UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di
Studi Umanistici
e della Formazione

Corso di Laurea in Scienze della
Formazione Primaria

Il *coding* a scuola: un progetto interdisciplinare per una classe seconda primaria

Relatore

Andreas Robert Formiconi

Candidato

Silvia Pupeschi

Anno Accademico 2016/2017

RINGRAZIAMENTI

Devo ringraziare innanzitutto mia madre Tiziana e mio padre Francesco che mi hanno dato l'opportunità di seguire questo percorso e che mi hanno sostenuto in tutto e per tutto.

Ringrazio anche il mio compagno Gabriele che mi ha sempre donato la serenità e la forza di proseguire nelle difficoltà, donandomi preziosi momenti di felicità.

Un sentito grazie anche alla famiglia del mio compagno, Vito, Tiziana e Lavinia: la vostra accoglienza e il vostro affetto mi scaldano, giorno dopo giorno, il cuore.

Un pensiero anche alle mie amiche Sara, Giada, Giulia e Martina che mi hanno sempre sostenuta in questo lungo viaggio, strappandomi sorrisi e risate in ogni situazione.

Un ringraziamento speciale anche alle mie tutor e colleghe Annalisa, Gaia, Mariangela, Maria, Alessandra, Sabrina e Milena: la vostra infinita esperienza e sensibilità mi ha dato molta forza e sicurezza permettendomi di realizzare al meglio il mio progetto.

Infine un grande pensiero e grazie a coloro che fisicamente non possono essere qui, ma che porto sempre con me nel mio cuore: il nonno Angelo, che mi manca ogni giorno di più ma che sento sempre vicino in ogni mia scelta; la nonna Lidia, che sarebbe stata infinitamente orgogliosa di me; la nonna Nerina, che mi ha lasciato in eredità la sua grandissima forza di affrontare la vita nonostante tutti i problemi; la zia Rosaria, che ci ha lasciato troppo presto e troppo improvvisamente creando un vuoto incolmabile.

Nonostante la vostra dolorosa mancanza siete e sarete sempre le stelle che illuminano la mia strada, anche durante la notte più buia.

INDICE

INTRODUZIONE

Se un uomo ha fame gli puoi dare un pesce,
ma meglio ancora è dargli una lenza
e insegnargli a pescare.
Seymour Papert

Nel contesto educativo internazionale si sono andati delineando negli ultimi anni il concetto di pensiero computazionale e di *coding*, quindi è stata posta maggiore attenzione a tutte quelle pratiche che permettono di programmare un computer o uno strumento elettronico ad esso affine.

In Italia questo processo si sta delineando grazie alla disponibilità dei Fondi Strutturali Europei e della legge 107/2015, anche detta "La Buona Scuola", nella quale viene riposta maggiore attenzione nei confronti del *coding* e dell'insegnamento mediato da computer. L'interesse per questi temi ha spinto il Governo a reperire fondi per la sperimentazione attiva nelle scuole di queste pratiche: come si vede sul sito del Ministero[1] sono state suggerite varie aree tematiche in cui sperimentare questa *forma mentis*.

Discipline elettive del linguaggio computazionale sono la matematica e la geometria, ma si trovano alcune proposte anche nell'ambito della grammatica e della musica poiché si ritiene fondamentale incentivare queste metodologie di approccio logico anche a materie diverse dalla matematica.

La proposta del Governo, inoltre, prevede una progettazione in verticale tra i

1 https://labuonascuola.gov.it/area/m/5113/

vari ordini e gradi di scuola al fine di produrre in modo organico una conoscenza diffusa dell'informatica, partendo dalla scuola dell'infanzia e concludendo il processo alla scuola secondaria di secondo grado, passando da un'acquisizione delle nozioni di base negli ordini scolastici inferiori per poi raggiungere livelli più elevati negli ordini superiori.

Prendendo le mosse da questi obiettivi e raccordandoli con i traguardi e gli obiettivi della sezione Tecnologia delle Indicazioni Nazionali per il curricolo (2012) della scuola dell'infanzia e del primo ciclo di istruzione ho ideato un progetto didattico che è stato sperimentato presso l'Istituto Comprensivo Istituto Nord di Prato.

La scelta è ricaduta su questo istituto per due motivi: primo per la notevole strumentazione multimediale a disposizione e per la scelta delle maestre della classe in cui avrei svolto il tirocinio di inserire nel curricolo un progetto che trattasse di *coding*. Da questi due aspetti è scaturita l'idea di introdurre esercizi di stimolazione del pensiero computazionale nella classe per poter poi introdurre, di conseguenza, il concetto di figura geometrica, operando una distinzione tra contorno e superficie interna ed esterna. Tramite questo approccio sarebbero stati introdotti i germi di una conoscenza più radicata e profonda quale sono l'area e il perimetro dell'ente geometrico.

Grazie all'entusiasmo della mia Tutor Scolastica Annalisa Furzi e del prezioso aiuto del relatore Andreas Robert Formiconi, professore del Corso di Laboratorio di Tecnologie Didattiche, ho pensato di realizzare un progetto che unisse le conoscenze accademiche alla pratica didattica: in questo modo è stato possibile integrare il tirocinio e il progetto MARC ad esso collegato con la Tesi di Laurea, realizzando un prodotto unico che sintetizzasse i cinque anni di percorso.

Il lavoro scaturisce dalla mia profonda passione per le tecnologie, la matematica e le scienze in genere: di fatti la mia preparazione è di stampo

scientifico-informatico come testimonia il mio diploma di Liceo Scientifico con sperimentazione PNI, Piano Nazionale Informatica. Da sempre ho trovato molto interessante l'uso delle *Open Source*, per questo quando sono iniziate le lezioni del Corso di Laboratorio di Tecnologie Didattiche ho sentito sin da subito la necessità di lavorare con questi mezzi per la stesura della Tesi.

Andando avanti con la lettura della bibliografia messa a disposizione dal professor Andreas Robert Formiconi, ho ritrovato all'interno del suo pensiero e a quello di Seymour Papert (Formiconi, 2016; Papert, 1980) ciò che è ad oggi la mia idea di apprendimento, specialmente della matematica: una disciplina spesso insegnata come se fosse una "materia morta" (Papert, 1980) e che potrebbe beneficiare di una rivisitazione dell'uso di mezzi e metodologie per l'insegnamento della matematica, anche per sopperire alle necessità degli alunni con DSA, i quali sono, ad oggi, sempre più frequentemente presenti nelle nostre classi.

Attraverso il linguaggio *Logo*, ci si propone di esplorare alcuni concetti di base della didattica della matematica, come può essere la distinzione tra figure geometriche, oppure tra perimetro ed area, introducendo in modo intuitivo il concetto di unità di misura. Allo stesso tempo si propone un approccio che permetta all'alunno di muoversi all'interno di un percorso, di comprendere i vari punti di vista e di immedesimarsi in un corpo esterno che, grazie ad una serie opportuna di comandi, riesca a muoversi nello spazio circostante.

Questi obiettivi sono stati scelti secondo il pensiero di Papert, cioè di insegnare concetti matematici e/o geometrici complessi mediante l'uso del computer: è stato quindi naturale e il linguaggio Logo, da lui ideato. Per questi motivi aggiungiamo agli obiettivi di base anche la comprensione e la capacità di applicazione dei comandi di base del linguaggio *Logo* per produrre figure geometriche e distinguerne i contorni e le aree in modo intuitivo.

Nel primo capitolo viene inizialmente illustrato il contesto entro cui si inserisce

l'insegnamento del *coding*, illustrando le teorie relative ai *digital natives* e alle evidenze scientifiche in merito. Successivamente viene descritto cosa sia il pensiero computazionale e il *coding*, proponendo una distinzione tra questi due termini troppo spesso usati come sinonimi. A conclusione del primo capitolo si ritrova una descrizione del pensiero pedagogico di Seymour Papert, l'ideatore del linguaggio *Logo* e fautore dell'apprendimento mediato dal computer.

Nel secondo capitolo viene approfondita la storia del linguaggio *Logo* e nel paragrafo successivo saranno descritti i comandi e le funzionalità legate ai *software* che prevedono questa programmazione. Nell'ultima parte del capitolo viene fatta una distinzione tra *TurtleArt* e *Scratch*, illustrando le motivazioni che hanno portato a scegliere un *software* di programmazione a blocchi nonostante sia preferibile utilizzare la tipologia testuale.

Nel terzo ed ultimo capitolo viene illustrato il progetto ideato ed effettivamente realizzato. Nel primo paragrafo sono esposti gli aspetti teorici e, nei paragrafi successivi, verranno descritte le attività che sono state effettivamente svolte in aula, quindi le variazioni apportate in corso d'opera grazie ai *feedback* ricevuti dagli alunni volta per volta nel corso delle lezioni.

Infine si trova una sezione in cui vengono illustrate le criticità e le potenzialità del progetto sostenuto e in cui sono presentate delle proposte didattiche da promuovere in continuità non solo con le classi successive, ma anche con le precedenti. In questo modo è possibile seguire e ravvisare in esse il pensiero di Papert, implementando e promuovendo il pensiero computazionale, il quale risulta essere l'elemento fondamentale per costruire la logica promossa dall'ideatore del linguaggio *Logo*.

