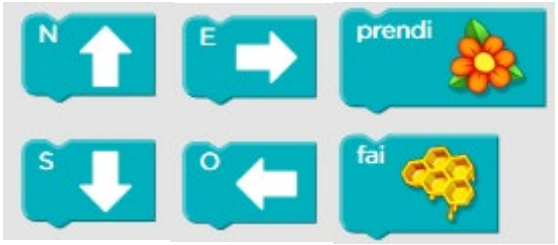
Anche nel secondo esercizio di verifica (**esercizio 14** – Risposte Multiple) si tratta di scegliere la sequenza di blocchi che usa il blocco di ripetizione ed è equivalente alla sequenza di blocchi mostrata sotto a sinistra. La soluzione corretta è raffigurata sotto a destra (il termine "ripeti" è rappresentato nella versione inglese "repeat").

4.3.2 Corso 1: Lezione 14 (Ripetizione – Ape)

Anche in questa lezione si ripetono dapprima singole azioni (ripetizione semplice – esercizio 2), poi si mettono in sequenza più ripetizioni (esercizi 3 e 5), successivamente si ripetono sequenze di azioni (esercizi, 4, 6 e 7), infine si ripetono sequenze di azioni (ripetizione composta – esercizi 8, 9 e 10).

L'**esercizio 1** ha il solo scopo di presentare allo studente la nuova ambientazione, che è quella di un prato nel quale l'esecutore (un'Ape) può spostarsi mediante azioni che la fanno muovere nelle quattro direzioni corrispondenti ai quattro punti cardinali. Col suo movimento deve raggiungere fiori da cui raccogliere il nettare e favi nei quali produrre il miele. Le azioni a sua disposizione (raffigurate sotto a destra) includono quindi, oltre a quelle di movimento, due azioni che realizzano le attività, appunto, di raccolta del nettare ("prendi") e produzione del miele ("fai").

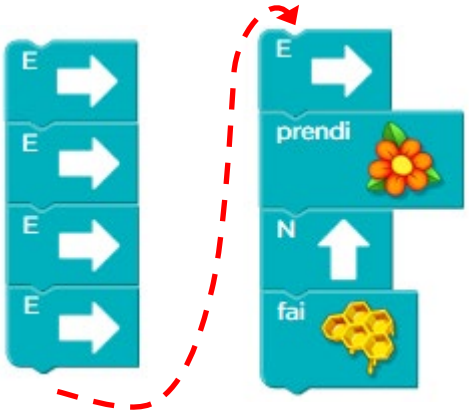




Queste ultime due sono però azioni che ottengono il loro risultato soltanto nel contesto opportuno. Più precisamente, quando l'Ape si trova su di un fiore che ha del nettare (le dosi disponibili sono segnalate da un numero in basso a destra nella casella che contiene il fiore) l'azione "prendi" ha il risultato di prelevare dal fiore una dose di nettare (decrementando quindi di 1 il relativo numero).

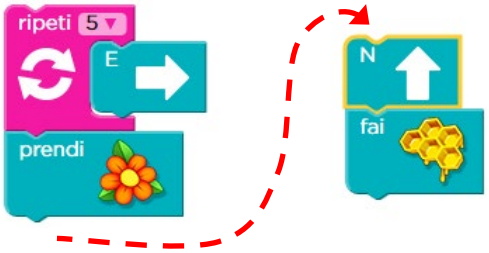
Analogamente, quando l'Ape si trova su di un favo che deve essere riempito di miele (le dosi da produrre sono segnalate come per il caso del nettare del fiore) l'azione "fai" ha il risultato di depositare una dose di miele nel favo (decrementando quindi di 1 il relativo numero)

In questo caso la soluzione è la semplice sequenza di blocchi (mostrata a fianco) che spostano l'ape sul fiore, le fanno raccogliere il miele, la spostano sul favo e le fanno produrre il miele.

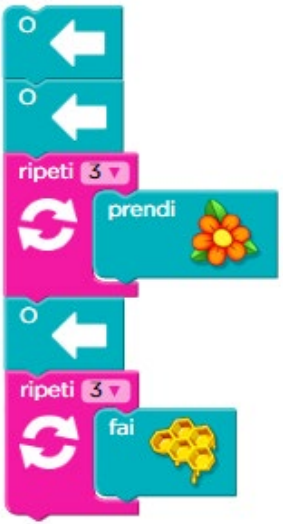


Nell'**esercizio 2** lo studente si trova esattamente nello stesso scenario ma ha a disposizione anche l'istruzione "ripeti" – già incontrata nella precedente lezione "Corso 1: Lezione 13 (Ripetizione – Labirinto)", e gli viene chiesto di usarla per scrivere il programma che fa raggiungere all'ape il suo obiettivo.

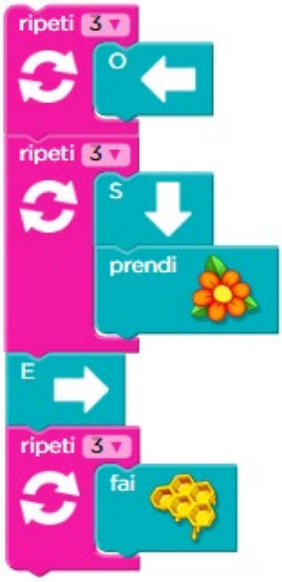
Non dovrebbe questo punto essere difficile, dopo aver svolto la precedente lezione sul concetto di ripetizione (paragrafo 4.3.1, pag. 62), individuare la soluzione, illustrata qui a fianco.



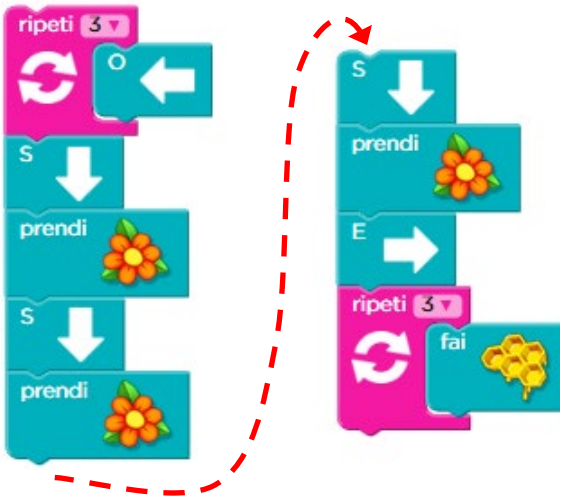
Nell'**esercizio 3** la situazione si evolve, dal momento che sul fiore vi sono più dosi di nettare da raccogliere ed analogamente vi sono più dosi di miele da produrre nel favo. Dovranno quindi essere messe in sequenza più ripetizioni, come mostrato nella soluzione sotto a destra.



L'**esercizio 4** offre un'ulteriore evoluzione, poiché l'ape deve ripetere non più singole azioni, ma sequenze di azioni. Questa ulteriore complicazione concettuale è stata già affrontata nell'esercizio 8 della precedente lezione "Corso 1: Lezione 13 (Ripetizione – Labirinto)" (vedi pag. 65). Si osservi infatti che i tre fiori ognuno con una dose di nettare possono essere trattati all'interno di un singolo ciclo, come illustrato dalla soluzione sotto a destra, nella quale si può notare la sequenza presente all'interno del secondo ciclo.



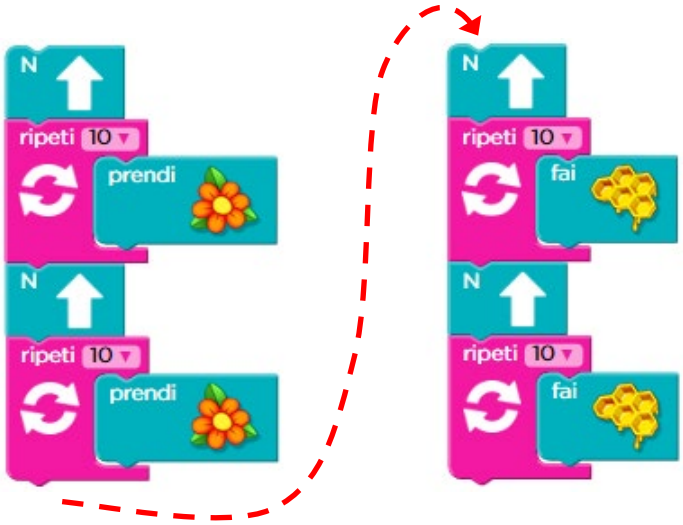
Come già osservato precedentemente, il superamento di questo passaggio non è così immediato per gli studenti cui questo modulo è indirizzato. Il passo di astrazione che permette di considerare una sequenza di istruzioni niente altro che un'istruzione essa stessa non è del tutto ovvio. Può essere utile che l'insegnante proponga dapprima la soluzione (corretta ma imperfetta) consistente nel ripetere esplicitamente tutti i blocchi necessari per raccogliere il nettare, come a fianco raffigurato, e poi stimoli i suoi allievi a cercare la soluzione che usa il blocco di ripetizione in modo da risolvere l'esercizio usando complessivamente meno blocchi.



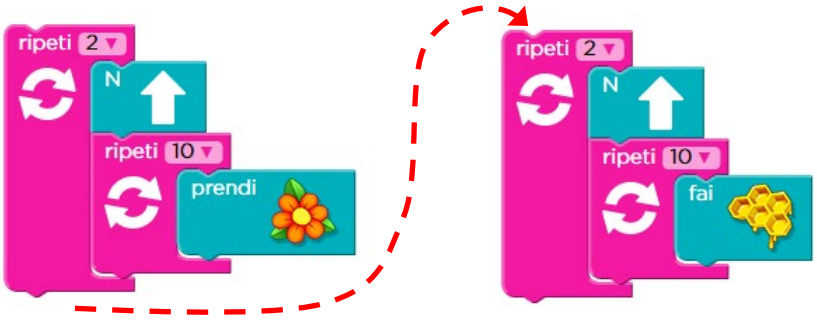
Nell'esercizio seguente (5) si affrontano ancora le sequenze di ripetizioni, mentre nei quattro ancora successivi (6, 7, 8 e 9) vi sono di nuovo ripetizioni di sequenze (in al-

cuni casi insieme a sequenze di ripetizioni), analogamente a quanto presente nell'esercizio sopra discusso (e si applicano le stesse osservazioni).

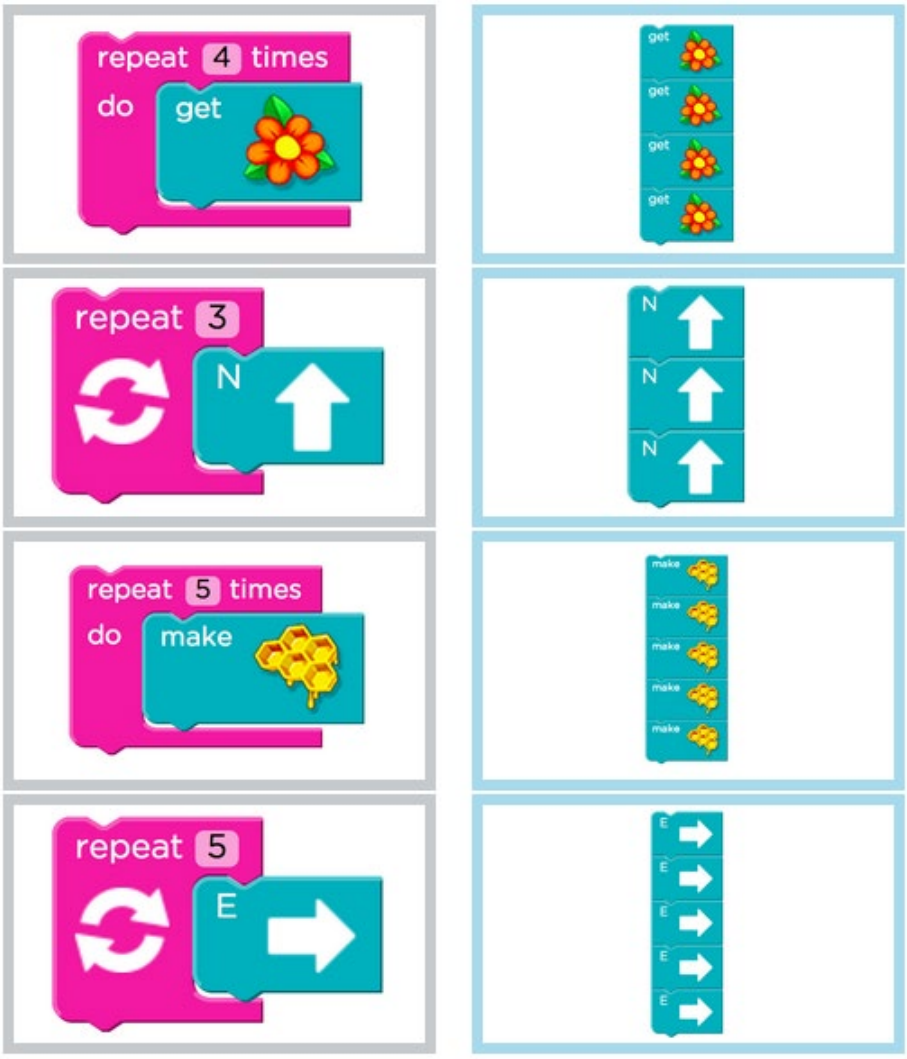
L'**esercizio 10** affronta di nuovo le sequenze di ripetizioni, e presentiamo la sua soluzione canonica, raffigurata qua sotto,



solo per far notare che uno studente particolarmente acuto potrebbe individuare un suo possibile miglioramento, se osserva che – sia nella fase di raccolta del nettare che di produzione del miele – vi è la ripetizione per due volte di una stessa sequenza di istruzioni ed è quindi possibile ottenere la soluzione sotto mostrata, che usa meno blocchi della soluzione canonica.



I due esercizi successivi (11 e 12) sono di riepilogo di problematiche già affrontate mentre l'ultimo, **esercizio 13** – Associazioni, è un esercizio di verifica. Esso richiede di associare ad ogni ciclo (che contiene al suo interno una sola istruzione) la sequenza esplicita di blocchi che esegue le stesse azioni. La corretta associazione è rappresentata qua sotto: notare che nelle figure si usano i termini inglesi: "repeat" per "ripeti" e "repeat … times" per "ripeti … volte".



4.3.3 Ulteriori lezioni sulla ripetizione

Per svolgere esercizi aggiuntivi sul concetto di ripetizione è disponibile un'ulteriore lezione del Corso 1.

Si può infatti usare la lezione 18, in cui l'esecutore è nuovamente l'Artista già incontrato per lavorare sul concetto di sequenza nel paragrafo 4.2.2 a pag. 58. In questa lezione 18, avendo a sua disposizione anche il blocco di ripetizione, potrà sviluppare i suoi disegni sullo schermo in modo più efficiente.

Anche in questa lezione si raccomanda l'attenzione dell'insegnante a come gli alunni svolgono gli esercizi, dal momento che gli esercizi sono chiaramente risolvibili senza usare il blocco di ripetizione. Così facendo non si ottiene però efficacia didattica.

4.4 La verifica

La verifica, con conseguente correzione degli errori, è un elemento fondamentale dell'apprendimento della programmazione dal momento che lo studente è condotto a riflettere su cosa compie ogni singola istruzione e sul fatto che la loro sequenza conduca o meno al risultato desiderato. A questo scopo lo studente si mette, in un certo senso, nei panni del computer, e questo esercizio lo conduce ad una migliore comprensione dei concetti.

Nelle lezioni dedicate a questo concetto gli studenti si trovano di fronte esercizi che sono stati svolti in modo non corretto. Devono quindi esaminare il codice esistente per identificare gli errori, che possono essere costituiti da:

- blocchi mancanti,
- blocchi superflui,
- blocchi collocati nell'ordine sbagliato,
- blocchi sbagliati rispetto all'obiettivo dell'esercizio.

Per lavorare su questo aspetto in questa versione semplificata del Modulo 1 si svolge solo la lezione "Corso 1: Lezione 5 (Verifica – Labirinto)", descritta nel paragrafo 3.3.1 a pag. 39, cui si rimanda. L'altra lezione sulla verifica (quella descritta nella sezione 3.3.2 a pag. 44) non è infatti adatta agli studenti cui è dedicata questa versione semplificata del Modulo 1.

5 Modulo 2: istruzione condizionale

Il nuovo concetto fondamentale introdotto in questo modulo è quello di **istruzione condizionale**. Mentre nel precedente modulo 1 tutti i programmi conducevano all'esecuzione di una stessa sequenza lineare di istruzioni, in questo modulo gli studenti imparano a scrivere programmi che prendono decisioni e possono quindi eseguire differenti sequenze di istruzioni.

Il programma scritto dallo studente non è più quindi lineare ma ramificato, ed i diversi rami corrispondono alle diverse possibilità incontrate dal programma nel corso della sua esecuzione.

Gli studenti imparano a combinare questo concetto con quelli di sequenza e di ripetizione.

Nel riprendere il concetto di ripetizione si lavora ulteriormente sul concetto di **efficienza** del codice, ovvero di quantità di istruzioni usate per ottenere lo scopo desiderato.

A partire da questo modulo è necessario che gli studenti siano in grado di leggere. Tutte le lezioni relative ai concetti introdotti da ora in avanti fanno infatti ampiamente uso di materiale testuale.

Inoltre, poiché le lezioni di ripasso prevedono l'utilizzo degli angoli, e nella scuola italiana gli studenti iniziano a lavorare sul concetto di angolo in terza elementare, nel caso questo modulo sia affrontato in seconda elementare il docente dovrà valutare se comunque introdurre informalmente tale concetto, nel contesto operativo degli esercizi svolti, oppure se saltare le relative lezioni ed effettuare il ripasso in altro modo (ad esempio con le lezioni aggiuntive indicate per la sequenza e la ripetizione nella versione semplificata del Modulo 1).

