Italiano a scuola, 2 (2020), pp. 171-200
https://doi.org/10.6092/issn.2704-8128/10895
https://italianoascuola.unibo.it
ISSN 2704-8128

Scrivere una fiaba col *coding*: una sperimentazione didattica con Scratch

GABRIELE CATANI

Creating a fairy tale through computer coding: a didactic experiment with Scratch

This study examines the repercussions of computer coding activities, introduced several years ago in the Italian schools, for linguistic education. In particular, the article describes an experiment conducted in a lower secondary school, where a fairy tale, which had already been analysed by Propp as a linguistic algorithm, was coded in its single events by using a programming language with blocks in the programming environment Scratch. Subsequently, the instructions contained in the source code in blocks were "translated" into Italian and turned into a narrative by the students.

Il contributo è dedicato alle ricadute per l'educazione linguistica delle attività di *coding*, da qualche anno introdotte nella scuola italiana. Si dà conto in particolare di una sperimentazione condotta in una scuola secondaria di primo grado sulla fiaba, già analizzata da Propp come una sorta di un algoritmo linguistico, che è stato quindi possibile programmare nei suoi singoli eventi attraverso un linguaggio di programmazione a blocchi attraverso l'ambiente di programmazione Scratch; successivamente le istruzioni contenute nel codice sorgente a blocchi sono state "tradotte" in lingua italiana e rese in forma narrativa dagli studenti.

GABRIELE CATANI (catani.gabriele1@gmail.com) è laureato in Scienze economiche per l'ambiente e la cultura presso l'Università Cattolica di Milano e in Italianistica, Culture Letterarie Europee, Scienze Linguistiche presso l'Università di Bologna, con una

tesi sull'uso di attività del *coding* per l'apprendimento linguistico. Appassionato di approcci interdisciplinari alla conoscenza, i suoi principali interessi vertono su linguistica e letteratura italiana e sull'applicazione delle nuove tecnologie alla didattica.

1. Linguaggio di programmazione e linguaggio naturale: analogie per l'apprendimento linguistico

1.1. Il codice come lingua

1.1.1. *Coding* e materie umanistiche

Si definisce *coding* la fase finale del processo di programmazione informatica, che consiste nello scrivere il codice sorgente in un dato linguaggio che il computer riesce a interpretare. In ambito scolastico odierno, fare *coding* attraverso l'utilizzo di determinati strumenti ha assunto un diverso significato, legato alla possibilità di sviluppare negli studenti un insieme di abilità legato al lavoro didattico sul cosiddetto "pensiero computazionale". Si è giunti a questo concetto, ideato da Seymour Papert per primo negli anni Novanta, a partire da un approccio costruzionista all'apprendimento trasversale a tutte le materie, come oggi sono trasversali a ogni disciplina scolastica le attività di *coding* e robotica proposte a scuola o in strutture specializzate. Tuttavia, non si hanno ancora dati su quanto l'apprendimento del pensiero computazionale influenzi le prestazioni scolastiche; inoltre, per quanto riguarda l'Italia, le esperienze condivise in rete di attività di *coding* svolte a scuola sono poche e legate soprattutto a iniziative di singoli insegnanti.

Per quanto riguarda le discipline umanistiche ci sono alcuni esempi di uso di ambienti di programmazione come Scratch e Minecraft nell'ambito di varie discipline scolastiche attraverso lo *storytelling* o il gioco: entrambe sono piattaforme che impiegano un linguaggio di programmazione facilitato che permette a bambini e ragazzi di comprendere il pensiero computazionale[1]. In Italia, nel 2019, il Progetto "Programma il Futuro" ha bandito il concorso aperto a tutte le scuole "Programma una storia", allo scopo di incentivare gli studenti a "programmare" un'opera letteraria[2].

C'è però qualcosa di più profondo che potrebbe legare l'insegnamento della lingua italiana al *coding*: il codice sorgente e la lingua italiana (come qualsiasi altra lingua naturale) sono due diversi tipi di linguaggio ma condividono alcune interessanti caratteristiche. Stefano Penge, autore del libro *Lingua, coding e creatività. Fare coding con le materie umanistiche* (2018) afferma che c'è stretta un rapporto stretto fra codice di programmazione e lingua:

[1] Si vedano i progetti e i post presenti nelle *community* per insegnanti e educatori di Scratch e Minecraft: http://scratched.gse.harvard.edu/index.html e https://education.minecraft.net/blog/ (ultima consultazione: 15.02.2020).

[2] https://programmailfuturo.it/progetto/archivio/concorso-2019 (ultima consultazione: 15.02.2020).

> Perché al di là della supposta maggiore facilità dello spostare blocchi anziché scrivere parole, c'è una parentela forte tra la scrittura di un codice sorgente e la scrittura di un testo di altro genere. [...] Chi progetta un programma ha in testa qualcosa che assomiglia al plot di un romanzo: immagina personaggi, scenari, colpi di scena, epiloghi felici o infelici. [...] Se è vero – e non è ancora stato dimostrato – che tutti dovrebbero saper programmare, è altrettanto vero che chi programma dovrebbe saper immaginare e scrivere una storia. Altrimenti non farebbe che mettere meccanicamente in sequenza istruzioni, e questo, ahimé, non ha mai portato da nessuna parte (Penge 2018: 41-42).

Sarebbe quindi possibile scrivere un testo narrativo in linguaggio di programmazione? Cosa accadrebbe se nell'ora di Italiano si ragionasse sulle modalità di espressione della propria lingua sfruttando come spunto di riflessione il *coding*?

1.1.2. Linguaggio di programmazione e linguaggio naturale a confronto

Il pensiero computazionale insegnato a scuola si basa su concetti propri dei linguaggi di programmazione a paradigma imperativo, in cui rientrano attualmente quelli più diffusi e utilizzati. Dato che le istruzioni vengono svolte secondo un ordine prestabilito, quindi in maniera algoritmica, si definisce un linguaggio di programmazione come la descrizione di un algoritmo. I linguaggi di programmazione di alto livello sono ritenuti linguaggi vicini a quello umano perché contengono parole di una lingua umana (quasi sempre l'inglese) e perché seguono logiche a noi familiari. I linguaggi adoperati comunemente dai programmatori sono tutti di alto livello in quanto sarebbe molto difficile per un essere umano dialogare con il computer direttamente in linguaggio macchina. Non sarebbe altresì possibile programmare direttamente in lingua naturale: diversi studi condotti sin dalla nascita del computer e recenti esperimenti hanno confermato che non è conveniente in termini di efficacia programmare adoperando esclusivamente la lingua naturale (Good, Howland 2017: 78-92).

La differenza più importante tra codici e lingua riguarda il significato delle parole in base al contesto. Nel caso dei linguaggi di programmazione, a una data costruzione sintattica corrisponde uno e un solo significato; ciò significa che le istruzioni che compongono il codice devono avere un significato univoco, mentre il linguaggio naturale è ricco di sfumature, allusioni e ambiguità: ad esempio la parola *porta* potrebbe indicare sia un nome che un verbo. Ogni linguaggio artificiale è inoltre un sistema lessicale chiuso, a meno che non vi sia la volontà esplicita di introdurre nuove parole, mentre le lingue umane sono aperte a nuove entrate lessicali e si mescolano fra loro, importando parole e modi di dire da una lingua all'altra. Le lingue, dunque, nascono, evolvono e cambiano in maniera appunto naturale nel corso del tempo.

D'altro canto, ci sono anche importanti somiglianze, la cui individuazione è stata determinante nell'ideazione della sperimentazione didattica di seguito descritta.

Innanzitutto, le caratteristiche fondamentali dei programmi costruiti mediante programmazione strutturata, una delle metodologie di progettazione e sviluppo di programmi più utilizzata, sono le stesse di cui necessita un qualsiasi testo scritto in lingua naturale: leggibilità, documentabilità (l'autore può spiegare significato e motivazione delle scelte), modificabilità e provabilità (il programma deve facilitare le attività di *testing* e *debugging*, cioè il controllo della correttezza sintattica e concettuale) (Chianese *et al.* 2017: 115). La regola fondamentale per ottenere programmi di buona qualità consiste nell'eliminare i particolari che inficiano la sua costruzione logica, e quindi la sua comprensibilità, così come accade quando si corregge una bozza di un testo scritto.