CAPITOLO 1
IL *CODING* A SCUOLA: COME, QUANDO E PERCHÉ

Se uno crede abbastanza fermamente di non poter fare matematica,
avrà quasi sicuramente successo nell'impedirsi di fare
qualsiasi cosa che gli paia attinente alla matematica.
La conseguenza di tale autosabotaggio è il fallimento personale,
e ogni fallimento rinforza l'assunto di base.
Seymour Papert

1.1 LA SCUOLA E LA DIDATTICA NELL'ERA DEI *DIGITAL NATIVES*- Per comprendere la necessità della Commissione Europea di produrre il FOAM 2014-2020, un documento esplicativo dell'implementazione del pensiero computazionale a scuola, bisogna prima capire il contesto entro cui si inserisce.

Negli ultimi anni è fiorita una vasta letteratura riguardante le nuove generazioni di fruitori di dispositivi elettronici: è evidente che si stia parlando dei cosiddetti *digital natives* (Prensky, 2001) -nativi digitali. I nativi digitali sono degli individui nati nell'era di massimo sviluppo dei nuovi strumenti tecnologici, quindi tutti coloro che fanno parte- a seconda delle varie definizioni- delle generazioni nate dal 1980-1990 in poi.

Il primo a denominare questo fenomeno fu Prensky nel 2001, tuttavia molti autori hanno dato altre definizioni ed espressioni per descriverlo, nonostante ciò, l'unica cosa che varia tra ogni descrizione è l'età anagrafica da considerare come spartiacque tra una generazione -quella dei *digital natives*- e l'altra -quella degli

immigrati digitali anche detti *digital immigrants* (Prensky, 2001).

Prensky, inoltre, ritiene che i nativi digitali siano anche dei *native speakers*, cioè individui capaci di comprendere i linguaggi multimediali: il mondo di questi soggetti è pervaso da tutti quegli strumenti tipici dell'era digitale, quindi per loro è naturale recepire in modo più efficace questa tipologia di comunicazione.

Secondo questa filosofia (Prensky, 2001) si ritiene che le nuove generazioni di studenti abbiano sviluppato nuove capacità cognitive grazie all'uso intensivo dei nuovi e sempre più sofisticati *device*: i fautori di questo fenomeno ritengono che i *digital natives* non soffrano il sovraccarico informativo, che siano capaci di una migliore valutazione delle informazioni e di rielaborarle in modo originale (Veen e Vrakking, 2006).

Altri autori ritengono che i nativi digitali sviluppino forme più divertenti di apprendimento (Tapscott, 1998), riescano a pensare in modo più veloce grazie alla *serendipity* (Small e Vorgon, 2008) e che siano abili nel padroneggiare le capacità di *multitasking*, cioè quella capacità di processare più informazioni simultaneamente (Veen e Vrakking, 2006).

Prensky (2001), infine, ritiene che tutte queste nuove capacità abbiano portato ad una "morte della pazienza" e alla necessità di relazioni basate sulla gratificazione istantanea. Tutte queste nuove capacità, inoltre, secondo i fautori di queste teorie, andrebbero sviluppate tramite delle istituzioni formative rinnovate e ristrutturate *ad hoc* per rispondere al meglio a questi bisogni.

È evidente come, secondo queste tesi, l'educazione debba adeguarsi ai cambiamenti della tecnologia, quindi le nuove scoperte non sono un possibile mezzo con cui veicolare l'insegnamento, ma sono lo strumento che lo studente e l'insegnante devono conoscere per poter accedere all'apprendimento. L'insegnamento tradizionale non è più il miglior metodo per apprendere, quindi viene preferito quello mediato dal computer (Prensky, 2001; Veen e Vraking, 2006).

Contro queste tesi alcuni autori come Benett et al (2008), invece, hanno sostenuto che esse fossero pervase dal cosiddetto *moral panic* teorizzato nel 1972 da Cohen. In accordo con questa ipotesi, gli studiosi ritengono che il fenomeno dei nativi digitali sia stato portato alla ribalta con un unico scopo, quello di criticare le istituzioni scolastiche tramite rappresentazioni fin troppo semplicistiche e non sempre corrette, sollevando l'opinione pubblica sul tema. La nascita di questa discussione, tuttavia, resta sterile e inconcludente e, alla fine del ciclo, si ritorna alla situazione iniziale senza sostanziali modifiche sul tema (Bennet et al, 2008).

Inoltre anche i *Net Gen* scettici (Ranieri, 2011), cioè tutti coloro che dubitano delle tesi sui nativi digitali, ritengono che non sia legittimo parlare di "generazione" poiché l'accesso alle nuove tecnologie non sembra essere sempre semplice e alla portata di tutti. Di fatti tutt'oggi è presente il *Digital Divide* -divario digitale-, un fenomeno descrivibile come una diversa opportunità di accesso alle tecnologie da parte dei giovani sia all'interno di uno stesso paese che a livello globale, generando, di fatto, diverse opportunità. (Bennett e Maton, 2010).

Dal punto di vista scientifico le evidenze, di fatto, mettono in discussione gli assunti di base sulle tesi dei nativi digitali, in particolare per quanto riguarda l'evoluzione autonoma della tecnologia e della sua capacità insita di mutare antropologicamente e socialmente le relazioni umane.

Alcune evidenze empiriche sulle pratiche tecnologiche smentiscono di fatto queste tesi, specialmente quella di Kennedy et al (2010) che si proponeva di classificare le diverse tipologie di studenti universitari secondo le loro capacità digitali. Secondo gli autori, si possono individuare quattro fasce di pratica -avanzati, ordinari, irregolari, base. In questo modo le tesi sui nativi digitali sarebbero state confermate grazie all'appartenenza della maggior parte degli studenti alla tipologia più alta di utenza, ma i risultati non hanno assolutamente portato a questa conclusione.

Dalle analisi dei dati, infatti, emerse sin da subito come la situazione fosse completamente diversa: il 45% degli studenti apparteneva alla categoria degli utenti di base e solo un 14% a quella degli avanzati, quindi la fascia media degli studenti si collocava nella tipologia più bassa di utenza e non in quella più alta.

Le ricerche di Bennet et al (2008), inoltre, evidenziano un differente accesso ed uso delle tecnologie anche in base ai fattori socio-economici e al *background* culturale degli utenti: i risultati delle indagini mostrano come la variabile dell'età non sia significativa.

Da questo breve *excursus* è evidente che l'opposizione dei *digital native* contro gli "immigrati digitali" risulta essere scorretta (Ranieri, 2011) e anche Prensky (2009) ha ritenuto opportuno non parlare più di nativi digitali, ma di "saggezza digitale". In questo modo si svincola il fenomeno dai fattori generazionali, rendendo di fatto questa capacità come obiettivo stesso dell'educazione nell'era digitale.

Altre ricerche mettono in dubbio anche la relazione tra uso intensivo di tecnologie e sviluppo di abilità digitali: studi internazionali (Flanagin e Metzeger, 2008; Eastin, 2008) evidenziano come manchi da parte dei giovani fruitori un'interrogazione spontanea sulla qualità dei contenuti trovati in rete, mostrando mancanza di valutazione critica delle fonti.

Questa incapacità di critica è in netto contrasto con i fautori delle tesi sui nativi digitali: le *Net Generation* avrebbero dovuto spontaneamente aver maggiore cognizione e discernimento tra le varie fonti individuate in una ricerca su *Internet,* non mancare completamente di interrogarsi su un tema tanto sensibile.

In contrasto con gli assunti dei fautori del *multitasking* emerge nelle evidenze di Carr (2011) e Ophir et al (2009) come l'uso simultaneo di più dispositivi elettronici riduca i livelli di attenzione. Quando la mente deve processare troppe informazioni simultaneamente, infatti, si hanno maggiori difficoltà ad esercitare un controllo cognitivo efficace sulle proprie operazioni, producendo forme di pensiero più

superficiale e non migliorando le capacità di *multitasking*.

Per quanto concerne gli stili di apprendimento e il loro sviluppo mediante l'uso delle nuove tecnologie Calvani (2010) afferma come manchi per essi una consistenza scientifica per la loro effettiva esistenza. Anche Clark et al (2006) affermano che "Gli stili di apprendimento appartengono ad un tipo di mitologia istruttiva improduttiva per la formazione. Nella migliore delle ipotesi la maggior parte dei corsi sugli stili di apprendimento sono un dispendio di risorse, nella peggiore conducono a metodi istruttivi che ritardano l'apprendimento. Un buon esempio sarebbe che esistono stili di apprendimento visivo e verbale. Non esistono tuttavia evidenze solide sull'argomento".

Tutte queste tesi sulle potenzialità e necessità dei nuovi studenti nell'era di nativi digitali hanno portato a formulare varie dicotomie (Ranieri, 2011) che, di fatto, sono state smentite dai vari studi già citati. Le opposizioni descritte da Ranieri (2011) sono di fatto opportunità da cogliere per migliorare l'apprendimento mediante l'uso dei nuovi *device*, senza dover necessariamente vivere i contrasti tra i fautori e gli oppositori alla *Net generatiom*.