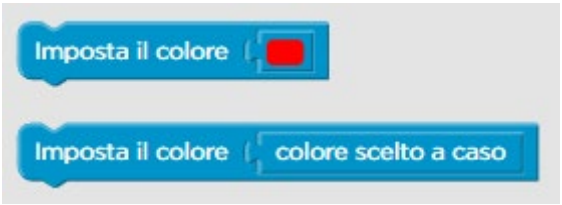
Infine, in quei casi in cui questo modulo viene usato come punto di inizio di un percorso di formazione alla programmazione informatica, la prima sezione di ripasso sarà invece usata per acquisire i concetti di sequenza e ripetizione e per lavorare sul tema della verifica.

5.1 Ripasso di sequenza, ripetizione e verifica

5.1.1 Corso 2: Lezione 4 (Sequenza – Artista)

Il concetto di sequenza è stato introdotto e discusso nella sezione 3.1 a pag. 13, cui si rimanda per ulteriori approfondimenti.

Il contesto della lezione 4 è quello del disegno di forme geometriche da parte di un artista, che ha a disposizione le seguenti istruzioni



Con il comando "vai" (che adesso ha l'opzione per andare all'indietro: cliccare sul triangolino con la punta verso il basso accanto ad "avanti" per selezionare l'opzione "indietro") l'artista disegna un tratto colorato mentre si sposta. Il comando "salta", invece, fa spostare l'artista come "vai" ma senza disegnare. Entrambe queste istruzioni hanno anche una parte variabile che serve a quantificare l'entità dello spostamento.

Nelle lezioni svolte nel Modulo 1, infatti, ogni spostamento era sempre di una sola casella. Adesso invece l'ammontare dello spostamento viene specificato come parte dell'istruzione. Lo spostamento è misurato in *pixel*, un'unità di misura comunemente usata in connessione con gli schermi dei dispositivi informatici.

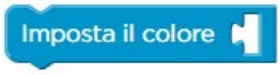
Il pixel si usa per misurare la capacità di rappresentazione degli schermi di computer, tablet e smartphone: poiché il pixel è l'oggetto più elementare che può essere raffigurato su uno schermo (si tratta infatti di un unico puntino) le dimensioni in pixel di uno schermo indicano la sua capacità di rappresentazione. Lo schermo di uno smartphone del 2015 ha un'altezza indicativa compresa tra 1.000 e 2.000 pixel ed una larghezza tra 600 e 1.000 pixel.

Per cambiare il valore preimpostato per l'entità dello spostamento in pixel, basta cliccare sul numero preesistente, scrivere il valore desiderato e premere il tasto "Invio" ("Return" o "Enter" sulle tastiere inglesi).

Anche in questo caso le indicazioni "destra" e "sinistra" per l'istruzione "gira" sono relative all'orientamento dell'artista. Analogamente a quanto accade per lo spostamento, l'unità di misura della rotazione non è più sempre stabilita in una casella ma viene specificata come parte dell'istruzione. In questo caso l'unità di misura usata per gli angoli è il *grado*. A seconda del livello di maturazione dello studente, il concetto può essere già noto o meno. I tipi di angoli usati in queste lezioni sono comunque pochi: angoli acuti (da 45 e 60 gradi), angoli retti (90 gradi), angoli ottusi (120 gradi) e angoli piatti (180 gradi).

In ogni esercizio di questa lezione l'obiettivo è completare e costruire la figura indicata dalle linee in grigio. Nel caso della figura dell'**esercizio 1** (vedi sotto a sinistra) l'obiettivo è completare il quadrato disegnando il lato superiore ed il lato destro. Dal momento che l'artista è già orientato verso destra la soluzione è quella mostrata sotto a destra.

Notiamo infine che l'artista può cambiare il colore del tratto con cui disegna quando si sposta con l'istruzione "vai". Un blocco permette di scegliere il colore rosso, mentre l'altro fa impostare un colore che il programma sceglie a caso ad ogni esecuzione (può quindi essere differente ogni volta che si usa il blocco). Se si rimuove l'opzione di scelta (cioè si rimuove il blocchettino contenente il colore rosso o quello con la scritta "colore scelto a caso"), ottenendo quanto mostrato a fianco, l'artista ritorna a disegnare con un tratto di colore nero.



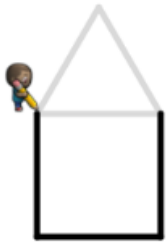
In tutti gli esercizi di questa lezione (ed in molte lezioni successive) sotto il quadrato che raffigura il contesto operativo per l'esecutore compare il regolatore di velocità mostrato a fianco.

Spostando l'indicatore verso sinistra, cioè verso l'icona della tartaruga, l'esecuzione delle istruzioni avverrà più lentamente, mentre spostandolo verso destra (la lepre) il programma verrà svolto più velocemente. Quando ci sono molte istruzioni da eseguire, avere la possibilità di accelerare è importante per non sprecare tempo. D'altro canto, quando non si capisce bene cosa il programma stia effettivamente facendo durante la sua esecuzione, poter rallentare è essenziale per avere il tempo di seguire bene la sequenza dei passi svolti.

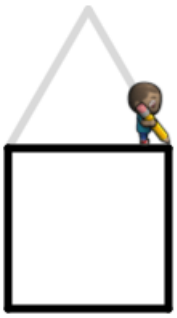
Negli esercizi successivi (2, 3, 4, 5) ci si esercita con gli stessi comandi a disegnare figure via via più complesse – fino ad arrivare ad un quadrato, usando tratti e salti di diverse lunghezze (i cui valori devono in alcuni casi essere calcolati come parte dell'esercizio) ma con rotazioni sempre di 90 gradi.

Nell'**esercizio 6** lo studente deve disegnare un triangolo (vedi sotto a sinistra), che il testo dell'esercizio indica come equilatero. Gli studenti che non hanno ancora affrontato argomenti geometrici potrebbero avere qualche difficoltà nel dedurre da questa indicazione il valore di cui è necessario girare ad ogni rotazione. Anche in questo caso si suggerisce di lasciar loro il tempo di rifletterci e fare da soli dei tentativi. Successivamente all'individuazione della soluzione (raffigurata sotto a destra), si può

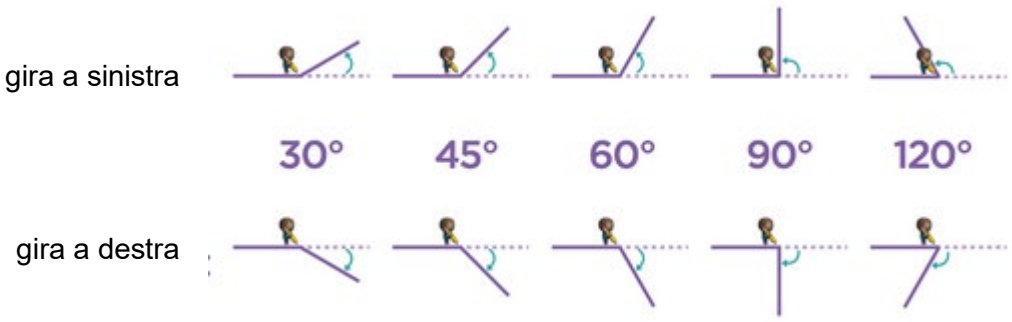
prendere spunto da questo esercizio per osservare che l'essere equilatero implica, per un triangolo ma non in generale, avere tutti gli angoli uguali.



È interessante osservare che il valore usato nella soluzione è quello di 120 gradi, mentre è ben noto che gli angoli di un triangolo equilatero sono di 60 gradi. Questo è dovuto al fatto che l'istruzione "gira" fa riferimento alla direzione corrente dell'artista. Dopo aver disegnato il primo tratto l'artista non è orientato verso l'interno del triangolo ma verso l'esterno (vedi figura a fianco) e deve quindi girare di 120 gradi per poter disegnare correttamente il secondo tratto.



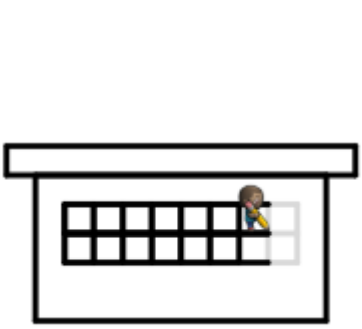
Particolare attenzione va fatta in questi casi al fatto che i gradi di cui girare si calcolano rispetto alla direzione in cui si sta muovendo l'artista. Si veda nella figura qua sotto (che rappresenta diversi casi per le possibili svolte sia a sinistra che a destra dell'artista) il caso, ad esempio, di una svolta a sinistra di 120 gradi che corrisponde a quanto l'artista deve fare in questo esercizio per disegnare correttamente il secondo lato del triangolo dopo aver completato il primo.



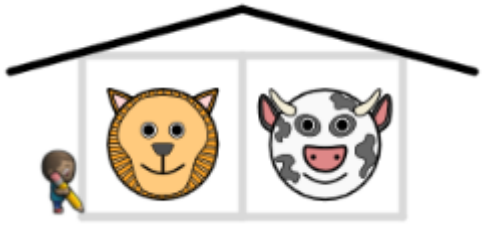
Può essere interessante (ed un utilissimo esercizio in termini di ragionamento spaziale e riflessioni geometriche) stimolare studenti più maturi a provare ad individuare una soluzione che usa solo rotazioni di 60 gradi. Non è del tutto intuitiva ma può valere la pena. Una soluzione (ottima in termini di numero di blocchi usati) è mostrata

sotto a sinistra mentre un'altra, ugualmente corretta ma che usa un blocco in più e disegna il triangolo in senso orario anziché antiorario, è sotto a destra.

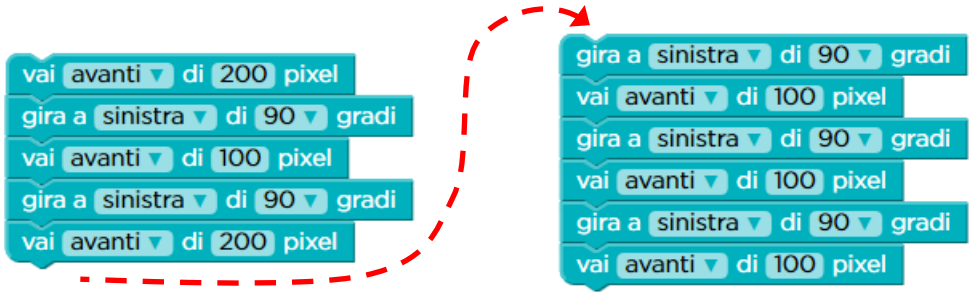
L'esercizio successivo (7) prevede ancora il disegno di un quadrato, con la necessità di fare attenzione – a causa dell'orientamento dell'artista – al disegno del primo tratto. Poi nell'**esercizio 8** vi è da completare una casa col disegno delle ultime due finestre costituite da due quadrati affiancati (vedi sotto a sinistra). Lo studente dovrebbe accorgersi abbastanza facilmente che l'ultimo tratto del primo quadrato ed il primo tratto del secondo quadrato possono essere disegnati da un'unica istruzione, come evidenziato in rosso nella soluzione mostrata sotto a destra.



L'**esercizio 9** in un certo senso è analogo, dal momento che richiede ancora il disegno di due quadrati affiancati, ma l'artista si trova in una differente posizione di partenza (vedi sotto a sinistra). La sua soluzione può essere affrontata seguendo lo stesso approccio di disegnare un tratto più lungo per finire il primo quadrato ed iniziare il secondo, come evidenziato in rosso sotto a destra.

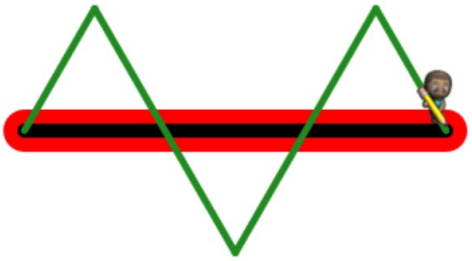


Tale soluzione è quella canonica che il sistema si aspetta, ma uno studente particolarmente riflessivo potrebbe intuire che esiste una soluzione che usa meno blocchi, basata sul disegnare insieme i tratti orizzontali dei due quadrati, come mostrato sotto.



Osserviamo che in questa soluzione con meno blocchi la penultima istruzione di disegno ("vai avanti") può essere sostituita da un istruzione di salto ("salta avanti").

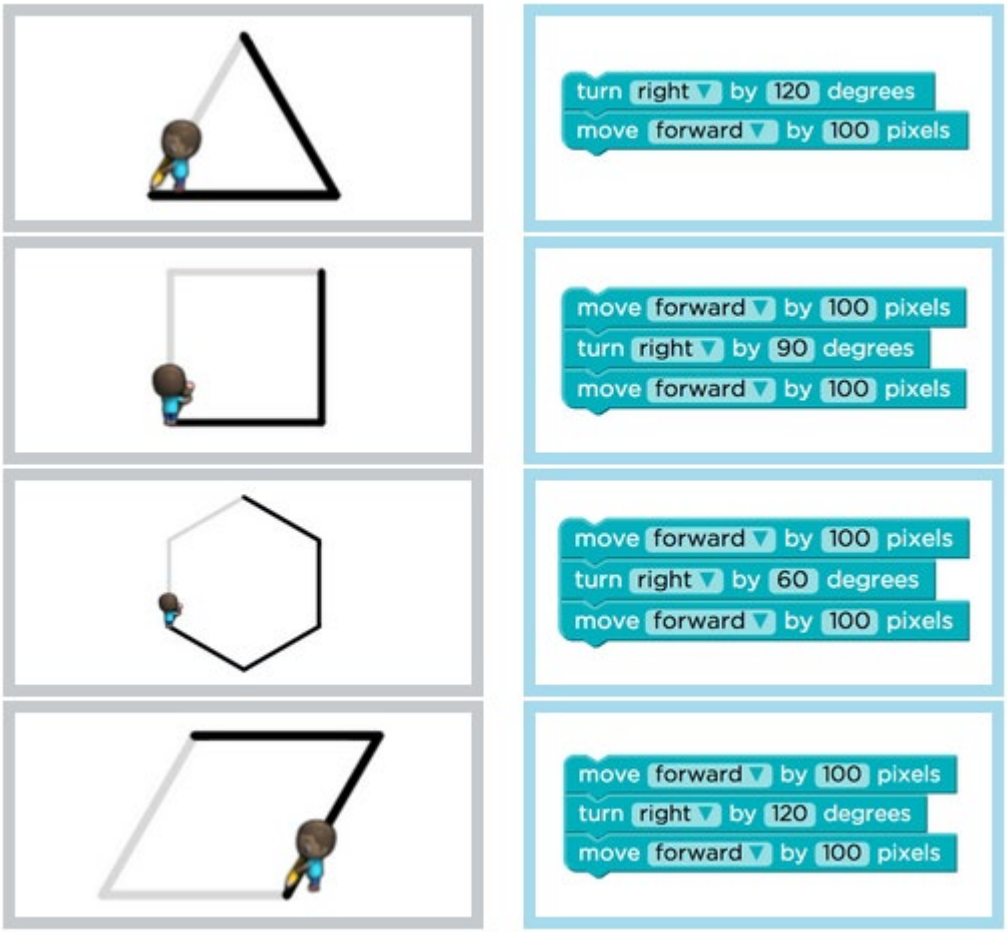
Successivamente l'**esercizio 10** è a tema libero: lo studente può disegnare tutto quello che vuole. C'è a disposizione una nuova istruzione (sotto a sinistra) che consente di impostare la larghezza del tratto con cui si disegna. Variandolo insieme al colore nel disegno dei vari tratti possono essere realizzate interessanti effetti, tipo quello mostrato sotto a destra



il cui programma è il seguente

Nel successivo esercizio (11) si affronta una situazione analoga a quelle precedente mentre l'ultimo, **esercizio 12** – Associazioni, è di verifica. In esso bisogna associare ad ogni figura la sequenza di blocchi necessaria a completarla correttamente.

Le possibili risposte sono programmi con le istruzioni in inglese (per le traduzioni si veda quanto riportato a pag. 31) e la soluzione corretta è la seguente.



5.1.2 Corso 2: Lezione 7 (Ripetizione – Artista)

Il concetto di ripetizione è stato introdotto e discusso nella sezione 3.2 a pag. 23, cui si rimanda per ulteriori approfondimenti.

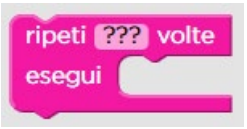
Il contesto di questa lezione è ancora quello del disegno di forme geometriche da parte di un artista, che ha a disposizione in più l'istruzione di ripetizione (il ciclo "ripeti … volte").

In questa lezione si ripetono essenzialmente sequenze di azioni e le difficoltà sono centrate principalmente sul capire le quantità numeriche da usare affinché i cicli raggiungano il risultato desiderato.

Mentre l'esercizio 1 propone allo studente una situazione di tipo già noto, in cui la soluzione è costituita da una sequenza di istruzioni, nell'**esercizio 2**, si ripropone la

stessa figura da completare (sotto a sinistra), ma questa volta ripetendo la sequenza di blocchi che si trova già pronta nell'area di lavoro (sotto a destra).

Adesso è disponibile un'ulteriore istruzione per guidare l'artista (cioè l'esecutore è in grado di realizzare una nuova azione), che fa sì che l'esecutore ripeta le istruzioni che verranno inserite al suo interno (vedi a fianco).



Come si vede il blocco con l'istruzione di ripetizione presenta dei punti interrogativi: per inserire il numero di ripetizioni desiderato basta cliccare sui punti interrogativi, scrivere il valore desiderato e premere il tasto "Invio" ("Return" o "Enter" sulle tastiere inglesi).