Inoltre, gli algoritmi adottano dei concetti presenti nel cervello umano e di conseguenza nel linguaggio naturale. Le strutture di controllo di sequenza, iterazione e condizione sono proprie anche del nostro modo di esprimerci. In una sequenza, le azioni sono svolte una di seguito all'altra, mentre il costrutto iterativo prevede che una certa azione debba essere eseguita un certo numero di volte o all'infinito; infine, il costrutto di selezione stabilisce che alcune azioni accadono se si verifica una certa condizione: tipicamente un linguaggio di programmazione si esprime con *if*... *then*, mentre viene introdotta una proposizione disgiuntiva nel caso in cui l'istruzione dia luogo a due azioni distinte con l'aggiunta di *else*. Si pensi a quando nella nostra lingua mettiamo in sequenza più informazioni, così che abbiamo bisogno di virgole e congiunzioni, a quando necessitiamo di ripetere un'operazione e a quando vogliamo esprimere un rapporto di causa-effetto in un periodo ipotetico.

La terza considerazione riguarda il particolare linguaggio intermediario fra uomo e macchina, detto "pseudocodice" o "pseudolinguaggio". Impiegato dai programmatori in fase di progettazione del programma da sviluppare, questo consente un confronto diretto con la lingua naturale, in quanto funge da ponte fra i due tipi di idioma. È indipendente da lessico, sintassi e semantica di uno specifico linguaggio: ciò lascia molto spazio al linguaggio naturale, poiché i programmatori pensano nella propria lingua madre. Lo pseudocodice è di grande aiuto nel processo di raffinamento, attraverso il quale si trasforma un'istruzione non elementare (un'istruzione in linguaggio naturale) in una serie di istruzioni elementari (di comprensione per l'esecutore, cioè il computer). L'algoritmo dunque viene espresso in un primo momento usando il linguaggio naturale, finché l'istruzione non elementare, ad esempio un diagramma di flusso scritto in lingua naturale, viene scomposta per raffinamenti successivi (Chianese *et al.* 2017: 106).

Le maggiori affinità fra linguaggio di programmazione e linguaggio naturale si riscontrano nei linguaggi a paradigma logico come Prolog, nato nel

1972[3]. In Prolog il problema viene rappresentato evidenziando le connessioni logiche tra *fatti*, *regole* e *obiettivi*, ma non viene indicata una procedura per risolverlo. Ciò significa che il problema viene espresso in forma logica anziché per mezzo di un algoritmo. Uno studio del 2014 pubblicato sul «Journal of Information Technology Education» (Ragonis, Shilo 2014) ha evidenziato le analogie fra la struttura di un testo argomentativo e quella del codice sorgente scritto con Prolog. Gli autori hanno notato che le abilità astratte richieste per la comprensione di testi argomentativi sono simili a quelle richieste per formalizzare problemi in un linguaggio di programmazione logica: dato che un testo argomentativo e la programmazione logica si fondano sulle stesse strutture elementari, la loro padronanza potrebbe permettere agli studenti di aumentare la loro capacità di comprensione di entrambi i tipi di testo, se si considera il codice come tale (Ragonis, Shilo 2014: 74).

1.1.3. *Coding* e linguaggio presso i laboratori didattici della Fondazione Golinelli

Mitchel Resnick (2018: 14-20), inventore di Scratch, sostiene che l'adulto dovrebbe reimparare ad apprendere come quando frequentava la scuola d'infanzia, cioè attraverso la sperimentazione per prove e errori, l'imparare facendo e sbagliando. Questa prospettiva sembra trovare qualche riscontro anche nel documento ufficiale *Indicazioni nazionali e nuovi scenari* (MIUR 2018: 13), in cui si legge che «lingua e matematica, apparentate, sono alla base del pensiero computazionale», affermazione che dà una base di fondatezza didattica alle interrelazioni fra codice e lingua naturale, oggetto di questo contributo.

Il contesto in cui è stato possibile maturare le idee qui presentate è legato a uno stage presso l'Area Scuola della Fondazione Golinelli di Bologna da ottobre 2018 a gennaio 2019, in cui è stato possibile partecipare come osservatore a diversi laboratori di *coding* rivolti a bambini e ragazzi dalla scuola dell'infanzia alla secondaria di primo grado.

Nel laboratorio *Robot e motricità*, svoltosi fra novembre e dicembre 2018 in sei incontri, l'obiettivo non è stato soltanto quello di trasmettere ai bambini i primi concetti di programmazione ma anche quello di renderli consapevoli della spazialità e delle potenzialità del loro corpo. Si è combinato cioè l'impiego dei *kit* di robotica Cubetto e Bluebot per stimolare un primo approccio al pensiero computazionale con momenti di impiego del corpo (tipicamente, un girotondo nel quale al segnale della formatrice i bambini dovevano spostarsi a sinistra, a destra o al centro). In linea con la teoria dell'*embo-*

[3] Per ulteriori informazioni su questo linguaggio di programmazione si veda ad esempio la pagina del sito dell'Università di Parma, *Un linguaggio di programmazione logica: il Prolog*, http://www.ce.unipr.it/research/HYPERPROLOG/prolog.html (ultima consultazione: 31.12.2019).

died cognition (Cacciari 2011: 69-72), si può ipotizzare che sperimentare le proprie capacità di movimento su di sé e sui robot con regolarità contribuisca a innescare, insieme ad altre esperienze, un miglioramento della capacità linguistica dei bambini, strettamente connessa quindi al controllo del corpo: si pensi, ad esempio, all'impiego di metafore legate al movimento corporeo in espressioni come *sale la tensione* o *crollano i prezzi* (Oliverio 2017: 32).

E proprio alle metafore fanno spesso ricorso i formatori dei laboratori per spiegare i concetti computazionali: ad esempio, per insegnare ai bambini a usare i blocchi della serie di comandi "controllo", si spiegava che *i blocchi con la bocca aperta devono mangiare altri blocchi*; inoltre, per far comprendere che Scratch rappresenta un asse cartesiano, spesso si ricorreva all'analogia con il gioco da tavolo detto *battaglia navale*.

Il ruolo chiave del linguaggio naturale ai fini della comprensione del *coding* è stato però messo in luce soprattutto nell'ambito del corso di formazione rivolto a insegnanti *Making e coding alla Scuola dell'Infanzia*, svoltosi presso l'Opificio Golinelli a dicembre 2018. Una delle esercitazioni ha fatto lavorare le partecipanti a gruppi di tre: in ogni gruppo ognuna aveva un ruolo: il programmatore, il robot e il saggio, il quale aveva il compito di fare da intermediario fra i primi due. Il robot era posto di spalle rispetto al programmatore, il quale poteva adoperare esclusivamente istruzioni verbali affinché il robot riproducesse esattamente il disegno che il programmatore aveva in mano. In questa e in altre attività, il linguaggio a disposizione di chi impersonava il programmatore era sempre limitato: ad esempio in un'occasione si poteva fare uso solamente di gesti. L'obiettivo è stato quello di far comprendere cosa significhi comunicare con un codice di istruzioni rigido e univoco, diverso dalla lingua naturale per possibilità di espressione, ma molto preciso, al fine di giungere a un risultato.

In ultima analisi, il linguaggio naturale condivide con il linguaggio di programmazione la stessa funzione: attraverso una catena di istruzioni regolate da una determinata sintassi è possibile realizzare prodotti finiti, come ad esempio una fiaba, oggetto della sperimentazione didattica di seguito descritta.

2. La sperimentazione didattica presso la Scuola Media "C. Pepoli" di Bologna

2.1. L'ipotesi di partenza

2.1.1. Fiaba e programma

Sulla base delle considerazioni illustrate nel paragrafo precedente, si è giunti alla conclusione che è possibile confrontare la struttura di un testo

narrativo appartenente a un certo genere letterario con la struttura, seppur abbozzata, di un ipotetico codice sorgente.

Uno spunto decisivo è stato fornito da un filmato prodotto dalla RAI e disponibile in rete in cui Marco Maiocchi, docente di Scienza dell'informazione presso l'Università Statale di Milano, paragona il metodo strutturale di Vladimir Jakovlevič Propp al processo seguito da un programmatore quando scrive un codice. Maiocchi afferma che Propp voleva dimostrare come una fiaba potesse essere generata attraverso una determinata sequenza di eventi narrativi, i "movimenti" nella traduzione italiana (Propp 2000: 98):

> Un linguaggio di programmazione è un linguaggio che serve per comunicare con un calcolatore ma deve essere un linguaggio formale, rigoroso e bisogna in qualche modo definirne una grammatica. [...] Un programma è una sequenza di comandi. Ciascun comando sarà una sequenza di elementi. Le sequenze di comandi e di elementi possono organizzarsi secondo determinate regole ed ecco che in fondo in fondo lo strumento che utilizza Propp per descrivere le fiabe è molto vicino allo strumento che l'informatico usa per descrivere i programmi[4].

In effetti *Morfologia della fiaba* fa pensare a interessanti analogie tra la fiaba così come analizzata da Propp e il funzionamento di un programma a programmazione strutturata.