Concludendo si può affermare che non esista una vera e propria generazione di nativi digitali, ma persone che hanno avuto l'opportunità di accedere alle nuove tecnologie grazie ad un favorevole *background* culturale e sociale. Non esistono, inoltre, neanche delle capacità legate all'uso delle nuove tecnologie né è possibile sviluppare una competenza digitale grazie ad esse, tuttavia nell'era in cui stiamo vivendo risulta essere necessario una nuova educazione all'uso di queste ultime.

Da queste conclusioni emerge la necessità di sviluppare una nuova consapevolezza all'uso dei mezzi digitali: per questo è bene prendere spunto dalle normative europee al fine di formare nuove generazioni maggiormente consapevoli dei mezzi a loro disposizione. La normativa, comprendendo come la scuola stessa sia un punto di riferimento cruciale in questo percorso, ritiene che le istituzioni di ogni

ordine e grado debbano fornire i mezzi per un primo *imprinting* con gli strumenti tecnologici, sia per fornire a tutti le stesse opportunità, sia per dare una corretta formazione di base al loro utilizzo.

1.2 IL *CODING* E IL PENSIERO COMPUTAZIONALE- Come già affermato nell'introduzione, ad oggi si parla molto di *coding*, tant'è che sono state promosse iniziative europee e sono stati formulati e incentivati progetti anche dal precedente Governo all'interno della Legge 107/2015. Nonostante ciò non sempre si parla di *coding* con cognizione di causa poiché è un termine che viene utilizzato con molte accezione di significato.

Per poter parlare di *coding* è necessario prima occuparsi dei processi mentali sottostanti questo procedimento, quindi bisogna introdurre il concetto di *Computational Thinking* anche tradotto come pensiero computazionale.

Questa forma di pensiero è un'abilità di base per ognuno di noi e come tale deve essere appresa e potenziata sin dall'infanzia poiché non è posseduta unicamente dal tecnico informatico, ma rappresenta la capacità di risolvere i problemi in modo efficiente. Il *Computational Thinking* prevede la riformulazione del problema in modo tale da trasformarlo o promuoverlo in forma simulata e più vicina al reale (Wing, 2006) ed è una capacità utilizzabile in ogni aspetto della vita quotidiana.

Il pensiero computazionale è quindi un'astrazione che permette di scegliere il metodo di rappresentazione più opportuno al fine di renderlo maggiormente risolvibile, senza modificare gli aspetti rilevanti presenti in esso. Questa *forma mentis* permette di evitare la ridondanza, che è estremamente svantaggiosa poiché sovraccarica eccessivamente il carico cognitivo (Calvani, 2011).

Questo processo, inoltre, non è vantaggioso unicamente per quanto concerne la

matematica, ma anche per l'apprendimento della grammatica, della biologia, della chimica, della statistica e di ogni disciplina umana: il pensiero computazionale influenza radicalmente e profondamente tutti gli aspetti della vita (Wing, 2006).

Per comprendere la sua importanza è sufficiente pensare a quando ci dimentichiamo dove abbiamo messo le chiavi di casa: ripercorriamo fisicamente le azioni che sono avvenute, in modo tale da capire ed individuare il posto in cui le abbiamo lasciate (Wing, 2006). Questa metodologia di pensiero è assimilabile ad un algoritmo in cui ripercorriamo i vari *step*, cioè i passi, che ci hanno permesso di arrivare ad ora, ma la cosa più sorprendente è che questa forma di *back-tracking* ci fornisce -quasi in modo automatico- la soluzione al nostro problema.

Ma quali sono le caratteristiche del pensiero computazionale? Secondo Jeannette M. Wing, vicepresidente della Microsoft Research, già rettrice del Corso di Scienze Informatiche presso la Carnegie Mellon University, il pensiero computazionale possiede le seguenti caratteristiche (Wig, 2006):

· Concettualizza, non programma (*conceptualizing, not programming)*: il pensiero computazionale permette di operare un pensiero a livelli multipli di astrazione, non è la mera programmazione fine a sé stessa;

· È fondamentale, non è un'abilità di routine (*Fundamental, not rote skill*): il pensiero computazionale è un'abilità fondamentale che permette ad ogni essere umano di operare in modo soddisfacente all'interno della società moderna, senza incappare in routine meccaniche e non sempre adeguate alle richieste;

· Un modo di pensare umano, non del computer (*A way that humans, not computers, think*): il pensiero computazionale è il mondo in cui ognuno di noi risolve i problemi, quindi non è assolutamente un tentativo di far assomigliare il pensiero umano ad un computer. I computer non hanno immaginazione e creatività, mentre gli essere umani sono i soli che possono rendere eccitanti i dispositivi elettronici: senza l'immaginazione dell'uomo non esisterebbe alcuna macchina;

· Completa e si combina con il pensiero matematico e progettuale (*Complements and combines mathematical and engineering thinking*): il pensiero computazionale non coincide con il pensiero matematico o progettuale: è evidente che un computer sia disegnato sulla matematica, ma l'interfaccia con il reale ha un sistema di interazioni ben più complesso che porta chiunque a rapportarsi con i dispositivi elettronici tramite una forma di *Computational Thinking*. Per questo motivo il pensiero computazionale si affianca e completa il pensiero logico-matematico, interagendo anche con il pensiero progettuale.

· Idee, non artifici (*Ideas, not artifacts*): il pensiero computazionale non è presente unicamente nei *software* e negli *hardware* che ogni giorno utilizziamo fisicamente, ma è un metodo di approccio e di risoluzione dei problemi che adoperiamo nella nostra vita di tutti i giorni. Non solo permette di approcciarsi e risolvere i problemi che si presentano, ma ci fornisce sia gli strumenti per organizzare e gestire i nostri impegni, sia per interagire con le altre persone;

· Per tutti, dappertutto (*For everyone, everywhere*): il pensiero computazionale permette di valorizzare i tentativi di ciascuno, quindi non deve essere relegato ad una specifica tipologia di persone o di contesti di appartenenza. Questa filosofia deve scomparire come tale per poter essere assimilata nel modo di pensare di ciascuno, valorizzando l'attitudine a fare tentativi e a volgere gli errori in occasioni di approfondimento dei problemi.

A partire da questa definizione di pensiero computazionale fornita da Jeannette M. Wing nel 2006 -il primo ad usare questo termine fu lo stesso Papert nel 1996 senza però definirlo in modo approfondito- è nata una vasta letteratura.

Sull'argomento sono state promosse numerose ricerche e il concetto di pensiero computazionale è diventato molto popolare non solo nelle aule accademiche, ma anche nelle scuole di ordine e grado inferiori, lasciando emergere una pressante necessità di delimitare il dominio del *Computational Thinking* all'interno dell'ambito

dell'istruzione.

Le evidenze affermano come manchi una maturità della letteratura scientifica sull'argomento (Kalelioğlu et al, 2016) e, alla soglia dei dieci anni dalla definizione fornita dalla vicepresidente della Microsoft Research, si può affermare che la questione del pensiero computazionale all'interno dell'educazione sia, a tutt'oggi, ancora aperta.

La *review* di Kalelioğlu et al, nonostante sia priva di supporti teorici adeguati e manchi di riscontri sperimentali, permette di capire i limiti delle ricerche precedentemente effettuate e, assieme al lavoro svolto da Weintrop et al (2016), consente di riscontrare la gravità della mancanza del pensiero computazionale all'interno dell'istruzione: le varie definizioni si susseguono una appresso all'altra, senza fornire con chiarezza una spiegazione univoca di tale forma di pensiero.

Alcuni studi ritengono che il *Computational Thinking* sia un processo di *problem solving* che include la formulazione di problemi non risolvibili tramite un computer ma che necessitano di abilità intuitive e non schematiche (Kalelioğlu et al, 2016). In questo processo sono coinvolte tutte quelle abilità come l'organizzazione logica, l'analisi dei dati, la rappresentazione tramite modelli e simulazioni, la capacità di costruire processi sequenziali, l'identificazione e la risoluzione tramite la combinazione di possibili risorse (Kalelioğlu et al, 2016).

In accordo con questa definizione, si può aggiungere che il pensiero computazionale si ritrovi in quell'abilità di pensiero procedurale che formula problemi e ne individui le soluzioni in qualsiasi disciplina (Mannilla et all, 2014).

In definitiva si può affermare che il pensiero computazionale sia la forma di pensiero più affine alle strategie di *problem solving* grazie alla sue proprietà intrinseche di creatività e di astrazione (Wing, 2006): questo è anche il motivo per cui molto spesso troviamo abbinato al *coding* il termine *Computational Thinking*.

Una volta compreso che il termine *coding* non coincide con il *Computational*

Thinking, è necessario porsi alcune domande per comprendere meglio la differenza.

Il *coding*, talvolta erroneamente identificato con programmazione, è un termine che indica il processo di scrittura di istruzioni su un computer al fine di far eseguire un programma. Invece la programmazione si riferisce all'attività di analisi di un problema, della sua risoluzione e della scrittura di un algoritmo per attuarla: quest'ultimo processo si definisce come *coding* e necessita di un suo specifico linguaggio (Bocconi et al, 2016).