Non dovrebbe esser difficile per lo studente, o perché sta ripassando concetti già appresi, o per il suo livello di maturazione, capire che la soluzione è quella a fianco.

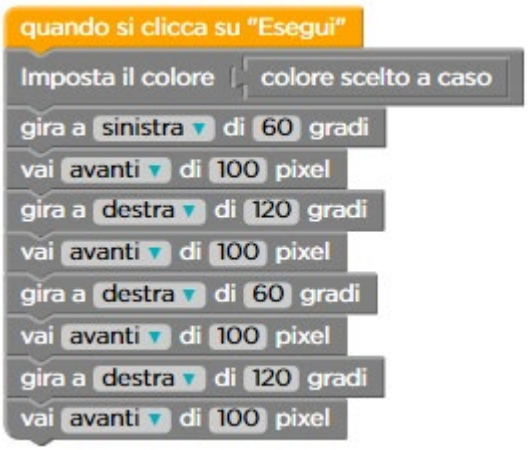
Nell'esercizio successivo (3) si deve ancora ripetere una sequenza di istruzioni, così come in quello ancora seguente (4), in cui però il numero di ripetizioni richiede la conoscenza della quantità di gradi di cui è composto un angolo giro. Può essere un'occasione (come molte altre nelle lezioni che hanno come protagonista l'artista) per rivedere il relativo concetto geometrico.

L'**esercizio 5** complica un pochino lo scenario, dal momento che non vi è più una sequenza pronta per la ripetizione, ma lo studente la deve determinare da solo. In questo caso l'obiettivo è il disegno di un ottagono e lo studente deve quindi calcolare di quanti gradi è necessario girare (si veda la tabellina riepilogativa delle direzioni a pag. 76). La soluzione è quindi quella sotto a destra.

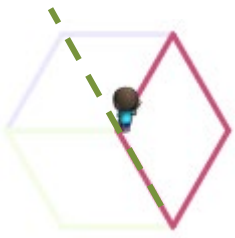
Il successivo esercizio 6 affronta di nuovo il numero di gradi in cui è composto un angolo giro e richiede la ripetizione di una sequenza allo scopo di disegnare un cerchio. L'esercizio ancora seguente (7) è molto semplice (dal momento che richiede soltanto il disegno di una "V") ma è propedeutico al successivo, **esercizio 8**, nel quale il disegno della "V" deve essere ripetuto per formare un motivo geometrico complesso (vedi sotto a sinistra). La sequenza di base da ripetere viene comunque mostrata già pronta nell'area di lavoro e si tratta a questo punto solo di determinare il numero complessivo di ripetizioni (vedi soluzione sotto a destra).

Nell'**esercizio 9** l'obiettivo viene complicato dal fatto che il motivo geometrico da costruire è ancora più complesso (vedi sotto a sinistra), perché richiede di girare più volte e bisogna aggiungere alla sequenza già pronta nell'area di lavoro (che serve per disegnare la "V" che costituisce il motivo base) un'istruzione per girare appena finita la "V" nella direzione del cerchio che si sta percorrendo. La soluzione è mostrata sotto a destra.

L'esercizio successivo (10) è ancora molto semplice (dal momento che richiede soltanto il disegno di un rombo – bisogna unicamente fare attenzione all'orientamento iniziale dell'artista) ed è anch'esso propedeutico al successivo, **esercizio 11**, in cui il disegno del rombo va ripetuto per 3 volte mediante un ciclo. La difficoltà di questo scenario (vedi sotto a sinistra) consiste solo nel determinare di quanto ruotare al termine della sequenza, già pronta nell'area di lavoro (vedi sotto a destra), che permette di disegnare il rombo. Anche questo costituisce quindi un esercizio sul numero di gradi di cui è composto un angolo giro.



Eseguendo il codice così come si trova pronto nell'area di lavoro si vede che l'artista termina il primo rombo orientato lungo la direzione (evidenziata dal tratteggio) dell'ultimo lato del rombo che ha appena disegnato (vedi sotto a sinistra). Non dovrebbe quindi essere molto difficile determinare in che direzione girare e di quanto per disegnare il primo lato del secondo rombo. La soluzione è mostrata sotto a destra ed evidenzia che il primo passo della sequenza che era già pronta nell'area di lavoro fa girare l'artista esattamente di quanto serve per disegnare correttamente il secondo rombo una volta terminato il primo.



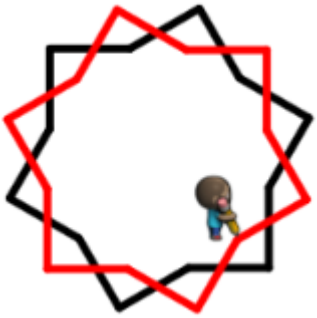
Anche nell'**esercizio 12** l'obiettivo viene complicato dal fatto che il motivo geometrico da costruire è più complesso (vedi sotto a sinistra) di quello dell'esercizio immediatamente precedente perché richiede di girare più volte e bisogna aggiungere alla sequenza già pronta nell'area di lavoro (che serve per disegnare il rombo che costituisce il motivo base) un'istruzione per girare appena finito il rombo nella direzione del cerchio che si sta percorrendo. La soluzione è mostrata sotto a destra.

Osserviamo che lo stesso risultato si può ottenere, oltre che con questa soluzione canonica, anche girando a sinistra invece che a destra, all'ultima istruzione del ciclo. Analogamente, la figura desiderata si ottiene aggiungendo l'istruzione per girare di 90 gradi prima della sequenza per disegnare il rombo, invece che dopo: anche in tal caso si può indifferentemente girare a destra o a sinistra. Ciò che cambia nei vari casi è la sequenza con cui vengono disegnati i vari rombi e se la sequenza si svolge in senso orario o antiorario.

L'**esercizio 13** è un esercizio a tema libero e si può disegnare qualunque cosa venga in mente. Un esercizio che prende spunto dal precedente esercizio basato sulla ripetizione delle "V" può essere quello di capire come fare a disegnare poligoni regolari di forma stellata, tipo l'esagono stellato mostrato sotto a sinistra, il cui programma è sotto a destra.

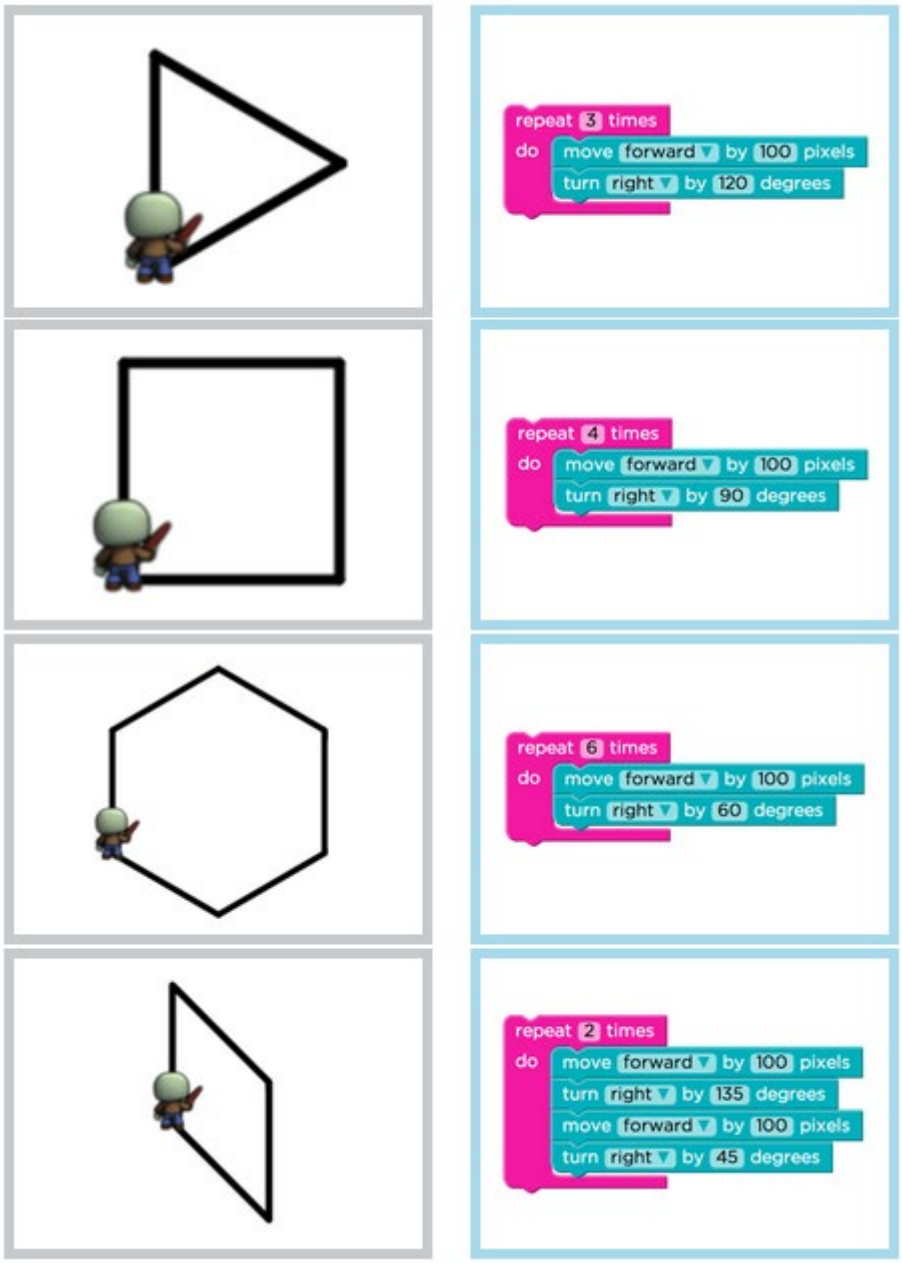
L'osservazione fondamentale per capire perché il programma genera correttamente l'esagono è osservare che ad ogni iterazione del ciclo si gira prima a sinistra di 90

gradi e poi a destra di 30 gradi. Il risultato netto è che l'artista, che all'inizio è orientato nella direzione corrispondente al punto cardinale Est, ha variato la sua orientazione – dopo la prima esecuzione della sequenza contenuta nel ciclo – di 60 gradi (la differenza fra 90 e 60) verso sinistra, cioè verso il Nord. Ripetendo per 6 volte tale sequenza viene completato l'angolo giro e quindi l'esagono. Ulteriore variazione di questo tema è disegnare un secondo poligono della stessa forma disposto simmetricamente rispetto al prima (vedere sotto a sinistra), che si ottiene col programma mostrato sotto a destra.



Al disegno ottenuto viene associato un indirizzo web, che può essere condiviso tramite i social network (in modo da mostrare il proprio lavoro agli amici) oppure può essere memorizzato per successive modifiche o ulteriori elaborazioni.

Gli ultimi tre esercizi sono di verifica. Nel primo di questi, **esercizio 14** – Associazioni, bisogna associare ogni figura geometrica al ciclo che la disegna. Le possibili risposte sono programmi con le istruzioni in inglese (per le traduzioni si veda quanto riportato a pag. 31) e la soluzione corretta è la seguente.

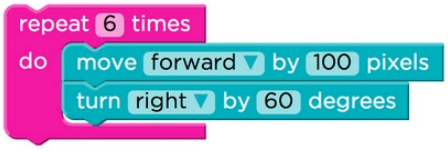

repeat 3 times
do move forward by 100 pixels
turn right by 120 degrees
repeat 4 times
do move forward by 100 pixels
turn right by 90 degrees
repeat 6 times
do move forward by 100 pixels
turn right by 60 degrees
repeat 2 times
do move forward by 100 pixels
turn right by 135 degrees
move forward by 100 pixels
turn right by 45 degrees

Nel successivo, **esercizio 15** – Risposte Multiple, bisogna invece selezionare la figura che viene disegnata dalla sequenza di istruzioni presentata.

I blocchi sono in inglese e le traduzioni sono quelle riportate a pag. 31.

La risposta corretta è pertanto quella corrispondente alla figura del rombo.

Nell'ultimo, **esercizio 16** – Risposte Multiple, bisogna sempre selezionare la figura che viene disegnata, questa volta da un ciclo che ripete una sequenza di istruzioni.



I blocchi sono in inglese e le traduzioni sono come sopra.

La soluzione è quella corrispondente alla figura dell'esagono.

5.1.3 Corso 2: Lezione 11 (Verifica – Artista)

Il tema della verifica è stato introdotto e discusso nella sezione 3.3 a pag. 38, cui si rimanda per ulteriori approfondimenti.

In questa lezione il contesto è ancora quello dell'Artista che col suo movimento disegna figure sullo schermo. Come nel Modulo 1, viene proposto del codice già pronto per risolvere l'esercizio, che però contiene uno o più errori. Lo studente deve quindi immedesimarsi al posto dell'esecutore per capire dove avviene lo sbaglio e quindi correggerlo.

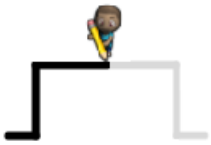
Ricordiamo che, come discusso alle pagine 38 e 44, gli errori possono essere costituiti da:

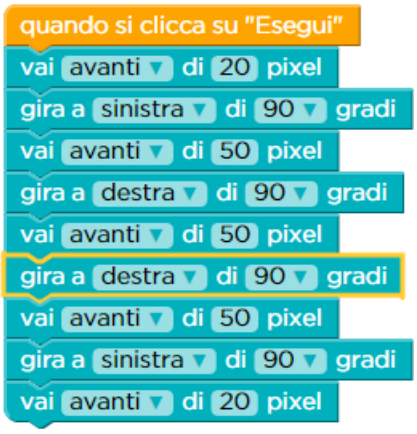
- blocchi mancanti,
- blocchi superflui,
- blocchi collocati nell'ordine sbagliato o nel posto sbagliato
- blocchi sbagliati o con parametri sbagliati rispetto all'obiettivo dell'esercizio.

Poiché in questa lezione non è disponibile il pulsante "Fai un passo", è opportuno eseguire lentamente il codice agendo sul regolatore di velocità precedentemente discusso nel paragrafo 5.1.1 a pag. 73 in modo da osservare qual è l'istruzione che contiene un parametro sbagliato

Nell'**esercizio 1** l'obiettivo è quello di disegnare il cappello la cui traccia è visibile sullo schermo.

La figura seguente mostra la situazione dopo l'esecuzione del blocco evidenziato. L'artista è orientato verso il basso e quindi nella successiva istruzione dopo quella evidenziata si muoverà di 50 pixel verso il basso. Riesci a capire dov'è l'errore?



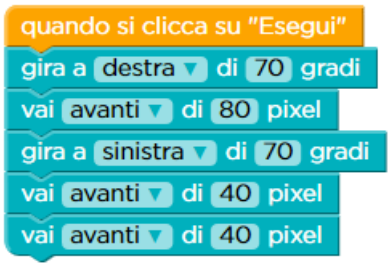


Si tratta dell'istruzione precedente quella evidenziata, che ha spostato l'artista in avanti di una quantità di pixel insufficiente a fargli disegnare l'intero bordo superiore del cappello: gliene ha fatto disegnare solo metà. Si tratta quindi di un'**istruzione con parametro sbagliato**.

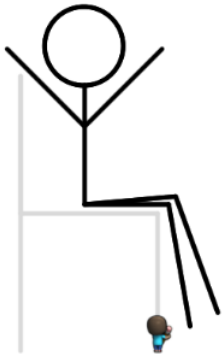
Per ottenere la soluzione corretta è dunque necessario che il parametro del blocco precedente quello evidenziato blocco celeste abbia un valore di 100 pixel anziché di 50.

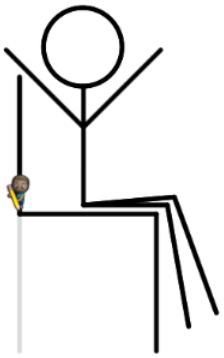
Mentre nei due esercizi successivi (2 e 3) si tratta di correggere un errore dello stesso tipo, il tipo di errore dell'**esercizio 4** è diverso. L'artista deve completare il disegno, tracciando l'intera gamba sinistra della figura rappresentata qui a fianco.

Però l'esecuzione del codice che si trova già pronto nell'area di lavoro, visualizzato sotto a sinistra, conduce a produrre la figura mostrata sotto a destra, in cui si vede che il piede sinistro è stato disegnato troppo lungo. L'errore è quindi dovuto ad un'**istruzione superflua**, proprio l'ultima, che va quindi rimossa.



Anche nel successivo **esercizio 5** si tratta di completare correttamente un disegno, quello mostrato sotto a sinistra. Il codice che si trova già pronto, visualizzato sotto al centro, produce però la figura che si vede sotto a destra, in cui manca una gamba della sedia.





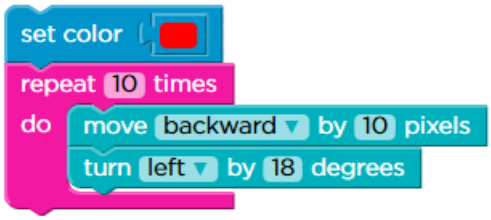
Osservando il codice in esecuzione si nota che l'ultimo tratto disegnato dall'artista è la metà di quanto serve. In questo caso quindi la correzione dell'errore può avvenire in due modi: duplicando l'ultimo blocco (**istruzione mancante**) oppure raddoppiando il valore del parametro nell'ultimo blocco, da 150 a 300 (**istruzione con parametro sbagliato**).

Nell'**esercizio 6** il codice già pronto (vedi sotto al centro), che dovrebbe disegnare la lettera "C" che serve per completare la parola "CODE", ottiene un risultato (vedi sotto a destra) diverso da quello desiderato.