Innanzitutto, entrambi contengono elementi costanti ed elementi variabili. Anche un personaggio è definito da un insieme di variabili e di costanti che Propp chiama "attributi" (età, sesso, condizione, aspetto, ecc.): nel corso della favola può ad esempio cambiare il modo in cui il personaggio è vestito, mentre il suo nome rimane sempre lo stesso, per cui questo dato è una costante. Una costante fondamentale nelle fiabe è il ruolo del personaggio: l'eroe o l'antagonista non cambiano mai, così come il tipo di azioni che compiono (Propp 2000: 25-30). Ad esempio la funzione 25 dello schema di Propp prevede che all'eroe sia imposta una prova difficile che può concretizzarsi in azioni molto diverse – mangiare un'enorme quantità di tori o di carri di grano o lavarsi in un bagno di ghisa arroventata – ma l'azione compiuta dall'eroe resta sempre il superamento di una prova (Propp 2000: 65).

Propp classifica le funzioni secondo un sistema di simboli che genera un vero e proprio linguaggio in codice. Per esempio il simbolo che identifica l'azione della funzione 1, *Uno dei membri della famiglia si allontana da casa*, è *e*, e il suo contenuto è riassumibile con un unico sostantivo, *allontanamento*. È quindi possibile racchiudere un'azione all'interno di un'unica parola, così

[4] Trascrizione dell'intervento di Marco Maiocchi tratta dal filmato *Vladimir Propp: la grammatica della fiaba*, disponibile all'url http://www.raiscuola.rai.it/articoli/vladimir-propp-la-grammatica-della-fiaba/3889/default.aspx (ultima consultazione: 28.12.2019).

come per esempio nel linguaggio Java la parte fondamentale dell'istruzione è racchiusa in un nome o verbo come *setup* o *write*.

Inoltre, se la struttura di un sottoprogramma, cioè del blocco di istruzioni chiamato "procedura" che viene periodicamente richiamato dal programma per essere rieseguito, è composta di titolo e corpo (quest'ultima è la parte in cui si scrivono le istruzioni), allo stesso modo una funzione di Propp è composta da un nome e da un simbolo che la identifica, e il suo corpo non è altro che la descrizione nel dettaglio dell'evento-funzione.

C'è anche un'importante analogia sul piano lessicale, oltre che strutturale: con il termine *funzione* in programmazione si intende una procedura il cui nome che compare nel titolo corrisponde a un unico risultato, a cui si giunge mediante le istruzioni scritte nel corpo. Ciò che avviene nello schema di Propp è simile, in quanto ogni funzione è associabile a un simbolo e a una parola che indicano il risultato delle azioni pertinenti a quella data funzione. Ad esempio, all'interno della funzione 8 il rapimento è il risultato delle azioni malvagie del *villain* della fiaba.

PROGRAMMA	**FIABA**
dichiarazioni e procedura di setup	situazione iniziale *i*
funzioni e procedure	funzioni
Motivazioni	motivazioni

Tabella 1: Tre concetti fondamentali del linguaggio di programmazione imperativo a confronto con tre caratteristiche fisse delle fiabe secondo Propp 2000

Un altro parallelismo interessante è riscontrabile quando si legge che un elemento ausiliario delle favole di magia sono le triplicazioni. Possono ripetersi tre volte dettagli di carattere attributivo come le tre teste di drago, oppure possono essere assegnati a un personaggio tre compiti o questo deve servire tre anni, e l'azione può ripetersi meccanicamente (Propp 2000: 79). Ciò rispecchia il costrutto di iterazione adoperato in programmazione quando si renda necessario ripetere un'istruzione più volte: se si considera la fiaba come un algoritmo, potrebbe essere necessario ripetere certi eventi per arrivare al finale della narrazione.

Infine, Propp individua nelle motivazioni dei personaggi, intese come i moventi e i fini delle loro azioni, il terzo elemento ausiliario: le motivazioni sono presenti anche in programmazione quando il programmatore voglia chiarire le ragioni della scrittura di un determinato blocco di programma.

Considerando ora la struttura generale della fiaba di Propp e quella di un programma a paradigma imperativo, si può rilevare un'equivalenza fra sottoprogramma e "movimento". Con questo termine Propp indicava la parte fondamentale della fiaba, la quale si origina da un *danneggiamento* o da una *mancanza* ai danni del protagonista o delle persone a lui care e termina con la soluzione dell'intreccio, ad esempio un matrimonio. I movimenti possono dunque susseguirsi l'un l'altro oppure svolgersi in parallelo e ancora una volta troviamo una cosa simile nell'esecuzione di un programma: esso può contenere sottoprogrammi, e quindi istruzioni, da svolgersi in sequenza o contemporaneamente.

Nella prima favola presente nelle *Appendici* all'opera, Propp (2000: 137) mette in evidenza come gli eventi seguano passaggi rigorosi, le funzioni, espresse sotto forma di simboli. La sequenza degli eventi, come se si trattasse di comandi scritti in linguaggio di programmazione, genera un risultato, ovvero il lieto fine della storia:

> Un re ha tre figlie (i). Queste vanno a passeggio (e^3), si attardano in un giardino (q^1). Un drago le rapisce (X^1). Bando (Y^1). Ricerche di tre eroi (W↑). Tre combattimenti col drago (L^1-V^1), liberazione delle fanciulle (Rm4). Ritorno (↓), ricompensa (n0).
>
> e^3 q^1 X^1 Y^1 W↑ L^1-V^1 Rm4 ↓ n0

Nella situazione iniziale vengono presentati dei personaggi con certi attributi e si dichiarano delle variabili e delle costanti; seguono delle funzioni che rispettano il costrutto di sequenza, seguite da altre che prevedono dei costrutti di iterazione (la triplicazione tipica delle fiabe); infine, tre funzioni di chiusura concludono la storia.

La maggior parte delle fiabe, tuttavia, è a due o più movimenti: ciò si verifica quando sono presenti più danneggiamenti, ognuno dei quali si sviluppa singolarmente. La fiaba a due movimenti è il tipo fondamentale di tutte le fiabe e questi possono svolgersi in sequenza o in parallelo.

Vi è qui un'altra somiglianza con la programmazione. Come le funzioni di Propp si dispongono all'interno dei vari movimenti rispettando una gerarchia temporale, così all'interno di un programma anche i vari sottoprogrammi si suddividono in livelli gerarchici. Una fiaba diventa così paragonabile a un programma organizzato gerarchicamente. Come riportato nella Fig. 1 (ripresa da Chianese *et al*. 2017: 119), A è il programma principale, A.1 e A.2 i sottoprogrammi e A.1.1 e A.1.2 i sottoprogrammi che si appoggiano sui precedenti. Nel secondo schema della stessa figura i due movimenti A e B, che si compongono di un insieme di funzioni diverse, sono paragonabili a due sottoprogrammi e possono disporsi in sequenza o svolgersi in parallelo da un certo punto in poi.

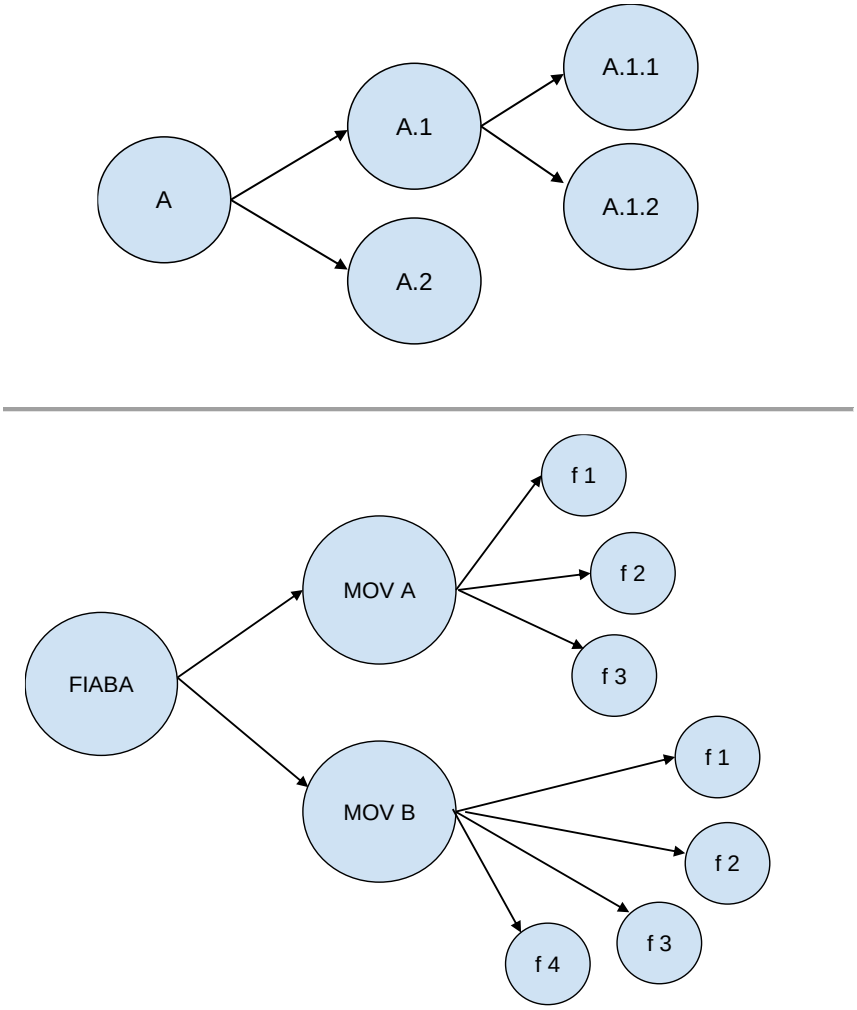

Figura 1: Struttura di un programma a programmazione strutturata (in alto) e di una fiaba organizzata per movimenti (in basso) a confronto