È evidente come emerga la differenza sostanziale tra il *coding* e il pensiero computazionale: il primo risulta essere l'attività di comporre una serie di istruzioni in linguaggio di programmazione per eseguire un *software*, tuttavia questa pratica necessita del secondo elemento per essere messa in atto.

Questi due elementi sono differenti, ma sono uno collocato all'interno del processo dell'altro: nonostante il pensiero computazionale sia più esteso del dominio della *Computer Science*, esso risulta essere l'unico mezzo per poter mettere in atto le pratiche e le attività legate al linguaggio di programmazione.

Il pensiero computazionale è, quindi, l'abilità necessaria per la messa in pratica del *coding* e non il contrario, quindi quest'ultimo risulta essere un'attività legata al "fare codice", cioè alla creazione di sequenze ordinate al fine di far eseguire un programma.

Nella programmazione, tuttavia, sono presenti altri processi che permettono di ottenere una buona riuscita della funzionalità di un *software* e si riferiscono ad altre capacità mentali sempre annesse al *Computational Thinking*. Questi processi sono noti come *testing* e *debugging*.

Il *testing* è quell'attività che permette di collaudare l'affidabilità del *software* in via di sviluppo che necessita, in caso di errori e imprecisioni, la fase di *debugging* cioè di individuazione e correzione dell'errore. Queste due attività sono fondamentali ed importanti quanto il *coding* in sé e per sé perché stimolano la metacognizione e lo

sviluppo di abilità di *problem solving* e, nelle attività portate avanti in fase sperimentale, sono state affrontate dai discenti in modo intuitivo e spontaneo, focalizzando l'attenzione unicamente sull'attività di *coding*.

Il processo di *coding*, quindi, non è scindibile da quello del *testing* e del *debugging*, ma l'approccio intuitivo degli alunni permette loro di affrontare spontaneamente queste fasi, accedendo a conoscenze più profonde.

1.3 PENSIERO DI SEYMOUR PAPERT- Arrivati a questo punto è necessario per comprendere la strutturazione del progetto un'introduzione al pensiero di Seymour Aubrey Papert, grande matematico del secolo scorso, ideatore del linguaggio *Logo*. Fondamentale per la formazione del suo carattere, al confine tra matematica e pedagogia, fu la sua collaborazione con Jean Piaget presso il laboratorio di epistemologia genetica dell'Università di Ginevra. Questo incontro permise al matematico di conoscere più approfonditamente la pedagogia, in particolare quella piagetiana che ritroviamo citata in varie parti della sua opera *Mindstorms. Children, Computers and Powerful Ideas* (Papert, 1980), nel quale spiegò minuziosamente il suo pensiero.

Come base di tutta la sua filosofia, Papert prende le mosse dall'idea di non identificare la matematica con il mero e banale far di conto (Papert, 1980) poiché il pensiero matematico non è la capacità di eseguire calcoli a mente in tempi brevi: è molto più ricco e profondo di un semplice tipo di allenamento mnemonico perché prevede anche momenti di originalità e di intenzione che non sono assolutamente rigidi.

All'interno di *Mindstorms. Children, Computers and Powerful Ideas* ritroviamo un capitolo molto interessante a riguardo dell'insegnamento e dell'apprendimento

della matematica: *Mathphobia: The Fear for Learning*, in cui Papert condanna la divisione operata nel mondo occidentale tra le diverse materie scolastiche causando, a suo dire, un terrore per la matematica a causa del suo cattivo insegnamento.

Nella nostra cultura, infatti, ci siamo dimenticati la radice e anche il significato del termine *math*: questa parola deriva dal greco e indica l'apprendimento nella sua forma più generale, quindi non si riferisce unicamente ai numeri (Papert, 1980). La nostra formazione offre un modello dissociato di apprendimento, in cui la parte verbale viene separata dalla parte matematica, impedendo, di fatto, ai bambini di comprendere appieno ciò che studiano.

Nel momento in cui manca la comprensione di ciò che si studia, automaticamente si generano ansie e paure legate all'apprendimento, quindi nasce un netto rifiuto di approfondire e capire ciò che si ha davanti, portando l'alunno a dileguarsi dal mondo della formazione, specialmente in campo matematico.

La paura dell'apprendimento e la nascita di superstizioni sulla propria capacità di apprendere, creano un mondo di tabù in cui ci si arricchisce di opinioni negative su sé stessi. Queste opinioni negative di sé, a loro volta, si radicano profondamente nella mente dell'individuo: chi soffre di questa tipologia di ansie mette in atto dei comportamenti che tendono ad autoconfermarsi e, di fatto, si perde di fiducia ed ulteriore autostima. Come afferma Papert, la cosa peggiore è che "La conseguenza di tale autosabotaggio è il fallimento personale, e ogni fallimento rinforza l'assunto di base. Ancora più insidiosi sono i pregiudizi che appartengono non solo agli individui, ma a un'intera cultura." (Papert, 1980; Formiconi, 2016, pagina 16).

Ma queste convinzioni trovano riscontri reali? Alcuni individui sono davvero incapaci di apprendere? Secondo Papert (1980) la risposta a questa domanda è negativa perché tutti i bambini possiedono i presupposti di base per comprendere il mondo, dalle parole ai numeri riferendosi al pensiero di Jean Piaget legato alle varie fasi dell'apprendimento.

L'ottica piagetiana da lui citata si basa su un continuo processo di apprendimento che si sviluppa dalla nascita alla morte: per dimostrare la valenza di queste posizioni Papert porta numerosi esempi ed evidenze in questa direzione (Papert, 1980). Ad esempio, già in età prescolare si ha un'acquisizione di una grandissima quantità di parole senza che vengano insegnate, inoltre anche il conteggio viene appreso spontaneamente dal momento che viene sviluppata l'idea di conservazione (Papert, 1980).

Un altro esempio delle capacità dei bambini di apprendere da soli, che Papert trae sempre dal pensiero di Jean Piaget, è quello dello sviluppo di una geometria intuitiva da parte dei bambini, che permette loro di muoversi in ambienti a loro familiari, o ancora come usino la logica per trarre vantaggio con i loro genitori senza che venga loro insegnato niente (Papert, 1980).

Questo processo viene chiamato da Papert “apprendimento piagetiano” o “apprendimento senza insegnamento” e non necessita dell'istruzione formale (Papert, 1980): il bambino è come un costruttore che trova la sua fonte nel contesto culturale da cui trae il materiale, per cui in presenza di sovrabbondanza di stimoli si ha un apprendimento celere, in caso contrario una lentezza di sviluppo dettata proprio dalla carenza di tali risorse.

Lo stesso autore afferma che l'apprendimento piagetiano si manifesta in ogni attività e come il bambino apprende la lingua madre, anche la matematica subisce la stessa tipologia di processo: l'unico problema che sorge è l'insegnamento dissociato che permane nell'approccio tradizionale della scuola. Questa tipologia di apprendimento genera una separazione tra elementi che in realtà sono uniti e che si completano vicendevolmente, generando lacune ed incomprensioni gravi (Papert, 1980).

Lo stesso autore aggiunge che un altro limite nell'apprendimento della matematica è veicolato dagli insegnanti stessi: pochi docenti riescono a giustificare e

a spiegare il perché si insegni questa materia tanto controversa. L'insegnamento della matematica scolastica e la comprensione del perché serva ad ognuno di noi è fondamentale: se si vuole veicolare in modo efficace questa tipologia di apprendimento bisogna innanzitutto capirlo a fondo.

Questa grave mancanza genera un ulteriore problema, cioè il tradimento della fiducia del bambino all'interno della relazione educativa poiché si rendono conto che l'insegnante, come loro, non comprende e non sa il perché stia spiegando e veicolando questo tipo di apprendimento, trasponendolo come un mero procedimento di operazioni poste in sequenza.

Come lo stesso Papert afferma: "La maggior parte dei bambini si rende conto che l'insegnante non ama la matematica più di quanto la amino loro e che la ragione per cui va fatta è semplicemente perché lo prevede il curricolo. Tutto ciò erode la fiducia dei bambini nel mondo degli adulti e nel processo di educazione. *E io penso che introduca un elemento di profonda disonestà nella relazione educativa*.[2]" (Papert, 1980; Formiconi, 2016, pagina 21).

A differenza dei suoi predecessori, Papert (1980) interpreta la matematica come costruzione sociale, un bagaglio inalienabile che ogni individuo deve fare suo per vivere nel mondo. L'insegnante di matematica, tuttavia, deve veicolare questa particolare materia non più come se fosse una "materia morta", ma ricondurre questo apprendimento alla nostra epoca (Papert, 1980). Di fatto è più semplice operare un continuo riscontro sulla correttezza delle operazioni matematiche, come se fossero parole di una lingua antica, ma questo processo inaridisce le mente e propone continui esercizi ripetitivi e facili da valutare che non stimolano l'apprendimento degli allievi (Papert, 1980).