Scoprire la soluzione rimanendo nel numero di blocchi richiesto (7) non è immediato perché richiede di individuare due errori: un'**istruzione mancante** ed un'**istruzione con parametro sbagliato**. La soluzione corretta è quella mostrata qui a fianco, con evidenziate le correzioni agli errori.

Nei successivi cinque esercizi (7, 8, 9, 10, 11) l'errore è costituito sempre da un valore sbagliato di un parametro. Suggeriamo di eseguire sempre il codice che si trova già pronto (eventualmente agendo sul regolatore di velocità per rallentare l'esecuzione) in modo da comprendere bene l'effetto di ogni blocco. Sarà più facile, dopo di ciò, riuscire a capire qual è l'errore e come intervenire per correggerlo.

L'**esercizio 12** (quello finale) è una verifica a risposte multiple. Viene proposto un programma (vedi sotto) con quattro istruzioni per disegnare un cerchio. Il programma però ne disegna solo metà, e si chiede di individuare quale sia l'istruzione da modificare.



Il programma è proposto in inglese. La corrispondenza tra le istruzioni in inglese e quelle in italiano è quella mostrata a pag. 31.

In questo caso non c'è la possibilità di eseguire il programma per vedere l'effetto. Se si vuole, si può riscrivere il programma usando l'esercizio 13 della lezione 7 del corso 2, che è un esercizio a testo libero, ed eseguirlo in quell'esercizio per vedere cosa accade. Altrimenti, si deve usare un ragionamento basato su considerazioni di natura geometrica, osservando che, ad ogni ripetizione dei blocchi contenuti all'interno del ciclo, l'artista gira di 18 gradi. Di conseguenza, con le 10 ripetizioni girerà solo di 180 gradi, cioè la metà di quanto serve per completare un cerchio. Bisognerà quindi modificare il numero di ripetizioni in modo che giri complessivamente di 360 gradi.

5.2 L'istruzione condizionale

Come detto nell'introduzione di questo modulo (pag. 73) il nuovo concetto fondamentale introdotto è quello di **istruzione condizionale**, che permette ai programmi di scegliere le istruzioni da eseguire, in funzione delle differenti situazioni che s'incontrano.

Per lavorare su questo concetto si usano la lezione 13 del corso 2 e i primi 6 esercizi della lezione 7 del corso 3. Nella prima di queste lezioni si usa l'istruzione condizionale con una sola alternativa ("se"), nella seconda l'istruzione condizionale a due alternative ("se-altrimenti") e a tre alternative ("se / altrimenti-se / altrimenti").

In tutti i casi l'istruzione condizionale chiede se una certa condizione è vera. Nel caso di istruzione condizionale ad una sola alternativa, questa viene eseguita solo se la condizione è vera. Nel caso di istruzioni condizionale a due alternative, la prima alternativa viene eseguita solo se la condizione è vera, mentre la seconda viene eseguita solo se la condizione è falsa. Il caso di istruzioni condizionale a tre alternative verrà spiegato direttamente più avanti.

Gli esercizi lavorano anche sulla combinazione delle istruzioni accoppiata all'uso dell'istruzione condizionale.

5.2.1 Corso 2: Lezione 13 (Istruzioni condizionali – Ape)

I primi due esercizi di questa lezione riprendono l'ambientazione dell'ape che deve raccogliere il nettare dai fiori e produrre il miele nei favi (la quantità di nettare o miele è indicata da un numeretto vicino al fiore o al favo). Le istruzioni da usare sono quelle già note di spostamento/azione e di ripetizione.

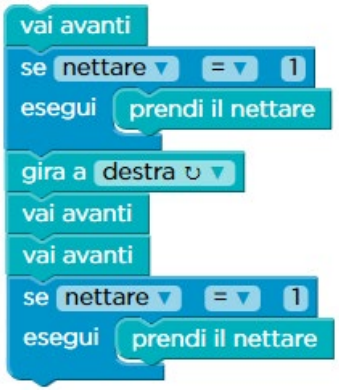
Con l'**esercizio 3**, in cui l'ape si trova in una situazione in cui non sa se il fiore viola contenga o meno il nettare (vedi sotto a sinistra, c'è un punto interrogativo vicino al fiore), si introduce l'istruzione condizionale, che serve in questo caso a verificare quale sia la condizione del fiore. Il codice che si trova già predisposto nell'area di lavoro (vedi sotto a destra), dopo aver fatto arrivare l'ape sul fiore, usa l'istruzione condizionale ("se") per verificare se il fiore viola ha una dose di nettare. La condizione da verificare in questo caso è "nettare = 1", si tratta cioè di verificare se il fiore ha una dose di nettare. Solo se la condizione è vera l'ape potrà eseguire la raccolta del nettare.

Per risolvere correttamente l'esercizio bisogna quindi inserire il blocco "prendi il nettare" all'interno del blocco "se". Quando si manda in esecuzione il codice, il punto in-

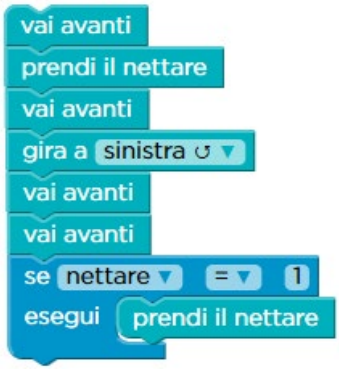
terrogativo si trasforma in 1 oppure 0 per indicare la presenza o l'assenza del nettare. L'istruzione condizionale potrà quindi determinare correttamente in quale situazione si trovi ed eseguire correttamente il blocco "prendi il nettare" solo se il fiore lo possiede.

Il successivo esercizio 4 è dello stesso tipo, salvo che non ci sono blocchi già pronti, e bisogna quindi costruire la soluzione a partire da zero.

Nell'**esercizio 5** vi sono due fiori viola (vedi sotto a sinistra), e per ognuno di essi, una volta raggiunto, bisogna usare un'istruzione condizionale per eseguire correttamente le istruzioni (vedi soluzione sotto a destra).



Nello scenario dell'**esercizio 6** (vedi sotto a sinistra) si trovano un fiore rosso (la cui quantità di nettare è nota) e un fiore viola (con quantità sconosciuta di nettare). Per quest'ultimo l'ape dovrà quindi raccogliere il nettare solo dopo aver verificato (con l'istruzione condizionale) che esso lo possieda davvero (vedi sotto a destra).

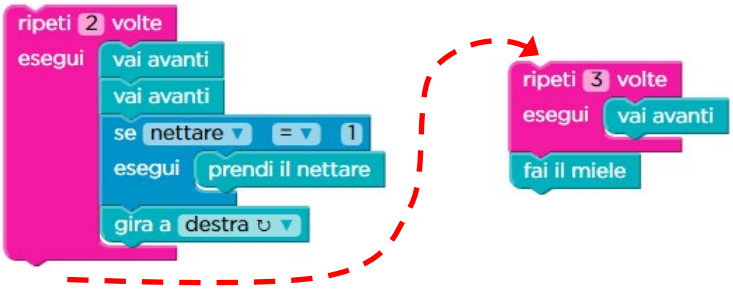


I successivi due esercizi (7, 8) sono dello stesso tipo, mentre nell'**esercizio 9** è necessario combinare l'istruzione condizionale e l'istruzione di ripetizione. L'ape si trova infatti di fronte un'intera fila di fiori viola (vedi sotto a sinistra) e deve quindi ripetere più volte l'azione di controllo realizzata dall'istruzione condizionale. Sappiamo che in questi casi si usa un ciclo (cioè un'istruzione di ripetizione), per cui la soluzione è quella mostrata sotto a destra.

Con l'**esercizio 10** viene aggiunto allo scenario il favo su cui produrre il miele (vedi sotto a sinistra), ma la reale difficoltà dell'esercizio è data dalla necessità di risolverlo rispettando il vincolo sul numero di blocchi da usare. La soluzione più diretta, mostrata sotto a destra, porta ad usare 13 blocchi in totale, 3 in più di quanto indicato.

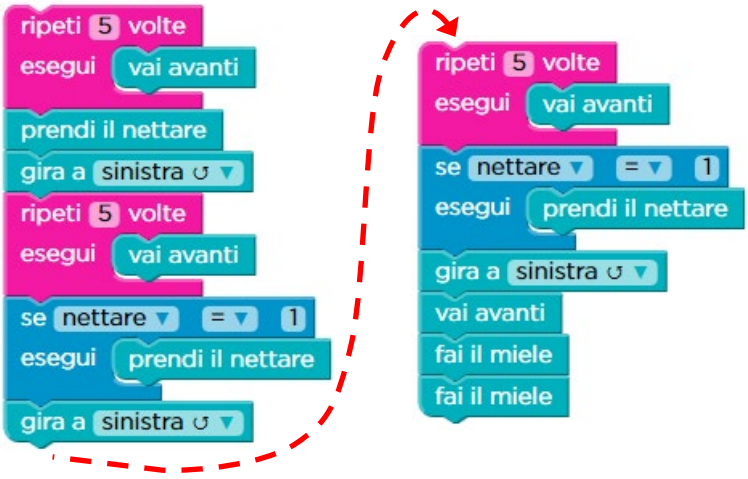
Questa soluzione è quindi corretta, ma non è la più efficiente in termini di numero di blocchi usati (**efficienza di spazio**). La chiave per una soluzione che risponde ai requisiti è osservare che l'ape ripete per due volte la stessa sequenza di movimenti (vai avanti – vai avanti – gira a destra). La differenza tra i due casi è che la prima volta l'ape va a finire sul fiore rosso, la cui quantità di nettare è nota. Ma non è un errore usare anche in tal caso l'istruzione condizionale: si tratta di un'istruzione superflua,

dal momento che è noto che questo fiore rosso possiede una dose di nettare, che però permette di scrivere un programma (vedi sotto) composto da un minor numero di istruzioni[15].



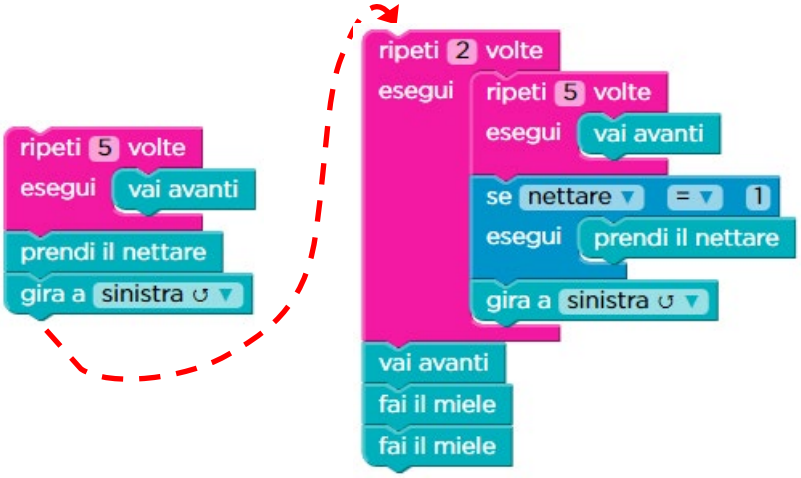
Il successivo **esercizio 11** è per alcuni aspetti analogo, dal momento che è presente un fiore rosso insieme a due fiori viola. I fiori però sono molto distanti tra loro, ed è quindi necessario usare ogni volta un blocco di ripetizione per far spostare l'ape fino a farla arrivare sul fiore.

La soluzione più diretta consiste quindi nel gestire separatamente la raccolta su ognuno dei tre fiori, come mostrato nella figura qui sotto.



[15] Osserviamo per completezza che, dal momento che la soluzione così ottenuta conduce ad eseguire un'istruzione superflua, essa in termini di efficienza di tempo è peggiore della precedente soluzione, che è invece meno efficiente in spazio.

Questa soluzione permette di risolvere l'esercizio con 18 blocchi come richiesto dalle indicazioni nella parte superiore dell'area di lavoro. Uno studente più attento potrebbe però osservare che la sequenza di istruzioni usata per raccogliere il nettare dai fiori viola ("ripeti 5 volte" / "se nettare = 1" / "gira a sinistra") viene ripetuta due volte. Potrebbe quindi chiedersi, sulla base di questa osservazione, se non si può usare ancora per questa evenienza il blocco di ripetizione per ottenere una soluzione con un numero inferiore di blocchi. Se è particolarmente maturo potrebbe quindi scoprire da solo la soluzione sotto raffigurata, che conduce a risolvere l'esercizio con 14 blocchi invece di 18 ed è dunque più efficiente dal punto di vista di spazio.



La soluzione sopra mostrata, che inserisce un ciclo all'interno di un altro ciclo, è al di là del livello medio di sviluppo raggiunto dagli studenti a questo stadio del percorso e verrà discussa più avanti (vedi sezione 6.2 "Il ciclo annidato", a pag. 109).

Infine, uno studente particolarmente brillante potrebbe accoppiare questa osservazione al metodo usato per la soluzione del precedente esercizio. Anche nel corrente esercizio, infatti, dal momento che la distanza da percorre per raggiungere il fiore rosso è uguale a quella per raggiungere i fiori viola, si può usare uno stesso ciclo per arrivare al fiore, prendere il miele e girare. Si sprecherà, durante l'esecuzione del codice mostrato qui a fianco, un passo nel quale ci si chiede (inutilmente) se il fiore rosso ha del miele, ma si ottiene una soluzione ancora più efficiente dal punto di vista di spazio, dal momento che usa solo 10 blocchi. La contropartita è un lieve peggioramento dell'efficienza di tempo.

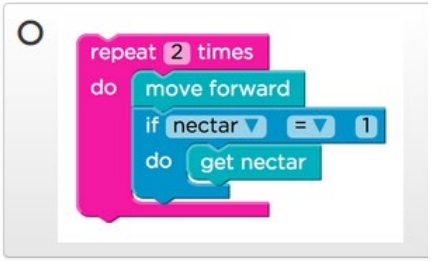
I due esercizi successivi (12, 13) presentano ancora la combinazione tra blocco di ripetizione ed istruzione condizionale.

Gli ultimi due esercizi sono invece esercizi di verifica. Il primo (**esercizio 14** – Risposte Multiple) è un semplice esercizio sui segni di uguaglianza e disuguaglianza: si tratta di scegliere l'unico confronto corretto (1>0). Nel secondo esercizio di verifica (**esercizio 15** – Risposte Multiple) bisogna scegliere la sequenza di blocchi corretta per far sì che l'ape raccolga correttamente il nettare nella situazione in cui si trova (visualizzata qui a fianco).

Le possibili risposte sono in inglese. Per la corrispondenza tra le istruzioni in inglese e quelle in italiano vedere a pag. 31 e qui a fianco.

if nectar	=	se nettare
get nectar	=	prendi il nettare

Pertanto la soluzione è quella mostrata a fianco.



5.2.2 Corso 3: Lezione 7 (Istruzioni condizionali – Ape)

L'ambientazione di questa lezione è ancora quella dell'ape, ma in questo caso la situazione d'incertezza (per risolvere la quale è necessario usare l'istruzione condizionale) è legata alla presenza di un fiore o di un favo e non alla natura del fiore (come nel precedente paragrafo). L'istruzione condizionale consente quindi di verificare se sotto la nuvoletta si nasconde un fiore o un favo. Nel primo caso si dovrà raccogliere il nettare, mentre nel secondo produrre il miele.

L'**esercizio 1** è semplice, poiché contiene una sola situazione d'incertezza (sotto a sinistra) e l'ape vuole solo raccogliere il nettare. È necessario soltanto ricordarsi di usare un ciclo per avvicinarsi al fiore senza usare un numero eccessivo di blocchi. La soluzione è quindi quella visualizzata sotto a destra.

Nell'**esercizio 2** vi sono due situazioni d'incertezza (vedi sotto a sinistra), ma l'obiettivo dell'ape è sempre soltanto raccogliere il nettare. Anche in questo caso va fatta attenzione ad usare un ciclo per risolvere l'esercizio soddisfacendo le indicazioni sul numero di blocchi da usare, come mostrato sotto a destra.

Anche nell'**esercizio 3** (sotto a sinistra) vi sono due situazioni d'incertezza, ma adesso l'obiettivo dell'ape è cambiato ed è solo quello di produrre il miele.

Inoltre, la diversa collocazione delle nuvolette richiede, per poter risolvere l'esercizio col numero di blocchi richiesto, di combinare in modo diverso il blocco di ripetizione e gli altri blocchi. La soluzione è la seguente.

Fino ad ora l'ape ha avuto un solo obiettivo ed è quindi stata sufficiente l'istruzione condizionale con una sola alternativa, cioè con una sola azione che può essere eseguita o meno in funzione della verità di una certa condizione. Ad esempio, nell'esercizio qui sopra, quando la condizione "se c'è un favo" è "vera" allora viene eseguita l'alternativa "fai il miele"; ma se la stessa condizione è "falsa" allora non si esegue alcuna azione.

A partire dall'**esercizio 4** (a fianco) l'ape vuole sia prendere il nettare che fare il miele. È quindi necessario usare una nuova versione dell'istruzione condizionale, detta "se – altrimenti", che specifica due possibili alternative. La prima indica l'azione da eseguire se la condizione è "vera", mentre la seconda specifica l'azione da eseguire se la condizione è "falsa". In questo scenario l'ape potrà quindi, eseguendo il codice mostrato qui sotto,

effettuare l'azione corretta sia che la nuvoletta nasconda un fiore sia che celi un favo. Il blocco o i blocchi di codice inseriti in corrispondenza di "esegui" sono quelli che vengono eseguiti quando la condizione indicata ("se c'è un fiore", in questo caso) è vera, mentre quelli inseriti in corrispondenza di "altrimenti" sono quelli che vengono eseguiti quando essa è falsa.