2.2. Metodologie didattiche

2.2.1. Letteratura potenziale: dalle regole alla creatività

Per meglio comprendere quale sia il valore aggiunto del mezzo informatico rispetto all'obiettivo di generare e analizzare una fiaba, si deve tener conto che Scratch è uno strumento che permette di creare un prodotto, un artefatto cognitivo che funga da mezzo di apprendimento. L'ambiente di programmazione a blocchi è dunque uno strumento per verificare una teoria, ossia che la scrittura di un testo narrativo non solo è soggetta a regole e vincoli che cambiano a seconda del linguaggio impiegato, naturale o artificiale, ma che è possibile costruire la narrazione disponendo gli eventi nel tempo in

modo preciso e secondo un processo algoritmico, similmente a quanto fa il programmatore quando utilizza i diagrammi di flusso nella progettazione del programma. Ogni lavoro che coinvolga fantasia e creatività è soggetto a regole, come ben compresero gli esponenti del gruppo letterario francese dell'*OuLiPo*, acronimo di Ouvroir de Littérature Potentielle[5], per il quale la presenza di vincoli e costrizioni è uno stimolo creativo che induce a uno sforzo di fantasia.

Si è dunque ritenuto interessante reimpiegare gli assunti teorici della letteratura potenziale e dello strutturalismo di Propp impiegando un mezzo tecnologico che coinvolgesse emotivamente gli studenti, che rafforzasse memoria e capacità di comprensione, che fosse sempre a disposizione e gratuito. Oltre a ciò, Scratch possiede indubitabili vantaggi: era conosciuto da alcuni studenti coinvolti nella sperimentazione, è di facile utilizzo, è molto vicino al linguaggio naturale, ha una grafica accattivante, è preciso (il colore e la forma dei blocchi rende evidente l'appartenenza a diverse categorie computazionali) e permette di realizzare storie e narrazioni. L'attività, intitolata *Morfologia della fiaba con Scratch*, ha mirato a far emergere competenze nello studente attraverso un processo fatto di sperimentazione, ragionamento logico, spirito critico, creatività, collaborazione e condivisione. Seguendo la strada indicata da Papert (1993) non è stato il computer a insegnare agli studenti, ma sono stati loro a prenderne il controllo, assumendo il ruolo di programmatori, mentre il docente e il tutor hanno avuto il ruolo di guide, di facilitatori[6].

Inaugurando un nuovo punto di vista sul mezzo informatico e suggerendo parallelismi con la letteratura, si è cercato di ragionare sull'insolita analogia tra fiaba e programma, combinando conoscenze linguistiche e informatiche e sottolineando che ciò che si stava facendo aveva come traguardo il miglioramento della propria capacità di esprimersi.

Punto di partenza è stato il ricreare con il *coding* varie azioni di un'ipotetica fiaba. I ragazzi si sono posti domande di questo genere: che blocchi devo usare per far svolgere al mio personaggio l'azione che ho in mente? Come traduco in italiano ciò che ho programmato? La presenza di tali vincoli, che portavano inevitabilmente lo studente a cercare sempre un modo nuovo di risolvere i problemi che mano a mano incontrava, lo inducevano anche esprimere la sua creatività.

[5] Per un inquadramento dei principali aspetti e alcuni rinvii bibliografici, si veda l'intervento di Paolo Albani, *La letteratura potenziale. Alcune note sparse*, http://www.paoloalbani.it/-letteraturapotenziale.html (ultima consultazione 31.12.2019).

[6] Nel caso specifico si tratta rispettivamente del docente di italiano della classe, prof. Gabriele Benassi, e dell'autore di questo contributo.

2.2.2. *Gamification*

Un'altra caratteristica di Scratch è quella di apportare meccaniche di *gamification* all'ambiente di apprendimento, in base alle quali l'utente *gioca* con il contenuto al fine sia di apprendere i concetti di base della programmazione sia di esprimersi in maniera creativa. Scratch possiede gli elementi fondamentali della *gamification*: fornisce *feedback* sia in tempo reale (se i blocchi non sono disposti correttamente il programma non funziona) sia all'interno della *community*, i cui utenti possono commentare e *remixare*, cioè modificare il codice di ogni progetto condiviso. Dal momento che ogni progetto racconta una piccola storia, diventa possibile fare *storytelling* sfruttando gli *sprite*, i personaggi / oggetti messi a disposizione dalla piattaforma, e adoperando gli sfondi, gli ambienti in cui i personaggi si muovono.

Inoltre un tipico elemento ludico insito in Scratch è la sfida: un laboratorio di *coding* di qualsiasi tipo si compone di continue missioni aventi nel loro complesso obiettivi di apprendimento precisi. Infine, un gioco concede l'importante libertà di fallire, dal momento che spesso si impara di più dai propri errori senza la pressione di un giudizio o di un voto, così che le conseguenze dell'errore sono ben visibili, ma l'assenza di una punizione incoraggia la volontà del riprovare per tentativi.

In riferimento alla sperimentazione in classe, ci si è basati su quattro concetti fondamentali della *gamification*:

- STORYTELLING, per cui ogni funzione raccontava una piccola storia. L'insieme delle funzioni avrebbe generato una fiaba collettiva.
- *FEEDBACK* dei compagni di classe in tempo reale e della *community* online.
- SFIDA, dal momento che ogni attività si è incentrata su una o più challenge (*vi darò una traccia: programmate coerentemente usando la fantasia; traducete ciò che avete programmato in italiano*…).
- LIBERTÀ DI FALLIRE, per cui non importava che i progetti *funzionassero* al momento della condivisione con la classe: l'importante era aver sperimentato per poi cercare insieme una soluzione per sistemare al meglio i blocchi.

2.2.3. *Project Based Learning*

L'organizzazione del lavoro si è ispirata all'approccio *Project Based Learning*, seppur limitatamente alla fase iniziale della metodologia, in quanto

l'esiguo tempo a disposizione non avrebbe permesso la realizzazione di un progetto didattico esaustivo[7].

L'approccio per progetti si adatta bene a un ambiente di apprendimento costruttivista: stimola gli studenti allo sviluppo di problemi autentici, dal momento che le difficoltà che sorgono nel corso del lavoro rispecchiano quelle che sorgono in situazioni reali, al di fuori dell'ambiente scolastico. L'obiettivo finale è la realizzazione di un prodotto, sia materiale che digitale, che un possibile utente potrebbe adoperare.

In accordo con il metodo *Project Based Learning*, il prodotto finale da realizzare è stato uguale per tutti, la fiaba, dapprima *programmata* su Scratch e poi resa in forma scritta. Si sono divisi i gruppi di lavoro per sottotemi, cioè ogni gruppo ha programmato la funzione assegnatagli e ha realizzato lo stesso tipo di prodotto degli altri gruppi, un progetto Scratch. Ogni studente, al termine delle attività, ha osservato il lavoro degli altri grazie all'esposizione del lavoro di ogni gruppo, proiettato sulla LIM. Si è creato un ambiente collaborativo in cui l'emergere continuo di problemi è stato analizzato sia all'interno del singolo gruppo, sia all'interno dell'intera classe.

2.3. Le fasi dell'attività

2.3.1. Finalità e obiettivi

Il lavoro di sperimentazione ha avuto luogo il 18 e il 19 gennaio 2019, in una classe seconda della Scuola Media "C. Pepoli" di Bologna. Al termine si è ottenuta la bozza di una fiaba creata in maniera collettiva a partire dalla suddivisione in funzioni secondo lo schema di Propp (2000). Ogni funzione e ogni evento della fiaba sono stati riprodotti sotto forma di linguaggio di programmazione a blocchi su Scratch, per poi essere reso in forma narrativa in lingua italiana[8].