In accordo col pensiero fin qui descritto, Papert sottolinea la necessità di sviluppare una nuova matematica che chiama *Mathland* (Papert, 1980), dove il

2 Corsivo dell'autore

modello di apprendimento tradizionale venga spazzato via da nuovi metodi e nuovi strumenti.

Tramite l'uso del computer, mezzo principale per una rivoluzione completa dell'apprendimento, viene superata la dissociazione tra pensiero verbale e quello matematico promosso dagli ultimi secoli di cultura occidentale. Il computer, infatti, permette di creare ambienti in cui si hanno "conversazioni matematiche" (Papert, 1980), le quali risultano essere molto liberatorie perché permettono di far sperimentare anche ai più restii la possibilità di realizzare e fare cose che fino a poco tempo prima erano ritenute troppo difficili da fare da soli.

Grazie all'avvento prima del computer e di internet poi, si è andata consolidando negli anni l'idea che "la conoscenza è vista risiedere nelle persone, negli artefatti, nei *setting* culturali, in una specie di "rete"" (Calvani, 2007, p. 48-49), promuovendo una nuova attenzione al contesto.

L'uso di un mezzo come il computer, secondo Papert, è di fondamentale importanza per comprendere e sviluppare al meglio il concetto di padronanza della matematica e, di conseguenza, permette di ridurre al massimo le interferenze causate dall'insegnamento tradizionale, aumentando, di fatto, la conoscenza acquisita (Calvani, 2011).

Papert (1980) ha compreso l'importanza dell'ambiente di apprendimento e per questo ha sviluppato un linguaggio che promuovesse l'acquisizione della conoscenza tramite riferimenti reali e non astratti come la vecchia matematica presupporrebbe.

Da tutte queste serie di presupposti Papert (1980) ha ideato l'ambiente *Logo* in cui nessuna attività è riconducibile direttamente alla matematica, ammorbidendo e superando le distinzioni tra un pensiero ed un altro. *Logo* è considerato il punto di incontro tra i due mondi e, al tempo stesso, il mezzo per veicolare la conoscenza che, normalmente, viene rifiutata da chi ha avuto esperienze negative con l'apprendimento della matematica.

Nella *Mathland* non vi è più spazio per una matematica antica, fatta di gesso, lavagna, carta. e matita, ma vi ritroviamo un nuovo mezzo, il computer con il linguaggio *Logo* e la *Turtle Geometry* -la Geometria della Tartaruga- in cui si veicolano concetti e argomenti coerenti e liberi da ogni vecchia reminiscenza. Nel computer ritroviamo un mezzo "matematicamente espressivo" (Papert, 1980) in cui non si impara la matematica, ma la si ricostruisce attraverso nuove strade che aggirano la difficoltà di insegnarla.

Papert (1980) con la Geometria della Tartaruga propone una riforma della matematica scolastica come sfida per la creatività e l'intuizione: la cosiddetta *New Math* operata e promossa nel corso degli anni, è il risultato di una profonda selezione della vecchia matematica, quindi non solo l'insegnamento non si è rinnovato, ma è stato riproposto sintetizzato e sempre descritto con i soliti termini (Papert, 1980).

In questo modo Papert ribadisce la sua intenzione di non avvalersi di un metodo tradizionale di insegnamento, anzi, nel capitolo successivo della sua opera, *Turtle Geometry: A Mathematics Made For Learning*, partendo dal concetto di punto euclideo, Papert (1980) propone una similitudine basata sulla "Geometria della Tartaruga".

Il concetto euclideo del punto, infatti, non è sempre di facile comprensione perché viene considerato bizzarro e con poche analogie con il contesto del mondo reale, sia perché non è mentalmente figurabile come un oggetto -non ha né colore, né forma, né dimensione- sia perché ha in sé solo la proprietà della posizione. Il concetto di punto, tuttavia, risulta essere il fondamento dell'intera geometria di Euclide e, per questo, è il tassello fondamentale di tutta la geometria tradizionale.

Nella Geometria della Tartaruga, invece, ci riferiamo a qualcosa che già si conosce -la Tartaruga- che risulta essere l'entità fondamentale di tutta la geometria papertiana (Papert, 1980). Questa entità, a differenza del punto euclideo, possiede proprietà che chiunque conosce perché ha in sé sia la posizione che la direzione. Non

è un concetto statico, ma è dinamico e permette di ricondursi sia alla realtà che di immedesimarsi in esso.

Questa grande concessione permette al bambino di individuare una prima rappresentazione formale nella Tartaruga: “I bambini si possono identificare con la Tartaruga e così sono in grado di traslare la conoscenza che hanno del loro corpo e di come si muovono nell'attività di apprendere la geometria formale (Papert, 1980; Formiconi, 2016, pagina 26).”

L'immedesimarsi e il riuscire ad accettare questi concetti proposti in modo più semplificato e intuitivo, permettono di operare sui movimenti della Tartaruga, che sono espressi in linguaggio *Logo*. L'apprendimento di questo linguaggio permette al bambino di interfacciarsi con la geometria classica senza che gli venga effettivamente insegnata: la risoluzione dei problemi in questo ambiente con un riscontro reale ed immediato veicola un insegnamento diretto e senza subire la frustrazione dell'errore (Papert, 1980).

Da questo pensiero scaturiscono, di conseguenza, nuovi criteri sottostanti la *Mathland*, quindi la Geometria della Tartaruga avrà basi concettuali completamente diverse e marcatamente distinte rispetto a quelle della *New Math*. Per operare in modo consapevole e concreto nella *Mathland* bisogna che valgano i seguenti criteri (Papert, 1980):

- Principio di appropriabilità: gli insegnamenti dovranno essere adattati e adattabili al bambino. Il bambino, quindi, non dovrà più adattarsi all'insegnamento ma sarà l'insegnamento stesso ad adattarsi a lui, quindi ogni argomento dovrà prendere le mosse dalla sua capacità di adattarsi adeguatamente agli allievi.
- Principio di continuità: la matematica proposta non deve essere isolata dalle altre conoscenze, ma deve ereditare da esse familiarità e valore. È necessario superare la dicotomia tra le varie materie scolastiche al fine di raggiungere

una vera competenza in ogni apprendimento.

- Principio di potenza: gli strumenti che vengono dati all'allievo devono permettergli di affrontare i suoi progetti. Le conoscenze e gli strumenti usati devono permettere di raggiungere obiettivi personali che sarebbero impossibili da affrontare con la vecchia formazione matematica.
- Principio di risonanza culturale: gli argomenti trattati attraverso la Geometria della Tartaruga non devono essere scissi dal contesto sociale, anzi, vi devono partecipare con un senso culturale allargato. Non più una matematica lontana dalla vita dei bambini, ma una matematica che vive attorno e con essi nei loro contesti sociali.

In definitiva si può affermare che il pensiero di Papert prevede una riduzione al minimo dell'insegnamento al fine di massimizzare l'apprendimento in un'ottica costruzionista: a livello pedagogico è una variante del costruttivismo ma, a differenza di quest'ultimo, afferma che la costruzione delle conoscenze trova maggiore valenza in un contesto in cui il soggetto si impegna nella produzione di qualcosa di concreto e condivisibile (Calvani, 2007), dove trova gli artefatti necessari per le proprie costruzioni.

La didattica promossa da Papert privilegia il concetto di "usare per imparare" e "non apprendere per applicare, ma fare per imparare" (Papert, 1980), ne consegue una preminenza dell'apprendimento attivo sull'insegnamento in un'ottica costruzione della conoscenza (Papert, 1980).

Attraverso il linguaggio *Logo* si privilegia una didattica basata sul *problem solving* e sul *problem finding*: l'alunno si trova spesso di fronte a vari problemi che devono essere risolti ed aggiustati, ma tramite l'uso della Tartaruga non vengono percepiti come tipicamente matematici e vengono risolti in modo non convenzionale (Papert, 1980). La pedagogia di Papert, quindi, si basa sull'errore non in accezione negativa, ma in un'ottica di costruzione di un sapere utile, condiviso, pratico ed

intenzionale, applicabile a contesti concreti (Papert, 1980).

Il fine del costruzionismo di Papert è la capacità di ciascun individuo di imparare e di saper risolvere anche situazioni che vadano ben oltre il mero ambito scolastico: "*The one really competitive skill is the skill of being able to learn*" (Papert 1998).

Il pensiero di Papert ha ampio respiro ed è completo sotto ogni punto di vista del contesto sociale e culturale di ciascun allievo, tuttavia ha fallito nei suoi intenti (Formiconi, 2016): gli insegnanti non erano né pronti né preparati alla *Mathland* e alla comprensione della struttura di *Logo*. La carenza nella formazione del docente e la cattiva ricezione e propensione ad accettare queste tipologie di insegnamenti ha portato *Logo* a fallire nei suoi intenti (Formiconi, 2016), non realizzando, di fatto, la *Mathland* sognata da Papert.

Papert stesso aveva già valutato l'opzione di un suo possibile fallimento, attribuendo la possibilità di non riuscire a completare la sua rivoluzione a problematiche proprie del mondo degli adulti: egli stesso affermò che "[...] tutto questo non avverrà se (la Geometria della Tartaruga) non viene accettata anche dagli adulti" (Papert, 1980; Formiconi, 2016, pagina 23).