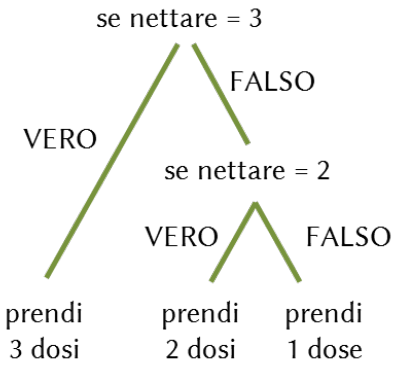
Nell'**esercizio 5** (sotto a sinistra) le nuvolette sono diventate quattro, agli angoli di un quadrato, con diverse caselle di spazio tra l'una e l'altra. È quindi necessario, dopo aver eseguito l'azione appropriata per ogni situazione d'incertezza, spostarsi in avanti per raggiungere la successiva nuvoletta. Il modo più diretto di farlo è quello raffigurato sotto a destra che ha conduce però ad usare un blocco in più di quanto richiesto dalle indicazioni presenti nella zona superiore dell'area di lavoro (che specifica 9 blocchi).

La soluzione che rispetta il requisito (mostrata qui a destra) richiede di inserire nel ciclo più esterno, che fa percorrere i quattro lati del quadrato, un ulteriore ciclo che esegue la ripetizione degli spostamenti in avanti dell'ape. Anche se tale concetto (ciclo annidato) non è ancora stato introdotto a questo stadio del percorso didattico (verrà fatto nella sezione 6.2 "Il ciclo annidato", a pag. 109), non dovrebbe essere difficile per gli studenti produrre questa soluzione. In ogni caso, anche la precedente soluzione, senza ciclo annidato, è accettabile a questo stadio.

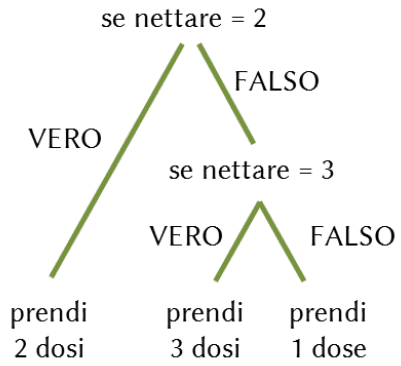
Il successivo **esercizio 6** (sotto a sinistra) introduce un'ulteriore complicazione. In questo scenario abbiamo di nuovo il fiore viola, ma la sua situazione di incertezza è più complessa. Non si tratta infatti di sapere se contenga o meno del nettare, ma – avendo la certezza che contiene del nettare – di determinare esattamente quante dosi

di nettare contiene: può infatti averne 1, 2, o 3. A seconda di quale sia la situazione reale bisognerà eseguire il numero corrispondente di volte l'istruzione "prendi il nettare". Per trattare una situazione di questo genere è necessario usare ancora l'istruzione condizionale a due alternative. L'esercizio propone già una traccia della soluzione, basata su chiedersi in prima battuta se il fiore possiede o meno 3 dosi di nettare ed inserire poi nell'alternativa corrispondente al caso negativo un'ulteriore istruzione condizionale a due alternative che consenta di capire (visto che non ci sono 3 dosi di nettare) se le dosi di nettare sono 2 oppure 1. Bisogna solo inserire le opportune azioni per entrambe le alternative, come mostrato sotto a destra.

Si ottiene in questo modo una sorta di istruzione condizionale a tre alternative, che permette di scegliere una fra tre possibili azioni a seconda di quale delle tre possibili condizioni si verifica. Qua a fianco è mostrata una raffigurazione grafica del corrispondente processo decisionale. Tale raffigurazione è spesso chiamata "**albero di decisione**", perché ad ogni snodo corrispondente alla verifica di una condizione vi è uno sdoppiamento delle possibili azioni che ricorda il processo di ramificazione degli alberi.



Osserviamo che l'albero di decisione, nel caso di scelta fra più di due alternative, può essere strutturato in funzione della propria visione concettuale del problema. Ad esempio, per lo stesso scenario sopra considerato si potrebbe ritenere più adatto alla propria visione del problema, il seguente albero di decisione, corrispondente al codice raffigurato al suo fianco.



In un caso di questo genere si ottiene un programma del tutto equivalente, non soltanto in termini di risultati, ma anche di efficienza di spazio e di tempo. In generale, però, la scelta della struttura dell'albero di decisione può influenzare l'efficienza (di spazio e di tempo) del programma risultante.

(questa pagina è intenzionalmente vuota)

6 Modulo 3: gestione di eventi

Nel Modulo 3 si impara un'ulteriore evoluzione del concetto di ripetizione. I costrutti di ripetizione possono contenere al loro interno ulteriori costrutti di ripetizione, dando luogo al concetto di **ciclo annidato**.

Il nuovo concetto fondamentale studiato nel modulo è quello dell'**evento**, cioè del meccanismo che permette al programma di reagire durante la sua esecuzione ad azioni dell'utente o ad altri avvenimenti rilevanti. Si introduce quindi il costrutto **gestore dell'evento**, costituito da una sequenza di istruzioni che vengono eseguite solo quando l'evento si verifica, insieme ad un ulteriore variante del costrutto di ripetizione, denominata **ripeti per sempre**, utile per la realizzazione di programmi interattivi che possono andare avanti finché si vuole.

Il modulo contiene una sezione iniziale utile sia per ripassare i concetti precedentemente appresi, nel caso in cui esso venga svolto dopo il Modulo 2, sia per introdurre e discutere tali concetti nel caso in cui questo modulo venga usato come punto di inizio di un percorso di formazione alla programmazione informatica.

6.1 Ripasso dei concetti già appresi

Per ripassare i concetti di **sequenza** e di **ripetizione** sinora introdotti usiamo la lezione 3 del corso 3 (Artista), il cui contesto è quello del disegno di forme geometriche da parte di un artista. Rimandiamo al paragrafo 5.1.1 a pag.73 per una descrizione dello scenario di riferimento in cui si svolge la lezione. Per il tema della **verifica** ed il concetto di **istruzione condizionale** dovranno invece essere usati direttamente il paragrafo 5.1.3 a pag. 87 e la sezione 5.2 "L'istruzione condizionale" a pag. 90.

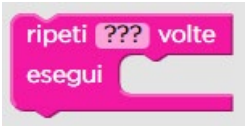
6.1.1 Corso 3: Lezione 3 (Sequenza e Ripetizione – Artista)

Nell'**esercizio 1** vi è il semplice obiettivo di disegnare un quadrato facendo esplicitamente percorrere all'artista i quattro lati. Non è disponibile il blocco di ripetizione.

L'esercizio è dunque dedicato al concetto di **sequenza** di istruzioni. Bisogna quindi solo fare attenzione a cambiare il valore che costituisce il parametro dell'istruzione "vai avanti", inserendo – al posto del 100 presente nel blocco nella cassetta degli attrezzi – il valore 300 indicato nel testo dell'esercizio stesso. Basta operare come indicato nel paragrafo 5.1.1 a pag. 73. Si ottiene quindi la soluzione mostrata a fianco.

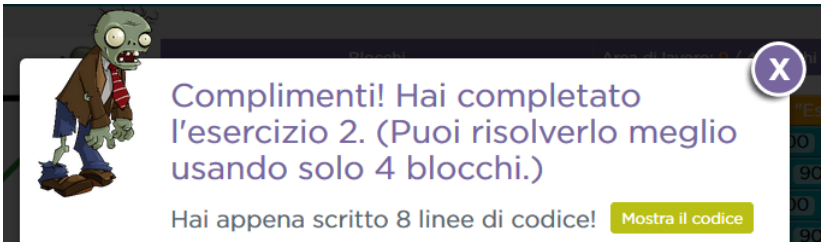
Anche nell'**esercizio 2** l'obiettivo è quello di disegnare un quadrato ma esso deve essere raggiunto definendo una soluzione basata sul concetto di **ripetizione** di istruzioni o meglio di ripetizione di una "sequenza di

istruzioni". Nell'area di lavoro è già pronta una sequenza di istruzioni (quella che nel precedente esercizio veniva esplicitamente ripetuta nella costruzione della soluzione)



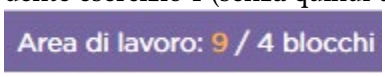
e si chiede di ripeterla per disegnare un quadrato. A tal scopo deve ovviamente essere usato il blocco di ripetizione. Tale blocco (a fianco) è un'istruzione che, in modo sintetico, permette di far ripetere all'esecutore una o più azioni per un certo numero di volte. Essa viene anche detta **ciclo**. Nel suo utilizzo deve essere specificato il numero di volte che l'istruzione o la sequenza di istruzioni inserite al suo interno devono essere ripetute. È sufficiente cliccare sui punti interrogativi, scrivere il valore desiderato e premere il tasto "Invio" ("Return" o "Enter" sulle tastiere inglesi). Si ottiene in tal modo la soluzione mostrata a fianco.

È importante notare che la presenza dell'insegnante è fondamentale, in casi di questo genere, per assicurarsi che lo studente comprenda qual è l'obiettivo dell'esercizio e lavori su di esso. Se infatti in questo esercizio lo studente trascura di leggere o non comprende la richiesta dell'esercizio, potrebbe fornire come soluzione lo stesso codice dell'esercizio precedente. In tal caso il sistema considera l'esercizio risolto, anche se in modo imperfetto, vedi sotto.



C'è dunque il rischio che, in un'ottica da videogioco, lo studente consideri di aver "completato il livello" e proceda avanti ignaro. L'intervento dell'educatore è in tali casi necessario affinché l'attività formativa sia efficace.

Ricordiamo che nell'area di lavoro in alto il sistema indica quanti blocchi ha usato la soluzione presente nell'area di lavoro stessa e quanti bisogna usarne al massimo. Nel caso in cui uno studente predisponesse per l'esercizio 2 la stessa soluzione del precedente esercizio 1 (senza quindi utilizzare il blocco di ripetizione) il sistema



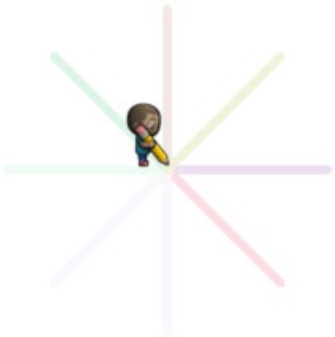
visualizzerebbe quanto mostrato a fianco. Ogni volta che si verifica una situazione di questo genere, in cui la soluzione realizzata raggiunge comunque l'obiettivo finale dell'esercizio ma usa più blocchi di quanto specificato, il sistema visualizza un messaggio col tipo di avvertimento più sopra mostrato.

Con l'**esercizio 3** l'utilizzo della ripetizione diventa un po' più sofisticato. Dovendo infatti disegnare un rettangolo (vedi sotto a sinistra), è chiaro che non si può, come nel caso precedente del quadrato, ripetere per 4 volte la sequenza di istruzioni che disegna un lato. Bisogna invece capire che la sequenza da ripetere non è quella per

disegnare solo un lato ma quella che ne disegna due, ottenendo la soluzione mostrata sotto a destra.

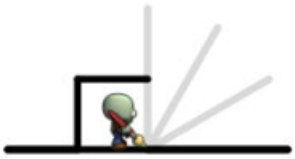
Anche qui la presenza dell'educatore è importante, dal momento che uno studente frettoloso, osservando che si richiede di disegnare una figura che non ha quattro lati e quattro angoli uguali, potrebbe fornire come soluzione la sequenza esplicita di blocchi che disegna il rettangolo un lato alla volta. Il sistema segnala anche in questo caso la possibilità di risolvere meglio l'esercizio, ma può essere necessaria la presenza dell'insegnante per assicurare l'efficacia dell'attività didattica.

L'**esercizio 4** richiede di disegnare una serie di raggi colorati (sotto a sinistra), proponendo una sequenza di blocchi già pronta e sollecitando a capire quale dev'essere il parametro del blocco "ripeti ... volte". Dovrebbe essere semplice per gli studenti capire il valore giusto, sulla base della traccia indicata per il disegno da realizzare, ma può essere utile invitare lo studente a sperimentare. Va fatta soltanto attenzione al fatto che, mentre il sistema si accorge se il numero inserito nel blocco "ripeti ... volte" è inferiore a quello necessario, accetta come validi anche valori superiori ad esso. Può essere l'occasione per una riflessione sulla differenza tra semplice risoluzione dell'esercizio e sua risoluzione ottima (mostrata sotto a destra).



Anche nell'**esercizio 5** (che richiede di tracciare una cerchio riempito completamente di raggi colorati, vedi sotto a sinistra) lo studente trova una sequenza di blocchi già pronta nell'area di lavoro[16]. In questo caso la sequenza contiene già il blocco "ripeti ... volte" e bisogna solo capire l'esatto valore del parametro: esso è pari al numero di gradi di un cerchio, cioè 360 (vedi sotto a destra).

Situazione analoga si trova nell'**esercizio 6**, che richiede di completare la figura mostrata sotto a sinistra. La sequenza di blocchi è già pronta e vanno solo inseriti i valori corretti dei parametri evidenziati in giallo. Questa volta non si tratta del numero di ripetizioni del ciclo ma della direzione in cui girare e della quantità di gradi di cui girare ad ogni passo (vedi sotto a destra).

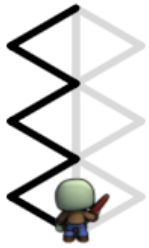


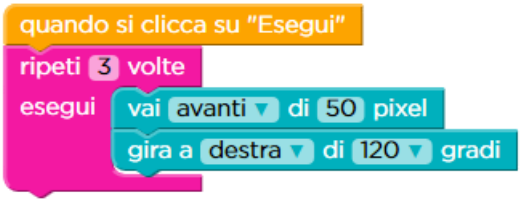
I due esercizi successivi (7, 8) coinvolgono dei semplicissimi calcoli di tipo geometrico per completare i disegni proposti e le soluzioni non richiedono di usare il blocco di ripetizione.

Nell'**esercizio 9** è necessario completare il disegno di un campo di calcio. Si tratta quindi di un esercizio simile al precedente esercizio 3, con la lieve difficoltà aggiuntiva che va disegnata anche la linea di centrocampo. Basta eseguire alcuni calcoli per determinare di quanto muoversi ed aggiungere le relative istruzioni al termine di quelle che disegnano il rettangolo, ottenendo la soluzione a fianco.

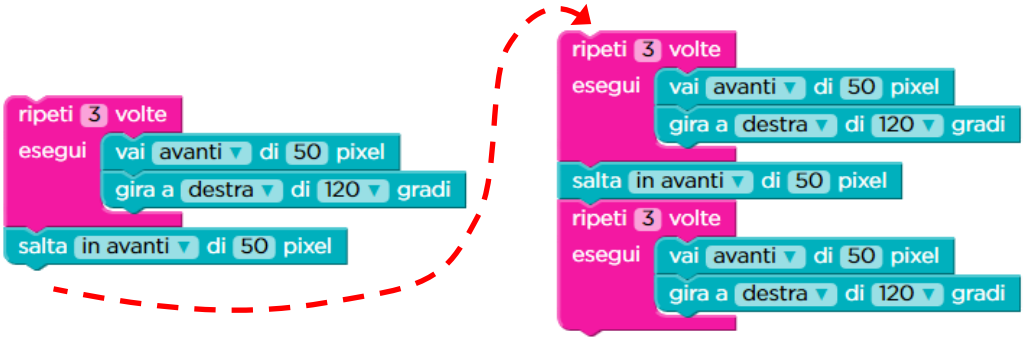
[16] In tal caso la sequenza già pronta di blocchi è in grigio ad indicare che non è possibile modificarla in alcun modo (a parte l'inserimento del valore del parametro del blocco "ripeti ... volte").

Nell'**esercizio 10** si trova una sequenza già pronta di blocchi che, se eseguita, disegna un triangolo equilatero (vedi sotto a destra). L'obiettivo dell'esercizio è completare la figura mostrata sotto a sinistra disegnando 3 triangoli equilateri.





Si potrebbe essere tentati di risolvere l'esercizio replicando esplicitamente tre volte tale sequenza ed aggiungendo l'istruzione di spostamento necessaria per far sì che ogni blocco successivo sia disegnato nella posizione desiderata, come mostrato sotto.



Il sistema in tal caso avviserebbe che si stanno usando più blocchi del necessario.

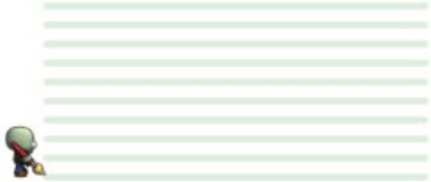
Per ottenere il rispetto del requisito relativo ai blocchi da usare è necessario inserire la sequenza di blocchi che disegna il triangolo equilatero (che si trova già pronta) e l'istruzione di spostamento all'interno di un ulteriore blocco di ripetizione.

Si realizza così un ciclo annidato, concetto su cui si lavorerà in modo esplicito nel prosieguo di questo capitolo (sezione 6.2 "Il ciclo annidato", a pag. 109). La soluzione è mostrata qui a fianco[17].

L'**esercizio 11** richiede un po' di attenzione per essere risolto rispettando il vincolo sul numero di blocchi da utilizzare. L'artista deve infatti disegnare una serie di 10 li-

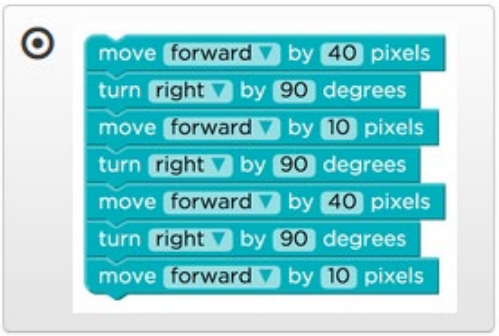
[17] In questa soluzione, come visto diverse volte nella sezione 3.2 "La ripetizione" a pag. 31, vi è durante l'esecuzione una ripetizione inutile dell'ultimo blocco contenuto nel ciclo, che permette però di ottenere una soluzione con maggiore efficienza di spazio.

nee verdi in parallelo (vedi sotto a sinistra). Se però si affronta la soluzione disegnando prima una riga in un verso e poi una riga nell'altro si ottiene una soluzione che, seppure corretta, usa più blocchi del previsto, dal momento che al termine di ogni riga – sia in un verso che nell'altro – è necessario spostarsi per posizionarsi per la riga successiva. La chiave per ottenere la soluzione rimanendo nel limite di blocchi indicato in alto nell'area di lavoro sta nel riportare indietro l'artista al punto di partenza subito dopo aver disegnato la riga, in modo da posizionare l'artista una sola volta per ogni ripetizione del ciclo (sotto a destra)[18].