Si è cercato di suggerire agli studenti un parallelo tra fiaba e programma informatico al fine di farli pensare a un testo come a un qualcosa di scomponibile in parti che si susseguono in maniera logica e dotate di una sintassi. Gli obiettivi della sperimentazione stati tre:

- Rendere maggiormente consapevoli gli studenti dell'utilità del procedimento algoritmico al fine della realizzazione di un testo.

[7] Per maggiori informazioni sul Project Based Learning (PBL) si veda quanto scrive Enzo Zecchi al link https://enzozecchi.com/article/attivita-diproject-based-learning-pbl-attraverso-il-metodolepida-scuola/ (ultima consultazione: 28.12.2019).

[8] Si esprime viva gratitudine al docente di italiano delle classi, prof. Gabriele Benassi, per aver reso possibile lo svolgimento dell'attività.

- Far comprendere l'importanza del rispetto delle regole di sintassi che consentono a un programma di funzionare e quindi a un testo di essere scritto e interpretato in maniera corretta.
- Far riflettere gli studenti sulle diverse potenzialità espressive dei due tipi di linguaggio a partire dallo scheletro narrativo fornito dal linguaggio di programmazione a blocchi.

Si è quindi cercato di fare raggiungere agli studenti le seguenti competenze:

- Applicazione del pensiero computazionale e creativo alla materia di Italiano.
- Aumento della capacità di comprensione del testo in seguito all'acquisita consapevolezza della struttura algoritmica di un testo narrativo semplice e miglioramento della propria capacità di esprimersi attraverso il mezzo scritto.
- Espressione della propria creatività e della capacità di lavorare in gruppo; applicazione e rinforzo di competenze linguistiche e informatiche.

La sfida proposta è stata quella di scrivere un testo narrativo dapprima in un linguaggio di programmazione e successivamente in italiano, passando da una struttura testuale regolativa a una narrativa.

2.3.2. La fase preparatoria

Il docente della classe ha creato un account docente su Scratch il giorno precedente l'attività: ciò ha permesso, similmente a quanto accade con Google Classroom, di creare delle classi virtuali composte da account studenti. L'insegnante, dopo aver creato la classe, può aggiungerne il nome e una breve descrizione, aggiungere gli account degli studenti e monitorare direttamente i progetti attinenti a uno specifico lavoro svolto in classe.

La fase di preparazione all'attività ha invece previsto lo studio delle 31 funzioni di Propp al fine di individuare quelle più adatte a essere programmate. In base a un criterio di facilità di riproduzione su Scratch e di rilevanza narrativa sono state selezionate 11 funzioni e si è deciso di creare 3 coppie unendo 6 funzioni fra di loro, in quanto legate da una relazione di causa-effetto (quelle che Propp chiamava *funzioni appaiate*) mentre una è stata scartata. Ne sono risultate dunque 7 istruzioni diverse e in base a ciò la classe, composta da 24 studenti, è stata divisa in gruppi da 3-4 persone, in modo da massimizzare la collaborazione e l'aiuto reciproco.

La conoscenza di Scratch all'interno della classe era variabile: alcuni studenti non ne avevano nessuna esperienza, altri hanno affermato di farne uso anche a casa in orario extrascolastico, mentre la maggior parte aveva una pic-

cola esperienza dell'ambiente che ha permesso durante l'attività di padroneggiare i comandi di base.

Le funzioni scelte sono state stampate su cartoncino azzurro, analogamente a quanto avviene con le *Scratch cards*, istruzioni in forma di carta da gioco aventi lo scopo di riprodurre comandi semplici e di fornire le basi a chi si avvicina per la prima volta a questo linguaggio[9].

Nelle prime righe della carta è stata riportata la descrizione della funzione in maniera molto simile a quanto scritto da Propp; sono poi state adattate le sottofunzioni originali per renderle in forma di *quest*, in modo da motivare e coinvolgere i ragazzi, il cui compito è stato quello di programmare la funzione su Scratch scegliendo una tra le varie opzioni. Ad esempio, le funzioni 25 e 26 sono state rese in questa forma:

25. L'eroe deve superare una prova difficile.
26. La prova è superata.
    A. L'eroe deve mangiare un enorme quantità di cibo o bevande. Create un gioco!
    B. Gli è proposto un indovinello da risolvere.
    C. Deve saltare degli ostacoli.
    D. Deve radunare un esercito o un enorme numero di cavalli. Clonate gli sprite!
    E. Deve far volare una nave. Cercate "sailboat" tra gli sprite!
    F. L'eroe vince una gara di ballo.

Le carte pensate per questa sperimentazione assegnano un'istruzione da ricreare in forma libera, mentre all'interno delle *Scratch cards* i blocchi dell'istruzione corrispondente sono riportati interamente: l'utente non deve fare altro che disporre i blocchi così come riportato sulla carta.

[9] Le Scratch cards sono scaricabili sul sito di Scratch e i loro contenuti sono proposti anche in forma di tutorial: https://scratch.mit.edu/ideas (ultima consultazione: 28.12.2019).

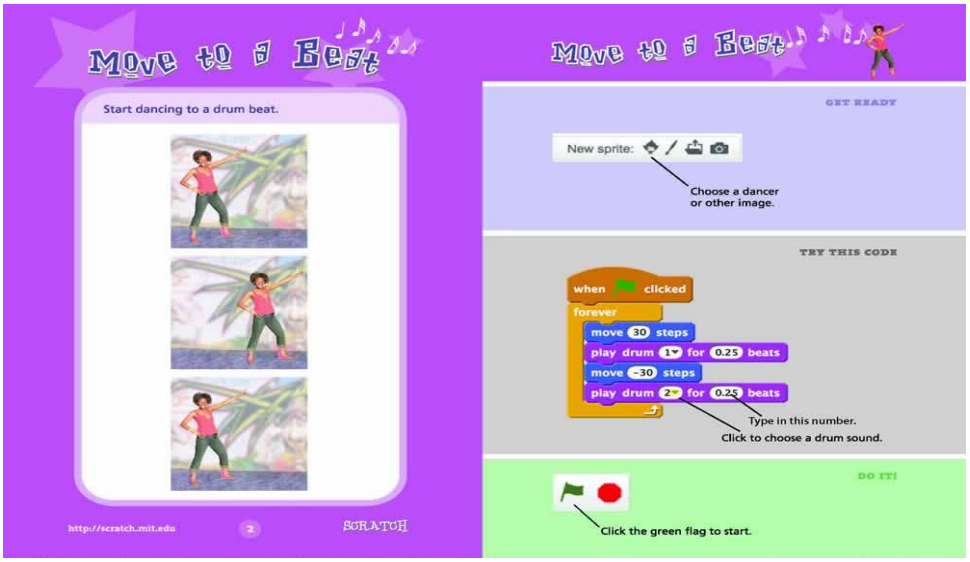

Figura 2: Una delle *Scratch cards* che insegna a far ballare i personaggi e ad aggiungere suoni

Successivamente sono state testate le funzioni su Scratch per controllare che fossero effettivamente riproducibili e che il livello di difficoltà non fosse troppo alto. Il risultato è stato soddisfacente, in quanto ogni funzione o coppia di funzioni si prestava a essere riprodotta attraverso precise categorie di comandi, come riportato nella Tab. 2. Inoltre, le funzioni della fiaba potrebbero essere impiegate come tutorial introduttivi a Scratch, dal momento che queste permettono di riprodurre conversazioni fra personaggi, giochi, movimenti, effetti grafici, clonazioni, suoni con pochi comandi.

Funzioni 4 e 5	conversazione fra sprite: uso dei blocchi *messaggio*
Funzioni 6 e 7	conversazione, uso dei costumi e degli sfondi
Funzione 8	uso dei comandi *mostra* e *nascondi*, uso dei costumi
Funzione 14	uso dei costumi e degli effetti grafici
Funzione 15	impiego dei comandi di base (categoria movimento), uso degli sfondi
Funzioni 21 e 22	uso dei costumi, delle variabili e degli operatori logici

Funzioni 25 e 26	ogni sottofunzione è adatta a una certa tipologia di comandi: a) clonazione, variabili, operatori logici; b) messaggi e operatori logici; c) movimento e tempo; d) clonazione; e) conversazione e movimento; f) movimento, tempo, suoni

Tabella 2: Funzioni di Propp e rispettive categorie di comandi di Scratch

2.3.3 Presentazione dell'attività in classe

Il lavoro della prima giornata si è svolto in due momenti distinti. La prima ora è stata dedicata alla spiegazione dell'ipotesi di partenza, dunque al parallelismo tra fiaba e programma e alla spiegazione del successivo lavoro da svolgersi in aula atelier.