Tuttavia questo pensiero ha lasciato una profonda traccia in tutto il mondo poiché sono nate e sono state sviluppate innumerevoli versioni di *Logo* anche in campo didattico (Formiconi, 2016): basta pensare a *Scratch* e *Snap!* e il loro largo utilizzo in progetti e attività didattiche in tutto il mondo, ma le loro analogie con l'ambiente papertiano verranno approfondite nel capitolo successivo.

Il pensiero di Seymour Papert e il mondo del linguaggio *Logo*, nato dalle sue innovazioni e teorizzazioni, sono fondamentali per promuovere una nuova didattica della matematica, soprattutto alle soglie di questa nuova era pervasa dall'uso non sempre consapevole della tecnologia e la scuola, come agenzia educativa, dovrà provvedere a riformulare gli insegnamenti in una nuova ottica tramite l'uso di questi

nuovi strumenti.

CAPITOLO 2
IL LINGUAGGIO *LOGO*

Nella Geometria della Tartaruga, il computer
è usato in modo completamente differente,
come mezzo matematicamente espressivo,
che ci libera dalla necessità di individuare
argomenti matematici possibili da imparare,
significativi e matematicamente coerenti.
Seymour Papert

2.1 LA NASCITA DEL LINGUAGGIO *LOGO*- Molto è stato detto a proposito del pensiero di Seymour Papert, della sua concezione sull'apprendimento della matematica e di come abbia messo a punto il linguaggio *Logo*, ma non è stata ancora fornita una spiegazione di cosa esso sia e di come sia nato. È necessario, infatti, individuare il contesto dove lo stesso Papert ha vissuto e ha lavorato al fine di comprendere il retroterra culturale in cui tale linguaggio ha trovato terreno fertile per prosperare.

Seymour Aubrey Papert nacque il 29 febbraio 1928 a Pretoria, Sud Africa. Nel 1949 si laureò in filosofia all'Università di Witwatersrand a Johannesburg, per poi conseguire un dottorato in matematica nel 1952 (Schofield, 2016)[3]. Conseguì un secondo dottorato nel 1959 all'Università di Cambridge, ma fondamentale per la costruzione del suo carattere fu il periodo di ricerca presso l'Università di Ginevra

3 https://www.theguardian.com/education/2016/aug/03/seymour-papert-obituary

(Formiconi, 2016).

Durante la sua ricerca sulla matematica e l'educazione dei bambini post-dottorato (Henderson, 2003), tra il 1958 e il 1963, entrò in contatto con Jean Piaget presso il laboratorio di epistemologia genetica e divenne uno dei suoi collaboratori preferiti (Formiconi, 2016). Questo incontro fu fondamentale perché gli permise di entrare in contatto con la pedagogia piagetiana: grazie ad essa acquisì una nuova sensibilità nei confronti dell'insegnamento della matematica. La collaborazione con Jean Piaget, infatti, gli donò la giusta spinta per tentare di rivoluzionare profondamente le tecnologie e i sistemi di apprendimento della matematica (Schofield, 2016)[4].

Papert divenne ricercatore presso il MIT -*Massachusetts Institute of Technology*- mentre la comunità degli scienziati informatici americani era scissa in due fazioni: una riteneva necessario sviluppare *software* intelligenti che promuovessero esercizi pratici che implementassero le funzionalità cognitive, mentre l'altra credeva necessario realizzarli il più possibile simili alla mente umana (Henderson, 2003; Schofield, 2016)[5].

Nel 1967 fondò assieme a Marvin Minsky, pioniere nello studio del funzionamento dei neuroni e dei percettroni che Papert aveva incontrato nel 1960 durante una conferenza di cibernetica a Londra, il celebre MIT CSAIL – *Computer Science and Artificial Intelligence Laboratory*-, acquisendo il titolo di co-direttore (Henderson, 2003).

Nello stesso anno ideò *Logo*, un linguaggio di programmazione ideato al fine di facilitare l'insegnamento della matematica e dei concetti geometrici mediante l'uso del computer: assieme al suo team, inoltre, creò una tartaruga robotica che disegnava quando si muoveva grazie a dei comandi precedentemente inseriti (Formiconi, 2016).

I movimenti della tartaruga erano codificati in linguaggio *Logo*, quindi si

4 https://www.theguardian.com/education/2016/aug/03/seymour-papert-obituary
5 https://www.theguardian.com/education/2016/aug/03/seymour-papert-obituary

dovevano inserire dei comandi di movimento -FORWARD, RIGHT, LEFT, BACK- per farla funzionare: questi codici sono la base della Geometria della Tartaruga (Papert, 1980).

Lo stesso Papert nella sua opera *Mindstorm. Children, Computers and Powerful Ideas* (1980) esprime i suoi intenti per quanto riguarda l'apprendimento con la Tartaruga: si propone di trasmettere nozioni di tipo matematico e geometrico in un modo semplice ed intuitivo, quindi maggiormente assimilabile dai bambini. Non si tratta più di una geometria statica, immobile e lontana dal mondo dei bambini, ma possiede caratteristiche reali e riconducibili alla loro vita quotidiana.

La Tartaruga, inoltre, possiede in sé il concetto di posizione -tipico del punto geometrico- poiché sta ed esiste dove viene messa, ma contiene in sé anche l'idea di direzione, cioè ha un verso in cui può dirigersi: in questo modo risulta essere un ottimo veicolo di "rappresentazione formale" per un bambino poiché può identificarsi con essa. Lo stesso autore afferma che "I bambini si possono *identificare* con la Tartaruga e così sono in grado di traslare la conoscenza che hanno del loro corpo e di come si muovono nell'attività di apprendere la geometria formale" (Papert, 1980; Formiconi, 2016, pagina 26).

Logo non riuscì a prosperare inizialmente perché, a quei tempi, i computer erano poco diffusi e molto costosi, quindi gli adulti non li lasciavano usare ai bambini: la Tartaruga perse visibilità (Henderson, 2003). Nonostante questo, l'invenzione papertiana riuscì a catturare l'attenzione di Kjeld Kirk Kristiansen, proprietario della *Lego*, l'illustre azienda danese di giocattoli, che vedeva di buon occhio l'ideazione di un ambiente di programmazione a misura di bambino (Schofield, 2016) [6].

Dal 1984 Papert iniziò a collaborare con la *Lego* per estendere il linguaggio alle famose costruzioni: riteneva necessario ideare un *set* di robotica che riuscisse a

6 https://www.theguardian.com/education/2016/aug/03/seymour-papert-obituary

concretizzare maggiormente il pensiero astratto dei bambini (Schofield, 2016)[7]. Dopo svariati anni di collaborazione, Papert riuscì a mettere a punto degli "*intelligent bricks*" che non solo potevano essere programmati in linguaggio *Logo*, ma erano anche abbinabili ai mattoncini classici della *Lego* (Schofield, 2016)[8].

In questi giochi, infatti, un sistema *Logo* anima motori, luci e sensori nelle macchine costruite dalla *Lego*: la loro ideazione ha portato alla nascita di molti esperimenti in cui si tenta di utilizzare questi strumenti, permettendo di sfruttare al meglio questo linguaggio (Henderson, 2003). Ebbero un notevole successo grazie anche alla diffusione dei programmi in linguaggio *Logo* in America e in Europa: i tempi erano cambiati, i computer non erano più eccessivamente costosi, quindi questa nuova tecnologia informatica divenne maggiormente accessibile (Henderson, 2003).

La vera rivoluzione si ebbe, tuttavia, negli anni Novanta quando *Logo* divenne un programma installabile tramite *floppy disk* (Formiconi, 2016). Il programma veniva lanciato su schermo nero e dovevano essere inserite le istruzioni in sequenza per far muovere la Tartaruga, tracciando un disegno: questa tipologia di interfaccia ebbe una grandissima risonanza per l'insegnamento della matematica ed è stato declinato in moltissime versioni, da *LibreLogo* alla generalizzazione rappresentata da *Scratch* (Formiconi, 2016).

Negli intenti di Papert la promozione delle nuove tecnologie avrebbe implementato l'apprendimento dei bambini promuovendo la creatività e l'innovazione: in sintesi, ha posto e ha tentato di formulare il problema dello sviluppo del pensiero computazionale, quindi *Logo* è nato per soddisfare questa esigenza.

Nonostante la portata rivoluzionaria di questo strumento, *Logo* non è stato recepito dai docenti e la sua diffusione nelle scuole è rimasta piuttosto ristretta: di fatti l'uso di codici era un'attività che non faceva parte del retroterra culturale degli

7 https://www.theguardian.com/education/2016/aug/03/seymour-papert-obituary
8 https://www.theguardian.com/education/2016/aug/03/seymour-papert-obituary

insegnanti dell'epoca (Formiconi, 2016).

Ad oggi non solo esistono numerose versioni di questa tipologia di codice, ma sono ancora molto utilizzate come si può notare dagli innumerevoli programmi che utilizzano questo linguaggio e dalla promulgazione delle normative europee del FOAM per il periodo 2014-2020: in questo documento si ritrovano obiettivi legati all'implementazione dell'uso del *coding* a scuola. A livello didattico *Logo*, che è un linguaggio di tipo testuale, viene spesso affiancato ai programmi a blocchi similari a *Scratch*, ma le differenze tra queste due tipologie di programmazione verranno affrontate nei paragrafi successivi.