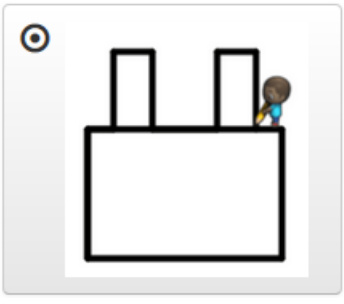


Il successivo esercizio 12 non richiede particolare attenzione, poiché si risolve usando un ciclo per disegnare un triangolo equilatero. I successivi due esercizi sono di verifica.

Nel primo di questi, **esercizio 13** – Risposte Multiple, bisogna individuare il programma che disegna un rettangolo col perimetro di 100 pixel. Le possibili risposte sono programmi con le istruzioni in inglese (per le traduzioni si veda quanto riportato a pag. 31) e la soluzione corretta è mostrata qui a fianco.



Nel successivo, **esercizio 14** – Risposte Multiple, bisogna invece individuare, tra le figure proposte, quella che presenta la simmetria rispetto all'asse verticale. A seconda del livello della classe il concetto potrebbe essere noto o meno agli alunni e può essere questa un'occasione per introdurlo informalmente agli studenti (ad esempio stampando ogni figura su un foglio di carta e spiegando la simmetria con la completa sovrapposizione delle due metà, destra e sinistra, della figura stessa).



[18] Osserviamo che le prime due istruzioni all'interno del ciclo possono anche essere ambedue di disegno o essere invertite di posto (prima salta e poi disegna).

La lezione si conclude con l'**esercizio 15**, a tema libero, in cui si può quindi disegnare qualunque figura venga in mente. Ci si può rifare, per utili spunti, a quanto suggerito nell'esercizio 13 a pag. 84, nel paragrafo 5.1.2.

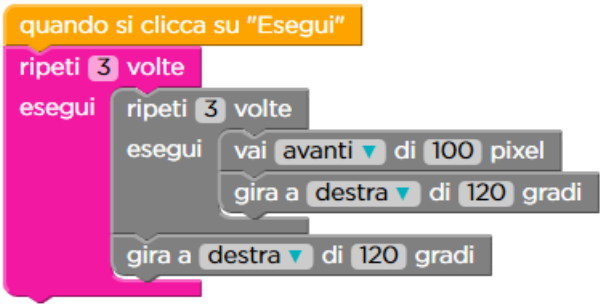
6.2 Il ciclo annidato

In questa sezione viene affrontato in modo esplicito il concetto di **ciclo annidato**. Esso costituisce un'evoluzione del concetto di ciclo, nel senso che consente di introdurre un'ulteriore ripetizione all'interno di una serie di ripetizioni. In senso generale si tratta di qualcosa di già noto: basta pensare – ad esempio – a come un'ora di tempo sia costituita da una ripetizione di minuti ognuno dei quali è una ripetizione di secondi. Per far eseguire all'esecutore un ciclo annidato basta, in questo contesto, inserire un blocco "ripeti ... volte" all'interno di un altro blocco "ripeti ... volte".

Gli studenti imparano ad usare i cicli annidati sia con la lezione 19 del corso 2 (prossimo paragrafo) che con la lezione 11 del corso 3 (paragrafo 6.2.2 a pag. 114), per guidare l'artista nel costruire figure via via più complesse.

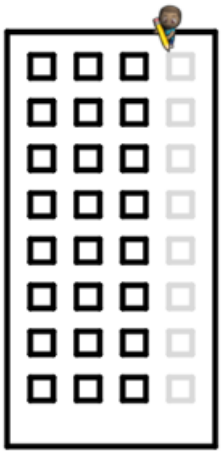
6.2.1 Corso 2: Lezione 19 (Cicli Annidati – Artista)

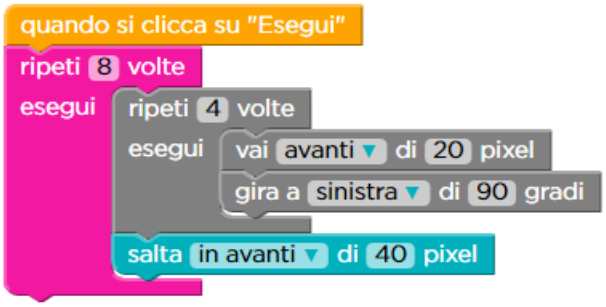
Con l'**esercizio 1** si invita lo studente a verificare cosa succede inserendo un ciclo all'interno di un altro ciclo. Nell'area di lavoro è già pronto del codice (visualizzato in grigio) che contiene un ciclo, e lo studente viene invitato ad inserirlo in un altro blocco "ripeti ... volte". Si raggiunge in tal modo l'obiettivo dell'esercizio, disegnare una figura (sotto a sinistra) costituita dalla ripetizione per 3 volte del disegno un triangolo (ottenuto a sua volta dalla ripetizione per 3 volte del disegno di un lato e della rotazione di 1/3 di angolo giro). La soluzione è sotto a destra.



Nel successivo esercizio 2 bisogna ancora ripetere un triangolo – ma per 6 volte, così come l'**esercizio 3** richiede di ripetere 12 volte il codice che si trova già pronto (che disegna un triangolo usando un ciclo), questa volta però calcolando di quanti gradi girarsi dopo ogni triangolo in modo da ottenere la figura desiderata (mostrata sotto a sinistra). La soluzione è sotto a destra.

L'esercizio seguente (4) è dello stesso tipo, tranne che per il fatto che invece di disegnare con la ripetizione più interna un triangolo si disegna un rombo, mentre nell'**esercizio 5** si trova già pronto del codice per disegnare un quadrato usando un ciclo. Questo serve per disegnare una delle finestre dell'edificio mostrato sotto a sinistra ed andrà inserito all'interno di un ulteriore ciclo che serve a disegnarle tutte. È necessario fare attenzione a quanto saltare per disegnare la finestra successiva: allo spazio che separa tra loro le finestre va aggiunta la dimensione della finestra stessa. Il programma che risolve l'esercizio è visualizzato sotto a destra.





Anche nel successivo esercizio 6 si tratta di completare un disegno (questa volta usando triangoli) così come nell'**esercizio 7**. In quest'ultimo caso, però, la serie di 10 triangoli da disegnare mediante il ciclo più esterno deve essere disposta a cerchio (vedi sotto a sinistra). Il codice che si trova già pronto serve per disegnare un triangolo e bisogna quindi calcolare, oltre alla distanza di cui spostarsi, anche l'angolo di

cui ruotare prima di iniziare il disegno del triangolo seguente. La soluzione è quella sotto a destra.

L'esercizio che viene dopo (8) è ancora una semplice ripetizione di un disegno costruito mediante un ciclo, mentre l'**esercizio 9** chiede nuovamente di ripetere per 12 volte una figura geometrica, per completare la corolla di un fiore. Questa volta si tratta di un rombo, figura che richiede un ciclo lievemente più complesso, dal momento che i suoi quattro angoli non sono tutti uguali. Il codice per disegnare il rombo è già pronto nell'area di lavoro, e si tratta solo di aggiungere il ciclo più esterno per ripeterlo il numero di volte desiderate e spostarsi ogni volta nella posizione corretta. La soluzione è sotto a destra.

Nell'**esercizio 10** si trova già pronta una sequenza di blocchi contenente un ciclo, sequenza che disegna una figura non facile da capire semplicemente guardando il codice stesso. Si suggerisce quindi di eseguire subito il codice che si tra già pronto, in modo da capire meglio cosa accade quando poi questo codice verrà annidato nel ciclo più esterno. Sotto a sinistra si trova il risultato di una ripetizione del codice in grigio, mentre a destra è la soluzione dell'esercizio.

Con l'esercizio successivo (11) cambia la figura di base usata nel ciclo più interno. Questa volta è un cerchio, che viene semplicemente disegnato ripetendo 360 volte lo spostamento di 1 pixel e la rotazione di 1 grado[19]. Il ciclo più esterno viene usato per disegnare 4 cerchi in tutto.

Questo schema di base per disegnare viene usato due volte, con lievi variazioni, per disegnare la figura (vedi sotto a sinistra) dell'**esercizio 12**, mediante il codice che si trova già pronto nell'area di lavoro.

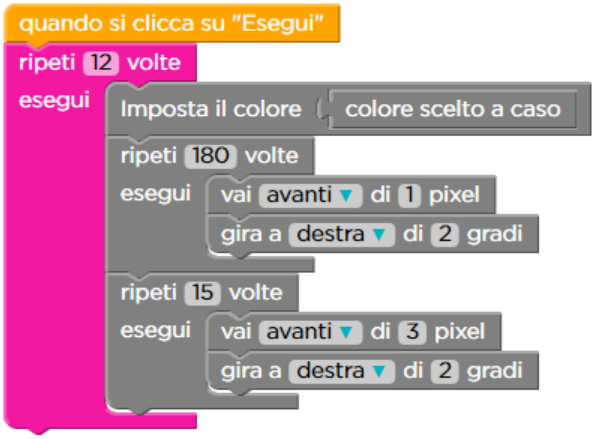
I cerchi più piccoli vengono disegnati ripetendo 180 volte lo spostamento di 1 pixel e la rotazione di 2 gradi (ottenendo così sempre la rotazione di un angolo giro necessaria completare il cerchio).

Ogni volta che si è completato il cerchio più piccolo si disegna un tratto del cerchio più grande, mediante un ciclo che ripete 15 volte la rotazione di 2 gradi (si ruota così di 30 gradi).

Lo studente deve completare questo codice già pronto annidando il tutto in un ciclo con 12 ripetizioni. In tal modo si disegnano 12 cerchi piccoli e si completa la rotazione (12x30=360) del cerchio più grande.

L'intero programma è mostrato sotto a destra.

[19] In effetti l'artista disegna quindi un poligono con 360 lati e non un cerchio. A seconda del livello della classe il docente può sfruttare questo fatto sia per osservare come si possa considerare la circonferenza come concetto limite del poligono sia per notare che nelle realizzazioni digitali i concetti che la matematica definisce in modo continuo vengono necessariamente discretizzati.

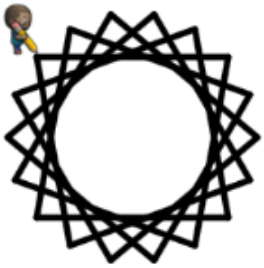


La lezione si conclude con l'**esercizio 13**, che è un esercizio a tema libero: si può quindi disegnare qualunque cosa venga in mente.

Un possibile spunto è partire dal ciclo che disegna il triangolo, ma variare l'angolo di cui si gira scegliendo un valore che non sia un sotto-multiplo di 360. In tal modo non si riesce ad ottenere un poligono regolare, perché il risultato della divisione di 360 per tale valore non è un numero intero.

Se però si ripetono le azioni di disegnare e di girare per un numero sufficiente di volte, si riesce a tornare al punto di partenza ottenendo una figura piacevole.

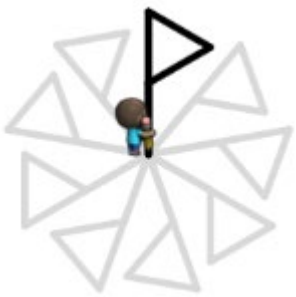
Ad esempio, scegliendo il valore di 100 gradi e ripetendo le azioni 18 volte si ottiene quanto segue.

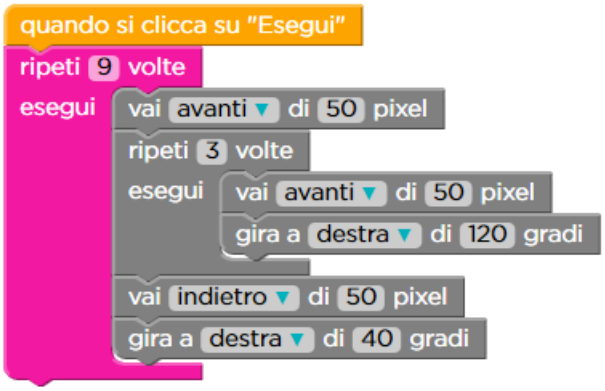


In generale il numero di ripetizioni è ottenuto dividendo il minimo comune multiplo tra il numero di gradi di cui si gira ad ogni passo e 360 gradi per lo stesso numero di gradi di cui si gira ad ogni passo. In questo caso m.c.m.(100,360)/100 = 18. Lavorando sulla base di questo schema si può ottenere, ad esempio, il disegno qui sotto, il cui codice è mostrato a fianco (le due istruzioni iniziali servono a posizionare l'artista in modo da poter realizzare l'intero disegno rimanendo sempre all'interno della tela).

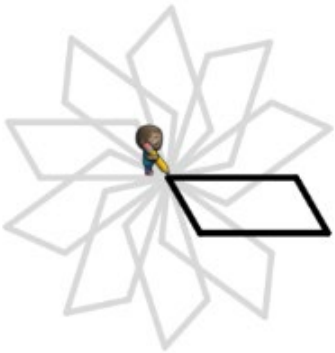
6.2.2 Corso 3: Lezione 11 (Cicli Annidati – Artista)

I primi esercizi (1, 2, 3, 4, 5) ripetono quanto già visto sul ciclo annidato invitando a ripetere mediante un ciclo un blocco di ripetizione che disegna un triangolo e ripercorrono situazioni già affrontate nel precedente paragrafo. Con l'**esercizio 6** si affronta il disegno di una figura più complessa (mostrata in grigio sotto a sinistra). Il codice che si trova già pronto permette di disegnare un bandierina (basta eseguirlo così com'è per osservare il risultato –in nero sotto a sinistra) e l'obiettivo dell'esercizio è completare la figura proposta. Si tratta quindi di individuare il numero corretto di ripetizioni per il ciclo in cui inserire il codice già pronto. La soluzione completa è presentata sotto a destra.

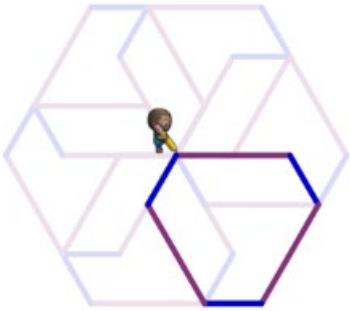


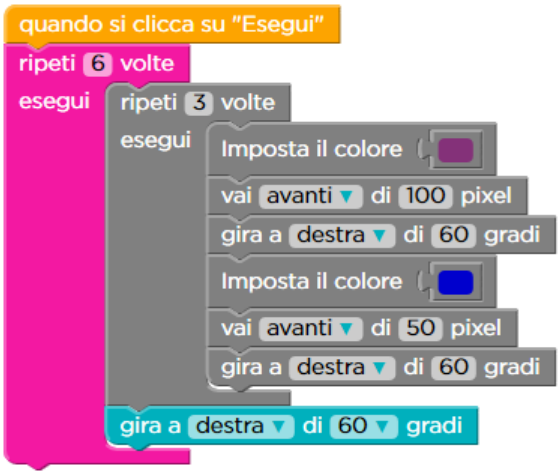


Nell'**esercizio 7** la figura disegnata dal blocco di ripetizione che si trova già pronto è un parallelogramma: anche in questo caso eseguire il codice così come si trova fa capire cosa si ottiene (figura in nero sotto a sinistra). L'obiettivo dell'esercizio è completare il codice già pronto col numero adeguato di ripetizioni, facendo attenzione a determinare di quanto deve girare l'artista una volta completato il disegno di ogni parallelogramma. La soluzione è sotto a destra.



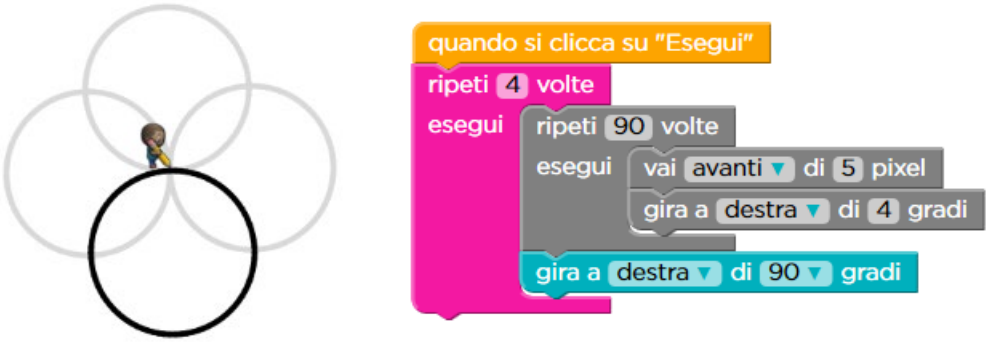
Il successivo esercizio (8) è basato sulla ripetizione di un quadrato e non presenta particolari difficoltà mentre nell'**esercizio 9** la figura base disegnata dal ciclo che si trova già pronto è particolare, poiché si tratta di un triangolo con i vertici tagliati (in colore sotto a sinistra). Ripetendola per il numero di volte suggerito (6) si ottiene un disegno piacevole, per il quale non è immediato capire come sia stato ottenuto. Il programma completo è visualizzato sotto a destra.