Dopo l'illustrazione degli elementi chiave della fiaba e come si distingue dalla favola, è stato esposto succintamente il lavoro di Propp, suggerendo un confronto tra fiaba e programma per singoli passi:

1. Sia una fiaba che un programma sono composti da parti che si susseguono secondo un ordine logico.
2. Ne consegue che sia una fiaba che un programma hanno la necessità di risolvere l'algoritmo che porta in un caso al finale della storia, nel secondo al termine dell'esecuzione delle istruzioni.
3. Ciò che rende possibile la sequenza logica delle azioni o delle istruzioni è il rispetto della grammatica di quel particolare linguaggio, il linguaggio naturale nel caso della fiaba e quello di programmazione nel caso del programma.
4. La riflessione è stata estesa non solo ai testi regolativi ma anche ai testi narrativi, come la fiaba, e ai testi argomentativi. Ad esempio, in quest'ultimo caso è necessario suddividere l'argomentazione in parti: enunciazione del problema, esposizione della propria opinione, confutazione della tesi contraria e conclusioni. Per facilitare la comprensione del concetto si è suggerito un parallelo con la *scaletta* che viene insegnata a scuola per facilitare la scrittura del tema di Italiano, sottolineando come lo strutturare le proprie idee in sequenza rispetti la struttura del nostro cervello, che necessita di scomporre i problemi e di concentrarsi sulle singole parti di esso.

Nella parte restante della prima ora è stato lasciato un po' di tempo ai ragazzi per formare i gruppi e successivamente sono stati creati gli account per ogni studente e per ogni gruppo. I gruppi formati sono stati sette, come le funzioni di Propp da programmare su Scratch (Tab. 3).

2.3.3. La programmazione su Scratch

Al *coding* su Scratch sono state dedicate due ore[10]. Le sedie e i computer in aula sono stati predisposti in modo che i ragazzi al loro arrivo fossero già in condizione di poter lavorare. Prima di distribuire le funzioni a ogni gruppo gli studenti sono stati informati che avrebbero avuto la possibilità di impiegare tutta la loro fantasia, con l'unico vincolo che l'eroe e l'antagonista dovessero essere uguali per tutti ai fini della coerenza della fiaba. Si è data la possibilità di scegliere l'eroe fra i due sprite *knight* e *prince* (il primo rappresentava la versione in armatura del secondo), mentre l'antagonista era rappresentato dallo sprite *wizard*.

Sono poi state distribuite le carte su cui erano riportate le funzioni, tenendo conto del livello di dimestichezza con Scratch dei componenti i vari gruppi. Ogni gruppo ha poi scelto l'opzione che preferiva all'interno della funzione da realizzare (Tab. 3) e si è iniziato a programmare e a esplorare le funzionalità di Scratch. Docente e tutor hanno fornito, quando opportuno, suggerimenti e aiuti nella costruzione del codice; in questo modo la classe è riuscita a completare positivamente la consegna nel tempo a disposizione.

Gruppo A	f. 4-5	ricognizione dell'antagonista e suo ottenimento di informazioni sull'eroe
Gruppo B	f. 6-7	l'antagonista inganna con successo l'eroe
Gruppo C	f. 8	danneggiamento: l'antagonista danneggia una persona cara all'eroe
Gruppo D	f. 14	fornitura di un oggetto magico all'eroe da parte di un aiutante
Gruppo E	f. 15	trasferimento, l'eroe insegue l'antagonista
Gruppo F	f. 21-22	persecuzione dell'antagonista ai danni dell'eroe e successivo suo salvataggio
Gruppo G	f. 25-26	l'eroe supera con successo una prova

Tabella 3: Le funzioni programmate da ciascun gruppo rispecchiano lo sviluppo cronologico della fiaba

2.3.4. La condivisione dei progetti

Nel corso di un'ora tutti i progetti Scratch sono stati presentati e condivisi. Prima delle loro presentazioni è stata ricordata alla classe la funzione di Propp che era stata rappresentata, cercando di illustrarne il contenuto e di spiegarne il ruolo all'interno di un'ipotetica fiaba. In seguito, un rappresentante di ciascun gruppo avviava il programma cliccando sulla bandierina ver-

[10] La galleria con tutti i progetti è disponibile al link https://scratch.mit.edu/studios/5863756/ (ultima consultazione: 28.12.2019).

de e/o manovrando i personaggi con la tastiera; successivamente era mostrato alla classe il codice, in modo da spiegare come le azioni dei personaggi erano state ricreate. Alcuni gruppi, coerentemente con le istruzioni, avevano ricreato l'azione fiabesca tramite le conversazioni con i personaggi, mentre altri sotto forma di gioco.

2.3.5. La traduzione dal codice a blocchi all'italiano

L'ultima ora è stata dedicata alla traduzione della funzione dal linguaggio a blocchi all'italiano. Ogni gruppo è entrato su Google Classroom, dove era disponibile un file doc condiviso con i capigruppo, cioè quegli studenti ai quali era stato aperto l'account Scratch studente e che in questa fase hanno avuto il compito di guidare i compagni nella stesura del testo. Ciò ha permesso agli studenti di scrivere in tempo reale e su un unico file il testo relativo alla funzione della fiaba precedentemente programmata. Il compito degli studenti è stato quello di osservare il codice su Scratch in modo da scrivere, per quanto possibile, una sorta di traduzione in italiano a partire da un tipo di linguaggio regolativo, come quello del codice, per arrivare a esprimersi con un registro descrittivo nella loro lingua naturale. All'eroe è stato dato il nome *Pepol* mentre al mago antagonista quello di *Carlo*, in onore di Carlo Pepoli, al quale la scuola in cui si è svolta la sperimentazione è intitolata.

Negli ultimi minuti a disposizione si è controllato se i gruppi avessero rispettato il vincolo del tempo verbale (si è optato per il presente), si è corretta la sintassi, si è guardato al senso generale delle frasi e se rispettasse l'azione programmata o riportasse contenuti non pertinenti. Grazie alle indicazioni in corso d'opera, per evitare che i ragazzi tendessero a inventare un'introduzione non necessaria alla loro funzione, quasi tutti i gruppi si sono limitati a descrivere l'azione programmata aggiungendo qualche minimo elemento narrativo.

Funzioni 4 e 5:

Carlo deve trovare informazioni sul principe Pepol.
Arriva in una foresta magica e incontra un elfo, il mago le[1] chiede se ha visto il principe e lui risponde che stà[2] andando verso il castello. Il mago inizia subito il viaggio

---

è una femmina?
senza accento

Funzioni 6 e 7:

Carlo vede Pepol passare per il sentiero, così decide di trasformarsi in un'elfa e di ingannarlo. Appare davanti a Pepol e gli offre un biscotto della fortuna, però avvelenato. Il principe si fida di lei, decide di accettarlo, quindi lo mangia e a un certo punto sviene.

Funzione 8:

Mentre la principessa sta osservando le onde del mare che si frastagliano sulla spiaggia, appare alle spalle della bella Anastasia una ranocchia dall'aspetto dolce. Le salta addosso e all'improvviso si trasforma in un malefico mago che punta il dito contro Anastasia trasformandola in un aggressivo orso bruno.
Suo marito il principe vede tutto da lontano.

Funzione 14:

Pepol sta viaggiando, per tornare al suo castello. In quel momento si trova all'interno di una giungla buia. All'improvviso intravede un mago, il cui nome è Carlo. Egli vuole far bere all'eroe una pozione che è capace di trasformarlo in un orco. Riesce a fargliela bere con l'inganno. Così Pepol si trasforma in un orco arrabbiato e triste a causa del terribile mago Carlo.

Funzione 15:

Il grande cavaliere ed eroe Pepol deve attraversare un enorme, immenso e centenario bosco. Deve attraversare una tempesta incredibile, scagliata da Carlo, il mago invincibile. Salvare la principessa e riportare la pace nel regno è la sua ambizione più grande. In pochi secondi un miliardo di fulmini cadono dal cielo, come se scivolassero sul ghiaccio. Casualmente.
Il nostro eroe viene colpito e trasformato in un pollo, incapace di muoversi: ma con la sua forza si ritrasforma e come una lepre schiva tutti i fulmini con la velocità di un computer che esegue i calcoli.
Arriva infine da Carlo.

Funzioni 21 e 22:

Pepol, sulla Luna incontra il malvagio Carlo che è in grado di trasformarsi in "Drago dell'Ender".
Però codesto drago ha un punto debole che si chiama "Banana multicolor", e il suo vero nome è "Thanos".
Un giorno Pepol fa 15 passi in avanti e inizia a combattere contro esso,ma per fortuna possiede Thanos e così lo batte e glielo rinfaccia.