Seymour Aubrey Papert è morto il 31 luglio del 2016, all'età di ottantotto anni, lasciandoci in eredità la sua più grande invenzione: *Logo*, che continua ad essere implementato dalla LCSI (*Logo Computer Systems Inc.*) la quale tuttora sviluppa *software* in questo linguaggio.

2.2 COMANDI E FUNZIONALITÀ- La Geometria della Tartaruga è un modo nuovo di fare la geometria classica, che riconduce al pensiero computazionale per superare gli errori e le difficoltà . Lo stesso Papert afferma che "La geometria della Tartaruga è un modo diverso di fare geometria, come il modo assiomatico di Euclide e il modo analitico di Cartesio sono differenti fra loro. Quello di Euclide è un modo logico. Quello di Cartesio è un modo algebrico. La geometria della Tartaruga è un modo computazionale di fare geometria (Papert, 1980; Formiconi, 2016, pagina 25).[9]"

Il linguaggio *Logo* si basa su pochi e semplici comandi legati alla direzionalità in modo tale da riuscire a far muovere la Tartaruga, la quale, inizialmente, era

9 Le citazioni che seguono sono tratte dalla traduzione dei primi due capitoli di *Mindstorms. Children, Computers and Powerful Ideas* effettuata ad opera di Andreas Robert Formiconi nell'opera *Piccolo manuale di LibreLogo, La Geometria della Tartaruga*

rappresentata da un robot cibernetico che scriveva mentre si muoveva (Formiconi, 2016).

Negli anni Ottanta Papert descrive due tipi di Tartaruga che hanno funzionalità diverse, ma comandi analoghi: la “Tartaruga da pavimento” e la “Tartaruga leggera”. La prima è dotata di ruote e può essere usata per disegnare, l'altra è rappresentata su uno schermo e traccia brillanti linee colorate: come lo stesso Papert afferma (1980) “Nessuna delle due è meglio di un'altra, ma insieme evocano un'idea potente: due entità *fisicamente* diverse possono essere *matematicamente* uguali (o “isomorfiche”).” (Papert, 1980; Formiconi, 2016, pagina 26).

Dopo questo breve *excursus* sulla Tartaruga e la sua nascita, si possono ora illustrare i vari comandi legati al suo movimento. Papert descrive l'avanzare in avanti della Tartaruga tramite il comando FORWARD, il quale dovrà essere accompagnato da un numero che descriva la misura del movimento.

Se invece si vuole far muovere la Tartaruga indietro rispetto alla direzione indicata, bisognerà usare il comando BACK, anch'esso accompagnato da un numero che indichi di quanto deve tornare indietro rispetto al punto di partenza. Questi due comandi esprimono un movimento rettilineo lungo la propria direzione in cui la Tartaruga sta puntando: non si hanno cambiamenti di direzione, ma solo di posizione.

Per quanto concerne la direzionalità si hanno due comandi che permettono alla Tartaruga di cambiare direzione ma che non influiscono sulla posizione: LEFT e RIGHT. Questi comandi indicano entrambi un cambio di direzione a destra o a sinistra e devono essere seguiti da un numero che indichi l'angolo di rotazione espresso in gradi.

La presenza ristretta di numeri e la necessità di apprendere un linguaggio invece che un concetto geometrico, permette al bambino non solo di far leva sulle capacità innate di comprensione dell'espressione verbale, ma anche di promuovere la sua naturale inclinazione ad impartire comandi che, di fatto, vanno a costituire la

programmazione.

Lo stesso Papert afferma che “Per fare disegnare un quadrato alla Tartaruga, si può provare a camminare lungo il contorno di un quadrato immaginario e poi descrivere le operazioni fatte utilizzano la Lingua della Tartaruga. E così facendo, si fa leva sulle capacità motorie dei bambini e sul piacere che provano nel muoversi. È un modo di impiegare la “geometria del corpo” propria del bambino come un punto di partenza per raggiungere la geometria formale (Papert, 1980).”

L'esempio portato da Papert è molto chiaro e riguarda la capacità del bambino di disegnare un quadrato con la Tartaruga: il fanciullo può muoversi per realizzare idealmente la figura per poi descrivere le operazioni fatte, promuovendo la capacità di apprendere tramite il corpo. Di conseguenza il movimento non è lo strumento per apprendere direttamente la geometria formale, ma la riflessione su come il corpo si muove permette di accedere ad un modo nuovo di concepire lo spazio attorno a noi.

In modo quasi spontaneo il bambino apprende prima il quadrato, poi i triangoli, poi i rettangoli per poi domandarsi come si realizzi un cerchio con la tartaruga: questa forma di *problem finding* si associa a quella di *problem solving* e, tramite una serie di errori e correzioni, giunge alla realizzazione del cerchio.

Questo processo è ovviamente veicolato dal docente che non dovrà fornire direttamente la soluzione al problema, ma promuovere una tipologia di riflessione sul proprio movimento per associarlo a quello della Tartaruga -“Quando ti muovi in cerchio tu fai un piccolo passo e poi giri subito un poco. E continui a fare sempre così.” (Papert, 1980; Formiconi, 2016, pagina 29).

Anche se l'insegnante si trovasse di fronte ad un bambino meno esperto e bisognoso di maggiore supporto, “[…] questo aiuto non dovrebbe consistere nella spiegazione di come fare a disegnare il cerchio bensì nell'insistere sul metodo [...] (Papert, 1980; Formiconi, 2016, pagina 29)” perché il fine di tale riflessione dovrà mirare alla costruzione di una connessione tra l'attività e la conoscenza -agire per

capire.

La Tartaruga, inoltre, può apprendere nuovi comandi e nuove parole riutilizzabili che inglobano, senza ripetere, quelle precedenti. Questo meccanismo sta alla base dslla programmazione più complessa. Ad esempio i nuovi comandi possono essere utilizzati per realizzare più volte la stessa figura geometrica ma con dimensioni diverse.

Il bambino può così apprendere principi matematici più generali lavorando per moduli e, contemporaneamente, introducendo il concetto di "stato", entrando in contatto con idee molto importanti, come quella di organizzazione gerarchica, oppure di pianificazione di un progetto, o ancora della nozione di *debugging*.

Nonostante non sia necessario l'uso del computer, Papert afferma che "[...] una volta che i codici per disegnare queste figure sono stati predisposti, questi diventano mattoni per costruire altre cose che consentono di creare nel bambino gerarchie di conoscenza. In tale processo si sviluppano capacità intellettuali importanti – un aspetto che si palesa guardando alcuni progetti che i bambini intraprendono spontaneamente dopo alcune sedute di lavoro con la tartaruga. (Papert, 1980, Formiconi, 2016, pagina 30)".

Alcuni bambini iniziano poi a sviluppare progetti e disegni che trascendano dalla mera matematica del quadrato e del triangolo per poter realizzare opere più complesse come, ad esempio, una casa.

Tutti questi tentativi e successi passano sempre dall'errore che, nell'istruzione formale, verrebbe vissuto in accezione negativa, ma in un ambiente come è *Logo*, viene accettato. Questa capacità fuori dal comune di poter reagire in maniera tanto positiva all'errore nasce dal naturale processo di *debugging* in cui i bambini verificano la buona riuscita e comprensione del codice da loro descritto. Questo processo nasce dalla natura stessa della programmazione, in cui si studia il difetto e, nella Geometria della Tartaruga, il premio è dato dalla buona riuscita del codice e

della risoluzione dell'errore.

Questa correzione e superamento dell'errore permette di far comprendere al bambino che "[...] le cose non sono mai del tutto giuste o del tutto sbagliate, piuttosto si aggiustano in modo continuo (Papert, 1980; Formiconi, 2016, pagina 32)". Questa acquisizione risulta essere utile non solo nel campo delle discipline puramente scolastiche, ma si riflette anche nella capacità di vivere ed essere parte integrante del mondo.

Papert individua, infatti, due aspetti della Geometria della Tartaruga: uno matematico che si riferisce al mondo dei numeri in sé e per sé e l'altro matetico (Papert, 1980), cioè che si riferisce all'apprendimento. Secondo Papert la Tartaruga strutta la sintonicità con la quale si apprende una nozione astratta attraverso la corrispondenza con l'attività motoria.

"La geometria della Tartaruga si impara bene perché è sintonica. E questo aiuta anche nell'apprendimento di altre cose perché incoraggia l'uso consapevole e deliberato di strategie matematiche di *problem solving*. (Papert, 1980; Formiconi, 2016, pagina 33)". Lo stesso autore è consapevole dell'uso di strategie di *problem solving* all'interno dell'apprendimento sintonico di cui parla e, anzi, promuove il suo incoraggiamento perché l'uso di queste strategie permette una migliore comprensione.