L'esercizio seguente (10) è un'ulteriore variazione sulla ripetizione del disegno di triangoli, mentre la figura base dell'**esercizio 11** è il cerchio, disegnato (come sempre accade nel mondo digitale) mediante approssimazione con un poligono con un numero di lati sufficientemente elevato (90, in questo caso, e si può osservare sotto a sinistra che ad occhio nudo è difficile accorgersi dell'approssimazione). L'obiettivo

dell'esercizio è agevolmente conseguito aggiungendo il numero corretto di ripetizioni e la rotazione dell'appropriato numero di gradi (codice sotto a destra).



La lezione si chiude con l'**esercizio 12** – Risposte Multiple, un esercizio di verifica. Si tratta di individuare la sequenza corretta di blocchi che disegna la figura geometrica proposta. Le possibili risposte sono programmi con le istruzioni in inglese (per le traduzioni si veda quanto riportato a pag. 31) e la soluzione corretta è mostrata qui sotto a destra.

6.3 L'evento, la sua gestione e la ripetizione infinita

In questa sezione affrontiamo un concetto fondamentale della programmazione informatica: quello di **evento** che, come implicito nel termine stesso, indica "qualcosa che accade". La sua importanza sta nel fatto che questo "qualcosa che accade" può essere la base per compiere azioni, che possono anche dipendere dalla specificità di ciò che è accaduto. In tal modo è possibile definire una sequenza di attività che non solo si snoda in modo diverso a seconda di ciò che accade ma è anche in grado di reagire agli specifici avvenimenti. Si ottiene in tal modo l'interattività tra l'esecutore e l'ambiente in cui esso opera.

Questi programmi interattivi (di cui i giochi sono l'esempio più visibile) sono quindi nella loro struttura fondamentale "guidati dagli eventi", nel senso che in essi tipicamente si decide cosa fare in risposta al verificarsi di certi eventi. In questi programmi ci sono quindi dei blocchi del tipo "*quando* accade qualcosa" e a questi blocchi si attaccano i blocchi con le azioni desiderate. Si realizza così un **gestore dell'evento**

(cioè un insieme di blocchi costituito da un blocco del tipo "*quando* accade qualcosa" seguito da uno o più blocchi di azione) che fa reagire il programma a ciò che è accaduto.

Ovviamente si continuano ad usare gli stessi concetti che abbiamo visto in altre lezioni, ad esempio l'istruzione "ripeti".

Per lavorare sul concetto di evento e su come usarlo per realizzare programmi interattivi si usano sia la lezione 17 del corso 2 (Laboratorio) che la lezione 16 del corso 2 (Flappy).

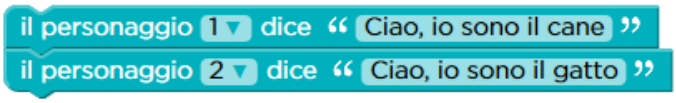
6.3.1 Corso 2: Lezione 17 (Laboratorio – Crea una Storia)

La lezione 17 del corso 2 ha come obiettivo la costruzione di un propria storia. Si impara quindi ad usare i gestori di eventi per gestire i click del mouse e le collisioni tra oggetti. Ci sono gli eventi relativi al fatto che i personaggi si toccano, si clicca sul personaggio e si preme sulle frecce.

L'**esercizio 1** presenta uno dei protagonisti, un cane, a cui bisogna far dire "Ciao a tutti". L'unica istruzione disponibile è "pronuncia ..." ed il suo parametro è ciò che il cane dirà quando si clicca su "Esegui".

L'**esercizio 2** presenta il secondo protagonista, un gatto, ed entrambi questa volta devono dire qualcosa. L'unica istruzione disponibile è più complessa, perché ha due parametri, il primo per individuare il personaggio che deve parlare (1 indica il cane, 2 il gatto), il secondo per specificare il testo da pronunciare. La soluzione è sotto a destra.

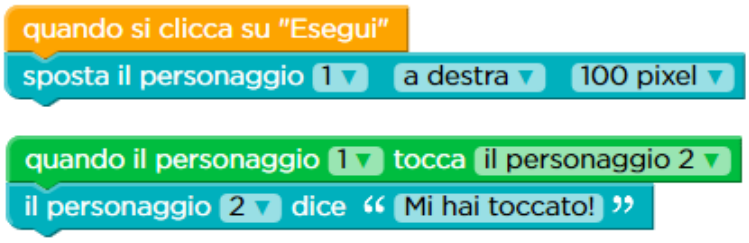




Con l'**esercizio 3** si introduce l'istruzione di spostamento per i personaggi, anche essa con più parametri, come quella che li fa parlare. In questo sono addirittura tre, il primo che indica il personaggio, il secondo la direzione del movimento ed il terzo la quantità dello spostamento. L'obiettivo è far arrivare il cane (personaggio 1) fino al gatto. La distanza da percorrere è 100 pixel.

Nell'**esercizio 4** viene introdotto il primo esempio di "gestore di evento". Il blocco che realizza un gestore di evento è sempre di colore verde ed inizia con la parola "quando". Il resto del blocco descrive il tipo di evento che il blocco dovrà gestire. Il gestore di evento in questo caso è

L'obiettivo di quest'esercizio è far arrivare il cane fino al gatto, come fatto nel precedente, e far sì che, quando questo avviene, il gatto dica qualcosa. L'esercizio chiede quindi di realizzare un gestore di evento in grado di reagire all'evento di due personaggi che si toccano. Tale gestore di evento si realizza attaccando gli opportuni blocchi al blocco verde "quando". La soluzione complessiva è quindi la seguente.



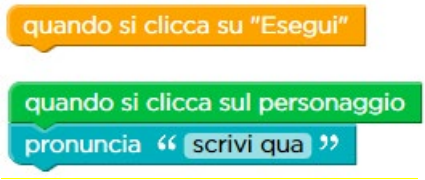
È interessante notare che il programma in questo caso è costituito da due parti scollegate. Quella che abbiamo visto finora, costituita dai blocchi attaccati al blocco "quando si clicca su Esegui" e quella costituita dai blocchi attaccati al blocco "quando il personaggio 1 tocca il personaggio 2".

Notiamo poi che anche il blocco "quando si clicca su Esegui" è un gestore di evento. È un gestore di tipo particolare, perché gestisce quel particolarissimo evento con il quale diciamo al sistema di mandare in esecuzione il nostro programma, e per questo è rappresentato in colore arancione. Infatti è questo gestore di evento che consente agli altri gestori di evento di diventare attivi. Se non cliccassimo su "Esegui" gli altri gestori di evento del programma non potrebbero realizzare alcun'azione, perché non sarebbero in grado di accorgersi degli eventi ai quali devono reagire.

Inoltre – e questo è vero in generale per qualunque blocco gestore di eventi, il blocco "quando il personaggio …" è staccato dagli altri perché entra in gioco indipendentemente da dove è arrivata l'esecuzione nel resto del programma. Questo sarà ancora più chiaro nel seguito.

Con l'**esercizio 5** si introduce un nuovo personaggio, un polpo – vedi sotto a sinistra, ed un secondo esempio di "gestore di evento". È raffigurato in verde come il precedente e serve a far sì che il programma possa interagire con l'utente rispondendo alle sue azioni (sotto a destra). La specifica azione che questo gestore di evento consente di trattare è il cliccare col mouse sul personaggio stesso.

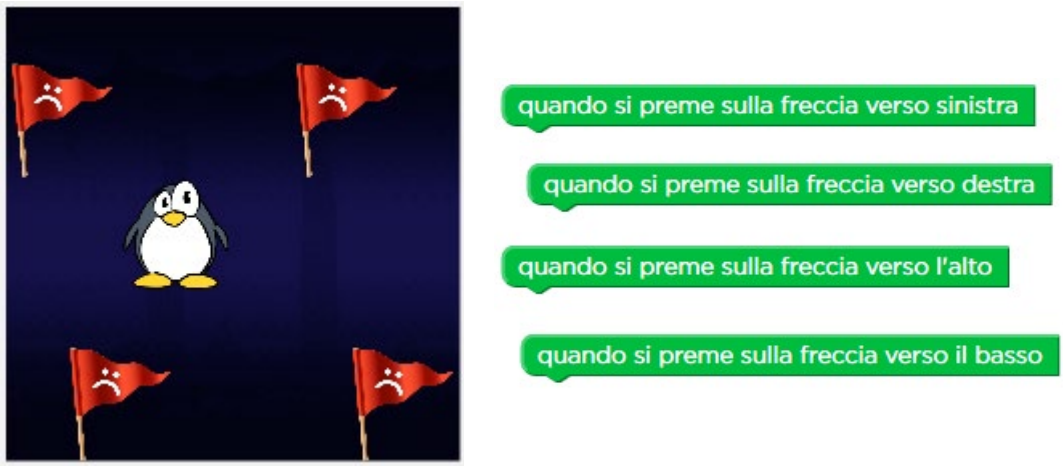
L'obiettivo dell'esercizio è far sì che il polpo dica "Ciao" quando si clicca su di esso. La soluzione quindi è quella mostrata qui sotto a sinistra. Osservate che non c'è niente attaccato al blocco "quando si clicca su Esegui", perché il programma non deve far niente in conseguenza di ciò, se non aspettare che si clicchi sul polpo.



Come detto precedentemente, il blocco arancione, oltre ad eseguire le azioni eventualmente attaccate ad esso, serve a rendere "attivi" gli altri blocchi in verde che gestiscono gli eventi.

In questo caso, quindi, quando si clicca su "Esegui", non viene eseguita alcuna azione (perché non ci sono istruzioni attaccate al blocco arancione), viene attivato l'unico gestore di evento presente ("quando si clicca sul personaggio") ed il programma, appunto, aspetta – senza far niente – che l'evento accada[20]. Quando si clicca sul polpo, allora le istruzioni attaccate al relativo gestore di evento vengono eseguite.

L'**esercizio 6** introduce un ulteriore personaggio, un pinguino (sotto a sinistra), e nuovi gestori di eventi (a destra). Questi ultimi aggiungono nuove azioni dell'utente cui il programma può rispondere: la pressione da parte dell'utente sui tasti freccia.



L'obiettivo dell'esercizio è quello di far spostare il pinguino in tutte le direzioni in modo da fargli toccare tutte e quattro le bandierine. Per risolverlo, dovremo quindi attaccare ad ogni gestore di evento il blocco con l'istruzione appropriata. Nella "cassetta degli attrezzi" c'è un istruzione "sposta …" con un parametro selezionabile che indica la direzione del movimento. La soluzione si ottiene quindi scegliendo il parametro corretto per ogni gestore di evento.

[20] Il sistema ha un meccanismo a tempo al suo interno in base al quale, se non clicca sul polpo entro una decina di secondi, viene mostrato un messaggio di errore con l'esortazione a riprovare.

Osserviamo che il sistema non si accorge se l'istruzione attaccata ad un gestore di evento non ha il parametro corretto. In teoria, uno studente potrebbe mandare il pinguino a destra quando si clicca sul tasto "freccia a sinistra" (e viceversa). Si otterrebbe un comportamento contro-intuitivo da parte del programma ma si riuscirebbe comunque a spostare il pinguino su tutte le bandierine ed il sistema considererebbe l'esercizio risolto. Sta quindi all'attenzione del docente verificare la correttezza di questo aspetto.

Osserviamo anche che in questo esercizio il blocco arancione "quando si clicca su Esegui" non è proprio presente, visto che – anche in questo caso – il programma non deve eseguire alcuna azione quando si clicca su "Esegui" se non rendere attivi i quattro gestori di evento.

Notiamo infine che l'utente può sia premere fisicamente i quattro tasti con l'immagine della freccia presenti sulla tastiera, sia cliccare sulle immagini di tali tasti freccia presenti sotto la scena col pinguino accanto al pulsante "Esegui".

A questo punto è opportuna una riflessione relativa alla terminazione dei programmi che scriviamo. Tutti i programmi che abbiamo scritto fino a questo punto vengono eseguiti o fino al momento in cui incontrano un'istruzione che non può essere eseguita (p.es., raccogliere il nettare quando non si è sopra un fiore) oppure – se tutte le istruzioni possono essere svolte senza problemi – fino al momento in cui eseguono l'ultima istruzione della sequenza di blocchi attaccata a "quando si clicca su Esegui". Dopo di ciò, il sistema verifica se l'obiettivo dell'esercizio è stato raggiunto o meno e risponde di conseguenza.

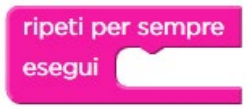
In questi ultimi due esercizi appena discussi (5 e 6), però, non vi sono istruzioni connesse al blocco arancione "quando si clicca su Esegui". Cosa accade quindi? In questi casi è all'opera ancora una volta un meccanismo basato su eventi, che non è esplicitato mediante i blocchi di codice nell'area di lavoro ma è presente all'interno del sistema. Questo meccanismo aspetta il verificarsi di uno tra due eventi: il primo evento è raggiungere l'obiettivo dell'esercizio: cliccare sul polpo (nell'esercizio 5), toccare tutte le bandierine (nel 6). Il secondo evento è la scadenza di un termine prefissato di circa una decina di secondi che il sistema lascia all'utente affinché compia le azioni che servono per raggiungere tale obiettivo. In altre parole, per una decina di secondi il sistema aspetta che l'utente risolva l'esercizio: se questo non accade il sistema decide che l'esercizio non è stato risolto.

Tutto questo è possibile perché vi è un obiettivo quantificabile in modo finito e pertanto misurabile: cliccare una volta, toccare quattro bandierine. Se avessimo un gioco in cui raggiungere un certo punteggio, l'obiettivo sarebbe tale valore predeterminato. In tutti questi casi il sistema può accorgersi del raggiungimento della mèta desiderata. Ma se vogliamo costruire un gioco in cui poter giocare quanto si vuole incrementando il punteggio senza un traguardo misurabile?

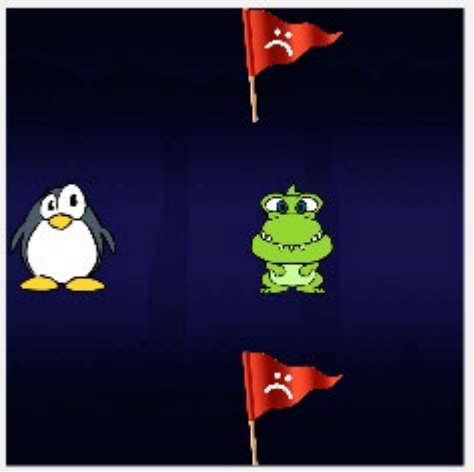
Questa osservazione motiva la necessità di introdurre un nuovo concetto, che può essere visto come una variazione del ciclo, anche se si tratta di una variazione molto

particolare. Stiamo parlando del concetto di **ciclo infinito**, rappresentato in pratica dal blocco con l'istruzione **ripeti per sempre**. Questo blocco permette di ripetere le istruzioni in esso incluse senza mai fermarsi. Accoppiato ai meccanismi di gestione degli eventi consente di realizzare storie e videogiochi che possono anche non terminare mai. In altre parole, quando nell'area di lavoro si trova un blocco "ripeti per sempre", il sistema ripete in continuazione senza mai arrestarsi le istruzioni contenute al suo interno. In realtà questo non è completamente vero, per due motivi: il primo è che se una di queste istruzioni non può essere eseguita (p.es., "fai il miele" quando l'ape non è sopra un favo) il sistema interrompe comunque l'esecuzione del programma; il secondo è che possono esserci altri meccanismi per far interrompere il programma (p.es., il verificarsi di un altro evento oppure un'esplicita istruzione "fine del gioco", vedi il prossimo paragrafo 6.3.2 a pag. 124).

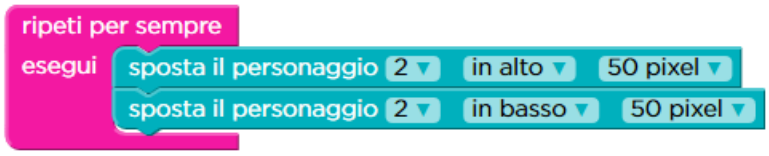
Nell'**esercizio 7** (ambientazione mostra qui a fianco) viene quindi introdotta la nuova istruzione



insieme ad un nuovo personaggio, un dinosauro, che deve essere spostato in continuazione in su e in giù. In effetti, ai fini dell'obiettivo dell'esercizio, che è quello di toccare entrambe le bandierine, tale nuova istruzione non servirebbe.



Però, se nell'area di lavoro (dove si trovano i gestori di eventi che gestiscono gli spostamenti del pinguino – che comunque non servono per questo esercizio) si inseriscono all'interno del blocco "ripeti per sempre" le istruzioni di movimento per il dinosauro (personaggio 2) con un valore del parametro di spostamento insufficiente a fargli toccare le bandierine (vedi a fianco) si riscontra che il dinosauro si muove in continuazione in su e in giù. In effetti, anche in tal caso, dopo una decina di secondi il sistema interviene per dichiarare l'esercizio non risolto: si tratta del meccanismo di eventi presente all'interno del sistema, che abbiamo precedentemente descritto e che è sempre attivo.



Per risolvere quest'esercizio bisogna dunque far spostare il dinosauro di almeno 100 pixel rispetto alla sua posizione di partenza. Di conseguenza la soluzione è quella a fianco raffigurata, o una simmetrica che lo fa spostare prima in basso e poi in alto, o variazioni di queste con valori ancora maggiori per gli spostamenti.

È quindi chiaro che in questo esercizio non sarebbe necessario l'uso dell'istruzione "ripeti per sempre", dal momento che il sistema considera l'esercizio risolto non appena il dinosauro tocca ambedue le bandierine. Siamo infatti in uno dei casi in cui lo scopo dell'esercizio è un obiettivo misurabile.