Funzioni 25 e 26: Il nostro eroe Pepol, percorre il lungo corridoio del castello sale le scale e si trova fuori dal castello. Lì c'è una prova che se l'avesse superata sarebbe riuscito a scappare. In questa prova deve riuscire a mangiare il più possibile fra torte, tacos e ciambelle. Mentre Pepol mangia, dal nulla arrivano palloncini per aver superato la prova e subito dopo il grande ponte levatoio si apre e Pepol scappa.

Tabella 4: La traduzione in italiano del codice a blocchi

Mettendo in ordine cronologico le funzioni si nota che la fiaba creata con Scratch segue uno sviluppo coerente: si parte da una situazione in cui l'antagonista cerca informazioni sull'eroe e seguono tre funzioni di danneggiamento di cui la numero 8 è quella più importante[11]. Inizia da questo momento il viaggio dell'eroe alla ricerca della principessa: viene ostacolato dal mago dai cui pericoli però si salva (anche se i ragazzi hanno scritto che l'eroe sconfigge il mago, in realtà la funzione rappresenta il salvataggio del protagonista da un'insidia) e per salvare la principessa si sottopone a una prova difficile, in questo caso una prova del cibo. Potremmo rappresentare le funzioni elencate con i simboli di Propp:

v^1 w^1 j^2 y^2 X12 Z^1 R^2 P^1 S^2 C *A

Si tratterebbe di una fiaba a un solo movimento, ma di lunga durata, priva di una situazione iniziale *i* e di una o più funzioni conclusive, le quali sono state successivamente integrate da ogni studente seguendo la propria fantasia.

Vi sono ovviamente buchi narrativi tra una funzione e l'altra, ma la cooperazione tra scrivente e destinatario, il lettore umano, permette a quest'ultimo di comprendere almeno il filo conduttore della storia (la lotta tra l'eroe e l'antagonista in diversi tempi e luoghi in seguito a un danneggiamento).

L'ultima fase del lavoro ha avuto come obiettivo un continuo raffinamento dal linguaggio in forma di codice a blocchi all'italiano, parallelamente a quanto avviene in programmazione, dove si chiama "raffinamento" il processo di traduzione da parte del programmatore di un'istruzione in linguaggio

[11] In realtà le funzioni di danneggiamento dovrebbero essere due, dal momento che la funzione 14, scritta dal gruppo D, dovrebbe riguardare il fatto che l'eroe viene aiutato da un donatore. La rappresentazione della funzione è risultata erronea a causa del fraintendimento di un'indicazione generale data all'inizio dell'attività, in base alla quale tutti i gruppi dovevano includere nel progetto i due *sprite* rappresentanti l'eroe e l'antagonista.

naturale in un'istruzione in linguaggio di programmazione. In questo caso, invece, è avvenuto esattamente il contrario: si è partiti dal linguaggio di programmazione per arrivare a esprimersi in lingua italiana. Il testo in linguaggio naturale è infatti un codice per esseri umani e «può essere visto come una sequenza d'istruzioni per l'interprete. Le sequenze testuali pilotano, guidano [...] la costruzione della coerenza testuale», secondo le parole di Angela Ferrari citate da Lo Duca 2014: 209.

2.3.6. Codice e narrazione a confronto

Il passaggio da un linguaggio estremamente povero e tecnico come quello regolativo proprio del codice a blocchi a un linguaggio sintatticamente elaborato e descrittivo come l'italiano ha portato gli studenti a trascurare alcuni micro-passaggi narrativi pur presenti nel codice. Si consideri questo esempio di traduzione della funzione 8 (Tab. 4) e si confronti la frase evidenziata in corsivo con il rispettivo blocco di codice presentato nella Fig. 3:

> Mentre la principessa sta osservando le onde del mare che si frastagliano sulla spiaggia, appare alle spalle della bella Anastasia una ranocchia dall'aspetto dolce. *Le salta addosso e all'improvviso si trasforma in un malefico mago che punta il dito contro Anastasia trasformandola in un aggressivo orso bruno.* Suo marito il principe vede tutto da lontano.

Il codice mostra un alternarsi serrato di *costumi* da parte del mago antagonista, che diventa sempre più grande man mano che cambia *costume*. Tuttavia l'animazione non mostra il mago nell'atto di *saltare addosso* alla principessa e nel testo non si fa menzione del fatto che il mago diventa sempre più grande. Le intenzioni erano quelle di ricreare terrore facendo trasformare un'innocua ranocchia in un mago sempre più grande e minaccioso, mostrando così il potere dell'antagonista, ma ciò non traspare nel testo in italiano.

Un altro esempio di mancata corrispondenza tra codice e decodifica in italiano è visibile confrontando il testo relativo alle funzioni accoppiate 21 e 22 con il rispettivo codice (Tab. 4):

> Pepol, sulla Luna incontra il malvagio Carlo che è in grado di trasformarsi in "Drago dell'Ender". *Però codesto drago ha un punto debole che si chiama "Banana multicolor", e il suo vero nome è "Thanos". Un giorno Pepol fa 15 passi in avanti e inizia a combattere contro esso, ma per fortuna possiede Thanos e così lo batte e glielo rinfaccia.*

Anche in questo caso, si confronti la parte evidenziata in corsivo con la Fig. 4. La prima frase in corsivo espone in modo molto vago il fatto che l'eroe abbia usato delle banane come ostacoli da frapporre tra lui e l'antagonista lanciato al suo inseguimento. Il codice ci mostra invece lo svolgersi dell'azione maggiormente nel dettaglio seppur molto sinteticamente: i blocchi ripor-

tati sopra ci dicono che nel momento in cui i due personaggi *bananas e wizard* si toccano entrambi scompaiono. Non c'è inoltre traccia nel progetto Scratch del fatto che l'eroe combatta contro il drago e che lo sconfigga, azione comunque non pertinente con le funzioni in questione. Che cosa è accaduto alle banane e al drago quando si sono toccati? Il testo in italiano non lo dice.

Questi due esempi mostrano come il codice sia un linguaggio pienamente in grado di esprimere contenuti narrativi semplici e che una traduzione in lingua naturale è possibile, anzi è lecito scendere in dettagli di cui a prima vista gli studenti non si sono accorti o che hanno sottovalutato.

Nonostante le differenze tra i due tipi di linguaggi, si è potuto notare che:

- La scomposizione di una narrazione in funzioni, e quindi in fasi, è di aiuto nel comprendere le caratteristiche di un genere narrativo.
- Il fatto di costruire al computer con le proprie mani la storia e, successivamente, di scriverla, aumenta il coinvolgimento degli studenti e quindi la memorizzazione e l'acquisizione di capacità sia informatiche che letterarie.
- Il linguaggio regolativo del codice, che prevede una serie di istruzioni al fine di realizzare una certa azione, può essere convertito in uno stile narrativo-descrittivo in italiano.
- Il codice di Scratch, se analizzato nel dettaglio, si rivela pienamente efficace nel trasporre in italiano le azioni e i pensieri dei personaggi.

2.3.7. La prosecuzione del lavoro in classe

Il lavoro iniziato con l'aiuto di Scratch è poi proseguito, fino al traguardo della scrittura integrale della fiaba, assegnata a ogni studente come compito individuale da svolgere a casa.

La resa in italiano delle funzioni ha attraversato un'evoluzione partita dai blocchi di codice: la classe ha cioè creato collettivamente le prime due macro-parti della fiaba (prologo e danneggiamenti da parte dell'antagonista), che sono state integrate successivamente secondo la fantasia del singolo studente (vittoria dell'eroe ed epilogo). Il docente della classe, proseguendo nel ruolo di facilitatore, ha suggerito agli studenti degli espedienti per unire il tutto secondo logica e in modo narrativamente efficace al fine di costruire uno *storytelling*. Si è così creata una narrazione vera e propria *to be continued*.

Lo sviluppo seguito alla programmazione su Scratch è quindi composto di una parte iniziale, realizzata completamente in classe, come la seguente:

> Tanto tempo fa, quando esistevano ancora i castelli, Pepol era un cavaliere forte, con gli occhi azzurri, proprio un bel giovane. Era amato da tutte le ragazze ma lui era più interessato ai cavalli che alle belle fanciulle. Aveva letto da qualche parte dell'esistenza di un cavallo meraviglioso che quasi nessuno aveva mai visto. Narravano che fosse un cavallo dorato con una bardatura in dia-

> manti sopraffini e rubini splendenti. Narravano anche che questo cavallo si affezionava a un solo padrone al quale donava sette gemme da utilizzare una sola volta nella vita. Ognuna di queste aveva poteri differenti e sovrumani: la gemma viola dava il potere, quella verde il tempo, l'arancione gli universi paralleli, la gialla la possibilità di leggere nel pensiero delle persone, la rosa l'invisibilità, la blu il teletrasporto e la rossa donava la trasfigurazione. Anche il mago Carlo, venuto a sapere del cavallo, decise di mettersi in cammino per trovarlo. Imparò ben presto che anche Pepol si era messo alla ricerca del cavallo.

In seguito, come esempio del lavoro da svolgersi a casa da parte del singolo studente, sono state rese in una forma narrativa più ricca e leggibile le coppie di funzioni 4-5 e 6-7 (Tab. 4):

> Per trovarlo giunse in una foresta magica nella quale incontrò un elfo a cui chiese se avesse visto il principe. L'elfo rispose: "Si l'ho visto, sta andando in direzione del castello." Ricevute queste informazioni il mago Carlo si affrettò verso il castello. Strada facendo si accorse che Pepol si era fermato per riposare e si era addormentato all'ombra di un albero. Iniziò dunque a chiamarlo e Pepol si svegliò accorgendosi che qualcuno lo stava chiamando. Per non essere scoperto e per ingannare il giovane principe, il mago si trasformò in un elfo. Apparve davanti a lui e gli disse: "Vedi questo? Ti darà la forza per proseguire il cammino". Così, ancora mezzo addormentato, Pepol volse lo sguardo verso il biscotto della fortuna che il finto elfo gli stava dando e, spinto da una certa fame mattiniera lo mangiò senza indugiare e si rimise in cammino. Dopo pochi passi sentì girare vorticosamente la testa e iniziò a sentire un tremendo mal di pancia e svenne. Il biscotto della fortuna si tramutò in sfortuna.
>
> Intanto in un regno vicino la bella principessa Anastasia si godeva la brezza del mare. Le onde, calme e regolari accarezzavano la sabbia. Si respirava un'aria pulita e profumata di salsedine. Lontano da lei il principe consorte… CONTINUA...

A questo punto ogni studente ha poi proseguito in autonomia la storia, ampliando la precedente traduzione delle funzioni e inventando un finale personalizzato.

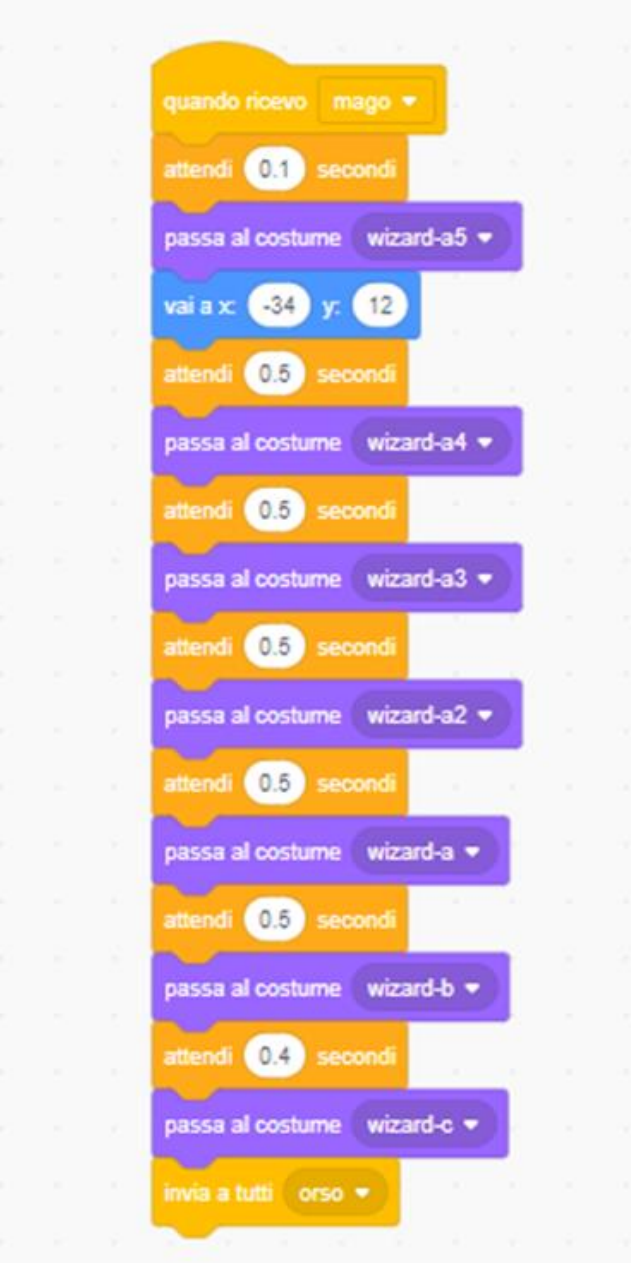

Figura 3: Blocco di codice relativo alla funzione 8, *danneggiamento*. Il serrato alternarsi dei costumi indica che il mago sta diventando sempre più grande

Figura 4: Blocchi di codice relativi alle funzioni 21 e 22, *persecuzione del protagonista da parte dell'antagonista e suo mettersi in salvo*

3. Conclusioni

Questo contributo ha esaminato i punti di contatto tra linguaggio di programmazione e linguaggio naturale con lo scopo di suggerire modalità di impiego del *coding* come strumento attraverso il quale comprendere e memorizzare meglio sia le caratteristiche di un genere narrativo specifico come la fiaba che le fasi di costruzione di una narrazione, valide come punto di partenza per lo studio di generi letterari più complessi.

Da un lato, ciò dimostra che il linguaggio naturale può essere convertito da un registro linguistico regolativo tipico del codice a uno descrittivo e narrativo, in quanto il codice a blocchi di Scratch è facilmente comprensibile e riscrivibile nella propria lingua; dall'altro, ciò permette di mettere a frutto le potenzialità del *coding*, in quanto un linguaggio di programmazione facilmente comprensibile, univoco nel significato e strutturato attraverso passaggi sequenziali consente allo studente di mettere a fuoco la struttura di un testo narrativo semplice in lingua naturale come la fiaba. Inoltre la correttezza del-

la sintassi del codice a blocchi, necessaria affinché l'animazione si svolga nel modo corretto, sottolinea per analogia l'importanza della correttezza grammaticale quando si voglia produrre un testo in autonomia.

Il codice a blocchi ha fornito una traccia composta da azioni elementari dotate di un significato preciso e pertinente rispetto al genere della fiaba, sul quale si è continuato a lavorare fino al raggiungimento di una forma narrativa completa in italiano.

Riferimenti bibliografici

MIUR (2018) = *Indicazioni nazionali e nuovi scenari. Documento a cura del Comitato Scientifico Nazionale per le Indicazioni Nazionali per il curricolo della scuola dell'infanzia e del primo ciclo di istruzione*, https://-www.miur.gov.it/documents/20182/0/Indicazioni+nazionali+e+nuovi+scenari/ (ultima consultazione: 31.12.2019).

Cacciari, Cristina (2011), *Psicologia del linguaggio*, Bologna, il Mulino.

Chianese, Angelo – Moscato, Vincenzo – Picariello, Antonio – Sansone, Carlo (2017), *Le radici dell'informatica. Dai bit alla programmazione strutturata*, Santarcangelo di Romagna, Maggioli.

Good, Judith – Howland, Kate (2017), *Programming language, natural language? Supporting the diverse computational activities of novice programmers*, in «Journal of Visual Languages and Computing», 39, pp. 78-92.

Lo Duca, Maria G. (2014), *Lingua italiana ed educazione linguistica. Tra storia, ricerca e didattica*, Roma, Carocci.

Oliverio, Alberto (2017), *Il cervello che impara. Neuropedagogia dall'infanzia alla vecchiaia*, Firenze, Giunti.

Papert, Seymour (1993), *Mindstorms. Children, Computers, And Powerful Ideas*, New York, Basic Books.

Penge, Stefano (2018), *Lingua, coding e creatività. Fare coding con le materie umanistiche*, Roma, Anicia.

Propp, Vladimir Jakovlevič (2000), *Morfologia della fiaba*, Torino, Einaudi.

Ragonis, Noa – Shilo, Gila (2014), *Drawing Analogies between Logic Programming and Natural Language Argumentation Texts to Scaffold Learners' Understanding*, in «Journal of Information Technology Education: Research», 3, pp. 73-89 (http://www.jite.org/documents/Vol13/JITEv13-ResearchP073-089Ragonis0415.pdf, ultima consultazione: 31.12.2019).

Resnick, Mitchel (2018), *Come i bambini: immagina, crea, gioca e condividi. Coltivare la creatività con il* Lifelong Kindergarten *del MIT*, Trento, Erickson.