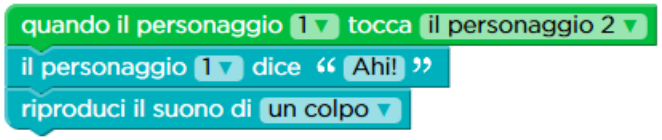
Un'osservazione finale: anche in questo esercizio non è presente il blocco "quando si clicca su Esegui": si tratta però solo di una scelta fatta per non affollare l'area di lavoro. A tutti gli effetti è come se esso fosse presente e il blocco "ripeti per sempre" fosse attaccato sotto di esso.

Nell'**esercizio 8** il blocco "ripeti per sempre" è invece necessario. In questo scenario sono presenti un pinguino ed un dinosauro e l'obiettivo è far sì che il pinguino tocchi il dinosauro, dica "Ahi!" e si senta il suono di un colpo. Nell'area di lavoro sono già pronti i gestori di evento per gestire gli spostamenti del pinguino.

Per rendere l'esercizio più interessante, però, il dinosauro non rimane immobile ma viene spostato in continuazione. Il codice qui a fianco è anch'esso già pronto nell'area di lavoro. Il blocco "ripeti per sempre" è quindi necessario perché si vuole ripetere una se-

quenza di azioni un numero di volte arbitrario, nell'attesa che si verifichi un qualche altro evento. Nel caso specifico, si vuole far spostare il dinosauro in alto e in basso nell'attesa che il pinguino arrivi a toccarlo (o meglio, che l'utente sposti il pinguino con i tasti freccia fino a fargli toccare il dinosauro).

Da un punto di vista pragmatico, in questo specifico caso – in cui abbiamo visto che è comunque all'opera il meccanismo ad eventi implicito nel sistema che interviene dopo circa una decina di secondi – si potrebbe usare un'istruzione "ripeti ... volte" con un parametro sufficientemente elevato. È però chiaro che, da un punto di vista generale, può essere difficile o addirittura impossibile determinare il numero di ripetizioni necessario, quando si vuole ripetere una sequenza di azioni nell'attesa che si verifichi un qualche altro evento. Ed è quindi proprio questa la situazione nella quale è molto comodo avere a disposizione un ciclo infinito, anche se non è assolutamente indispensabile come nella situazione di un obiettivo non misurabile precedentemente discussa. Concludiamo la discussione presentando qui a fianco la soluzione dell'esercizio.



Nell'**esercizio 9** si trovano gli stessi due personaggi e lo stesso codice dell'esercizio precedente per muovere in continuazione il dinosauro e per far spostare il pinguino mediante i tasti freccia. Però lo scenario si fa più complesso, sia perché si aggiunge anche un polpo (personaggio 3), sia perché l'obiettivo cambia: il pinguino in questo caso deve andare a sbattere contro il polpo e, quando questo accade, deve essere aggiunto un punto al punteggio. Vi è infatti a disposizione un nuovo blocco, che va appunto usato per risolvere l'esercizio, aggiungendolo ad un nuovo gestore di evento, utilizzato proprio per trattare il caso della collisione tra pinguino e polpo. Non appena viene aggiunto il punto l'esercizio termina.

L'**esercizio 10** ripropone lo stesso scenario e lo stesso codice del precedente, ma aggiunge due blocchi (vedi a fianco) per variare lo sfondo su cui si muovono i personaggi e la loro velocità.

Normalmente questi **blocchi di impostazione** si attaccano al blocco arancione "quando si clicca su Esegui", poiché servono per predisporre all'inizio l'ambientazione della storia o del gioco, ma possono essere anche usati in altro modo per variarne l'andamento in modo divertente: si provi, ad esempio, ad attaccare il blocco "imposta uno sfondo ..." con parametro "scelto a caso" ad altri gestori di evento.

Con l'introduzione dei blocchi di impostazione (o **di configurazione**) sono stati forniti tutti i concetti essenziali per creare una storia o un gioco.

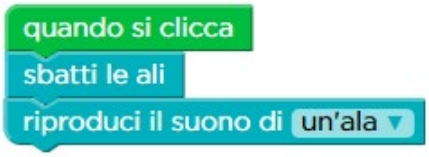
Nell'**esercizio 11**, in cui si può dare libero sfogo alla fantasia, dal momento che è un esercizio a tema libero, non vi sono infatti nuovi concetti, ma solo un arsenale più vasto di istruzioni dello stesso tipo. In aggiunta a quanto già visto, vi sono i blocchi per aspettare una certa quantità di tempo, per fermare un personaggio o farlo svanire o impostarne l'umore, per posizionare i personaggi in punti prefissati della scena, per far lanciare ai personaggi diversi tipi di proiettili nelle varie direzioni, per far sparire o rimbalzare il proiettile lanciato[21], per predisporre una schermata introduttiva.

Per un esempio che può servire da ispirazione si può guardare il gioco "Acchiappa il pipistrello!", disponibile all'indirizzo https://studio.code.org/c/105686966[22].

6.3.2 Corso 2: Lezione 16 (Flappy)

La lezione 16 del corso 2 ha come obiettivo la costruzione di una propria versione del gioco Flappy. Si tratta di un gioco nel quale bisogna tenere un uccellino (Flappy[23]) in volo mentre gli si fanno incontro una serie di ostacoli in modo da attraversarli senza andare a sbattere contro di essi. Vanno quindi usati i gestori di eventi per gestire i click del mouse e quanto succede all'uccellino mentre vola.

Nell'**esercizio 1** è presente solo il gestore per l'evento del click (che può avvenire dovunque, purché all'interno dell'area in cui vola Flappy). Al bordo superiore di tale area, in corrispondenza dell'uccellino, è presente l'immagine di un bersaglio e l'obiettivo dell'esercizio è far volare Flappy fino ad esso. Questo si ottiene con il codice mostrato sotto (in cui è stato aggiunto anche un blocco per sonorizzare l'azione dello sbattere di ali dell'uccellino) e cliccando con una frequenza sufficientemente elevata dopo aver mandato in esecuzione il programma cliccando su "Esegui".



[21] Dal momento che i blocchi di istruzione vengono presentati nella versione corrispondente alla lingua scelta per l'interazione con l'ambiente senza cambiare la struttura della frase, il blocco che fa rimbalzare o sparire i proiettili dice letteralmente in italiano – nel caso, ad esempio, si voglia far sparire un proiettile che è una palla di fuoco blu – "fai una palla di fuoco blu con sparizione" invece di "fai sparire la palla di fuoco blu", perché rimane la struttura della frase in inglese.

[22] Si ringrazia Michael Lodi per l'ideazione e la realizzazione di "Acchiappa il pipistrello!".

[23] "To flap" è il verbo inglese che vuol dire "sbattere le ali".

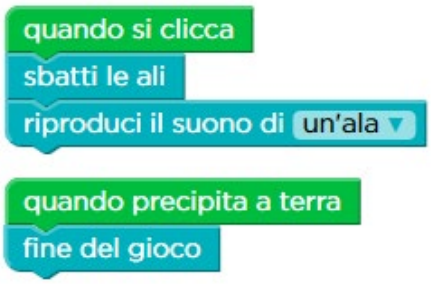
È interessante notare due aspetti: il primo è che nell'area di lavoro è assente il blocco arancione "quando si clicca su Esegui" e che il comportamento del sistema è lievemente diverso dal solito. Abbiamo visto nel precedente paragrafo 6.3.1 "Corso 2: Lezione 17 (Laboratorio – Crea una Storia)" a pag. 117 che l'assenza del blocco arancione vuol dire che nel momento in cui si clicca su "Esegui" il sistema non compie altre azioni se non attivare i gestori di evento che poi eventualmente intervengono se si manifestano gli eventi che essi devono gestire.

In questo caso, quindi, poiché l'uccellino è soggetto alla forza di gravità, subito dopo aver cliccato su "Esegui" Flappy dovrebbe cadere verso il basso a causa della gravità. Invece il sistema visualizza due scritte ("Get Ready!" e "Tap or Click") che indicano di cliccare (o toccare lo schermo, nel caso in cui l'utente stia usando un dispositivo mobile come uno smartphone o un tablet) e soltanto dopo il primo click inizia l'esecuzione vera e propria in cui l'uccellino viene spinto verso il basso dalla forza di gravità e tenuto in volo dagli sbattiti di ali che avvengono ad ogni clic.

In altre parole, in questo specifico caso, dopo aver cliccato su "Esegui" il programma non è davvero iniziato e la sua partenza effettiva avviene con il primo click. È solo con questo primo click vengono attivati i gestori di evento (uno solo in questo caso, cioè "quando si clicca") e l'azione della forza di gravità. Successivamente, pertanto, ad ogni click viene eseguita l'azione "sbatti le ali" che fa sì che l'uccellino riceva, come nella realtà, un impulso che lo spinge in alto di una certa quantità. Contemporaneamente la forza di gravità sposta col tempo l'uccellino verso il basso. Se la frequenza del click è troppo bassa Flappy va a sbattere sul terreno. Se è più elevata si mantiene in volo e se si clicca con una frequenza sufficientemente elevata Flappy si solleva abbastanza da raggiungere il bersaglio al bordo superiore e l'esercizio viene completato con successo.

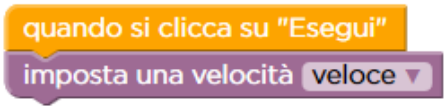
Il secondo aspetto cui prestare attenzione è che in questo caso il sistema non ha in esecuzione al suo interno un meccanismo a tempo in base al quale, se non si arriva sul bersaglio entro una decina di secondi, viene mostrato un messaggio di errore con l'esortazione a riprovare. L'esercizio può andare avanti un tempo indefinito con l'uccellino che viene tenuto in volo senza toccare il bersaglio. Il sistema però termina comunque l'esercizio (con un messaggio di errore) se, a causa di click con una frequenza troppo bassa, la forza di gravità ha la meglio e Flappy tocca il terreno.

Per superare l'**esercizio 2** (che ha la stessa ambientazione del precedente, tranne che il bersaglio è posto in basso sul terreno invece che in alto) è necessario aggiungere l'istruzione "fine del gioco" al gestore di evento che interviene quando Flappy tocca terra (vedi a fianco). Se non viene fatto e l'uccellino tocca il terreno il sistema interviene per avvisare che l'esercizio non è stato svolto correttamente.

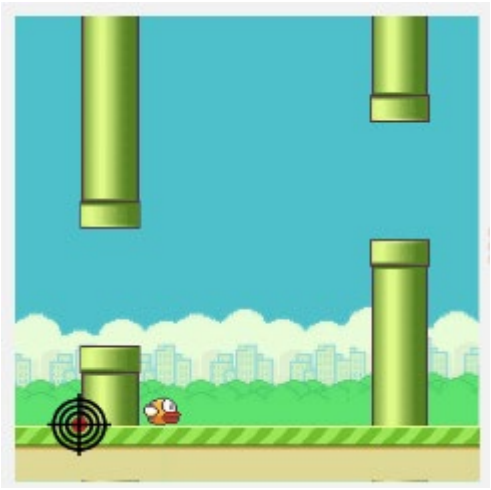


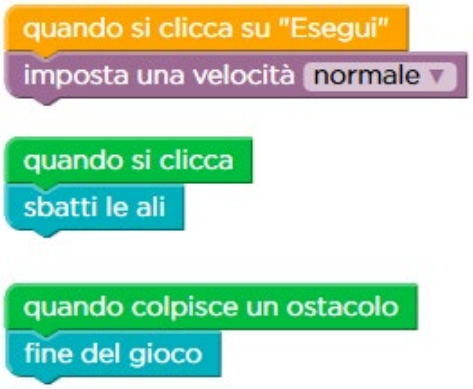
Con l'**esercizio 3** viene visualizzato nuovamente il blocco "quando si clicca su Esegui", dal momento che adesso è disponibile un blocco di impostazione (sono stati visti per la prima volta nell'esercizio 10 della lezione precedente "Corso 2: Lezione 17 (Laboratorio – Crea una Storia)" a pag. 123). Questi blocchi, che servono per configu-

rare l'ambientazione della storia o del gioco, sono spesso eseguiti all'inizio del codice in modo da definire lo scenario di svolgimento una volta per tutte. Nel caso specifico il blocco, raffigurato in viola, permette di variare la velocità orizzontale con cui si muove Flappy, verso i bersagli che a partire dal prossimo esercizio incontrerà durante il suo volo. Notiamo che anche in questo esercizio non c'è un meccanismo a tempo di interruzione e, dal momento che non è presente neanche il gestore di evento "quando precipita a terra", l'esercizio può andare avanti indefinitamente.



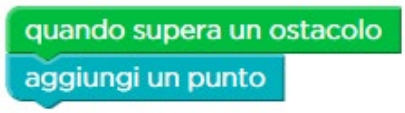
Come preannunciato sopra, nell'ambientazione dell'**esercizio 4**, mostrata sotto a sinistra, Flappy deve passare attraverso dei tubi che ostacolano il suo volo. Si trova già pronto il codice per farlo volare e predisporre la velocità iniziale, e si chiede solo di far finire il programma quando colpisce un ostacolo. La soluzione è quindi semplicemente quella mostrata sotto a destra.





Osserviamo che il sistema interrompe comunque l'esercizio anche se Flappy supera il primo ostacolo.

Con l'**esercizio 5** l'interazione con il gioco diventa più sofisticata perché si introduce la possibilità di gestire un punteggio che si incrementa di uno ogni volta che Flappy supera un ostacolo. È disponibile un ulteriore gestore di eventi "quando supera un ostacolo", cui bisogna attaccare il blocco di azione "aggiungi un punto" (vedi a fianco). In tal modo non appena l'uccellino supera l'ostacolo il contatore del punteggio verrà incrementato. In questo specifico esercizio, osserviamo che il sistema termina l'esecuzione del programma non appena si fa il primo punto e lascia scorrere l'esercizio se invece Flappy sbatte contro un ostacolo o sul terreno.



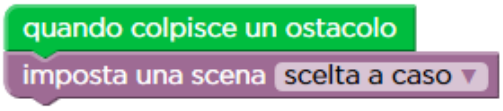
Un'ulteriore sofisticazione viene introdotta con l'**esercizio 6**, nel quale il blocco di azione "sbatti le ali" viene esteso col parametro relativo al numero di volte che le ali vengono sbattute (vedi a fianco). Maggiore è questo numero, maggiore è l'impulso

verso l'alto che l'uccellino riceve. L'obiettivo anche in questo caso è quello di far sì che Flappy riesca a segnare il primo punto. In tal caso l'esecuzione termina (ma prosegue indefinitamente se sbatte contro un ostacolo o sul terreno).

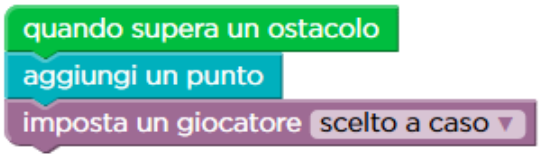
Con l'**esercizio 7** l'interazione diventa sempre più realistica, dal momento che si trovano già pronti nell'area di lavoro tutti i gestori degli eventi rilevanti (superare un ostacolo o sbatterci contro, cadere a terra, cliccare), con le relative azioni tipiche già attaccate. Si offre in più la possibilità di predisporre una scena di sfondo alla partenza dell'esercizio. Ad esempio, qui a fianco viene visualizzato cosa accade attaccando il blocco di configurazione "imposta una scena Sottomarina" al blocco "quando si clicca su Esegui".

È anche interessante osservare che adesso lo studente può cambiare il gioco in modo completamente arbitrario. Ad esempio, se non vuole mai perdere, può staccare i blocchi di azione "fine del gioco" che sono attaccati ai gestori di evento "quando precipita a terra" e "quando colpisce un ostacolo": così facendo il gioco andrà avanti indefinitamente.

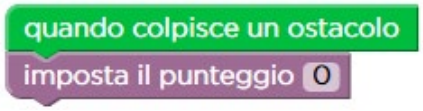
L'**esercizio 8** introduce un ulteriore impostazione, quella relativa al tipo di uccellino da usare nel gioco. Tale configurazione, così come le altre, può essere definita alla partenza del gioco (attaccandola al blocco "quando si clicca su Esegui") oppure può essere definita, per divertirsi rendendo l'esercizio più vario, ad un qualunque blocco gestore di eventi. Ad esempio, con la scelta indicata a fianco ogni volta che si colpisce un ostacolo lo sfondo del gioco cambia in modo casuale



E con quella mostrata qui a destra, ogni volta che viene superato un ostacolo, oltre ad acquisire un punto l'uccellino subisce una metamorfosi.



Nell'**esercizio 9** la sofisticazione del gioco aumenta, poiché diventa possibile, mediante il blocco "imposta il punteggio 0", riportare il punteggio a zero al verificarsi di un certo evento. Una scelta che trasforma l'esercizio in una vera sfida di resistenza e bravura è quella di attaccarlo al gestore dell'evento "quando colpisce un ostacolo", in



modo da poter definire un obiettivo di gioco consistente nel raggiungere il punteggio più alto possibile.

La lezione si conclude con l'**esercizio 10** nel quale si può realizzare un gioco in modo completamente arbitrario, perché è possibile configurare sia aspetti di ambientazione, quali il tipo di ostacoli che devono essere superati o il tipo di terreno da visualizzare, sia elementi che incidono sulla difficoltà del gioco, come la distanza dell'apertura attraverso cui passare e l'intensità della forza di gravità.

Per un esempio che può servire da ispirazione si può guardare quello disponibile all'indirizzo https://studio.code.org/c/74536128 [24].

[24] Questa versione di Flappy è stata ideata e realizzata dall'autore.

Volume pubblicato nel mese di Ottobre 2020