L'uso della Tartaruga, inoltre, è in accordo anche con i suggerimenti introdotti da Polya nella risoluzione dei problemi, cioè di porsi domande euristiche che, normalmente, non verrebbero poste, ma che nascano spontaneamente al momento dell'utilizzo del linguaggio *Logo*. I bambini, infatti, si interrogano costantemente su come possano riuscire ad ottenere i risultati sperati all'interno di un determinato processo, ma tali domande non vengono direttamente poste dal docente, ma nascono spontaneamente dall'incontro con il problema stesso.

Questo processo di porsi domande euristiche non si interrompe nel momento in

cui si esce da *Logo* poiché si estendono e diventano punto di partenza per situazioni problematiche future. In definitiva, Papert segue i consigli di Polya e li concretizza, superando la componente astratta in cui erano stati formulati. I bambini, inoltre, quando apprendono cercano qualcosa di simile a ciò che già conoscono, quindi le esperienze vissute in ambiente *Logo* riescono a far maturare una competenza che potrà essere applicata ai principi della matematica scolastica, trasferendo la conoscenza da un contesto all'altro.

I consigli di Polya, che non sono facilmente applicabili alla matematica scolastica poiché richiedono competenze di non facile comprensione per i bambini, grazie alla pratica avuta nella Geometria della Tartaruga trovano facile riscontrabilità, anche se i concetti non sono sempre molto evidenti. Nonostante ciò il bambino tende ad arrivare intuitivamente al concetto, quindi lo acquisisce anche senza esserne completamente consapevole.

Il fine che Papert vuol promuovere è un concetto che prescinde dalla matematica scolastica: si tratta della "protomatematica", che è un insieme di capacità necessarie alla reale comprensione delle nozioni matematiche e che, solitamente, non vengono acquisite.

La Geometria della Tartaruga, inoltre, ha basi non solo matematiche, ma anche relazionali ed affettive poiché "molti bambini sono venuti nel laboratorio LOGO odiando i numeri quali oggetti alieni e se ne sono andati amandoli (Papert, 1980; Formiconi, 2016, pagina 36)".

I bambini, tramite l'uso del linguaggio *Logo*, possono accedere ai contenuti matematici senza dover necessariamente passare dalle nozioni meramente scolastiche, ma basando le loro acquisizioni sull'intuizione e sull'applicazione di principi in un ambiente protetto che non vede l'errore in un'accezione negativa.

La Geometria della Tartaruga, talvolta, può generare modelli intuitivi di concetti matematici complessi che i bambini capiscono con difficoltà, come può

essere l'acquisizione del concetto di misurazione degli angoli: tramite *Logo*, intuitivamente comprendono come si misuri in gradi ed usano questa loro scoperta all'interno della programmazione, giungendo alle scuole medie inferiori con una consapevolezza maggiore rispetto ai loro compagni.

Un altro concetto matematico veicolato dal linguaggio *Logo* è il concetto di variabile, cioè usare simboli per nominare entità sconosciute: come in matematica, anche nella Geometria della Tartaruga vengono utilizzate per facilitare la memorizzazione e l'applicazione di comandi. L'applicazione delle variabili permette al fanciullo di accedere ad un un aspetto che è fondamentale in matematica e che prelude al calcolo simbolico.

Questo modo di insegnare la matematica permette di far sviluppare nei bambini un "potere matematico" che molto raramente si riesce ad acquisire senza il linguaggio *Logo*, che conferisce la possibilità di sperimentare nozioni in ambienti nuovi e lontani dalla vecchi cultura e didattica, stimolando le metodologie di *problem solving*.

L'uso del concetto di angolo o di variabile permette di sviluppare nuovi modi di parlare di matematica poiché il bambino, in accordo con le idee di Poyla, ricerca nella sua esperienza situazioni già vissute e simili a quelle che gli si presentano: giocando con la Tartaruga egli acquisisce concetti che potrà sfruttare in altri contesti.

Tramite alcune acquisizione si accede anche ad altri concetti che permettono al fanciullo di creare opere più complesse, sfruttando quelle che sono definite da Papert "idee potenti", cioè capacità di *problem solving*, di pensiero computazionale e delle acquisizioni matematiche.

Lo sviluppo di queste capacità permettono nuovi livelli di consapevolezza nel bambino, quindi "Si impara a apprezzare e rispettare la forza delle idee potenti. Si impara che l'idea più potente di tutte è l'idea di idea potente (Papert, 1980; Formiconi, 2016, pagina 42)."

2.3 *TURTLEART* E *SCRATCH*- Il linguaggio *Logo* è uno strumento introduttivo alla programmazione vera e propria che si trova in numerosi *software*, declinandosi in numerose versioni e prodotti. Dalla lettura del testo *Piccolo manuale di LibreLogo. La Geometria della Tartaruga* di Andreas Robert Formiconi (2016) è possibile tracciare un rapido e breve sguardo sulle differenze riscontrabili tra le varie tipologie di prodotti che popolano il mondo informatico.

È noto che esistono varie tipologie di programmi all'interno del cyberspazio, in particolare sici sono i "prodotti proprietari", i *software* liberi (*free software*) e i *software open source*. Per comprendere le dinamiche che si instaurano tra i vari produttori di programmi, è necessario parlare delle distinzioni che esistono tra queste tipologie di *software*.

Un "prodotto proprietario" è destinato alla vendita, quindi l'azienda produttrice non distribuisce il codice sorgente in chiaro, avvalendosi del diritto della proprietà intellettuale. In poche parole, è necessario acquistare il prodotto per usufruire delle sue funzionalità e, inoltre, si ritrovano delle restrizioni per quanto riguarda il suo uso e la sua diffusione.

Un *software* "libero", invece, è una tipologia di prodotto con il quale non solo la persona o azienda produttrice distribuisce il codice sorgente in chiaro, ma si rispettano anche quattro tipi di libertà che permettono alla comunità informatica di arricchirsi grazie al suo uso e sviluppo (Formiconi, 2016):

- Libertà di eseguire il programma come si desidera e per qualsiasi scopo;
- Libertà di studiare come funziona il programma e di modificarlo per poterlo adattare alle proprie necessità;
- Libertà di ridistribuire copie in modo da aiutare il prossimo;
- Libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti eventualmente apportati, cosicché tutta la comunità ne tragga beneficio.

Molto spesso si identifica il *software* libero quello *open source*, ma in realtà sono due cose differenti ed occorre distinguerle in modo adeguato. Innanzitutto con il *software open source* si distribuisce in chiaro il codice sorgente, ma non si fa assolutamente menzione delle quattro libertà sopra descritte, perdendo la connotazione etica tipica del *free software*. Di conseguenza chi produce il *software open source* ha scopi di lucro e le aziende che aderiscono a questo tipo di diffusione del programma lo fanno perché ritengono questa strategia adeguata alla loro tipologia di *marketing*.

Partendo da questa distinzione iniziamo ad analizzare i prodotti che hanno trasformato il linguaggio *Logo* modificandone, tuttavia, la componente testuale: il più noto è senza ombra di dubbio *Scratch*. L'ideatore di questo *software* è Mitchel Resnick, allievo di Papert, che tuttora opera nel *Media Laboratory* del MIT. A differenza di molti altri programmi in linguaggio *Logo*, *Scratch* non solo permette la produzione grafica, ma consente anche di realizzare animazioni e videogiochi, aprendo le porte alla sperimentazione di tecniche di programmazione piuttosto sofisticate.

Strutturato come servizio *web*, nel giro di pochi anni ha generato una grande comunità dove si diffondono e si scambiano programmi: si tratta del sito http://scratch.mit.edu nel quale sono raccolti più di 14 milioni di progetti realizzati dagli utenti di *Scratch* (Aivaloglou e Hermans, 2016). Come detto precedentemente, al suo interno integra il linguaggio *Logo*, ma non si basa su di esso per il semplice fatto che *Scratch* si avvale non di un codice scritto per procedere, ma di uno visuale: i comandi sono strutturati in blocchi colorati che si possono incastrare tra di loro come se fosse un grande puzzle (Formiconi, 2016).

L'idea di una programmazione a blocchi risulta essere molto attrattiva poiché la presenza dei mattoncini rende semplice l'uso di questo programma e, inoltre, riduce il grado di errore: non si possono costruire sequenze corrette a meno che non si si

incastrino correttamente le forme del codice. La riduzione di errori ortografici e sintattici permette un alto grado di successo almeno nelle fasi iniziali.

Dall'ideazione del codice a blocchi sono nati molti altri linguaggi che permettono un approccio grafico simile a quello di *Scratch* -basta pensare a *Snap!*, *Alice*, *Blockly*, *Andorid App Inventor*-, ma per questo progetto è stato scelto un *software* meno noto e che si avvicina maggiormente a *Logo*: si tratta di *TurtleArt*.

TurtleArt è un *software open source* che sfrutta il linguaggio *Logo* per poter creare disegni in 2D grazie all'uso di blocchi logici facilmente utilizzabili: lo stesso sito di riferimento[10] dichiara che questa scelta è stata presa al fine di esemplificare l'uso del programma per i bambini. Come *Snap!*, *TurtleArt* è molto simile a *Logo* ma rispetto ad esso ha alcune limitazioni. A differenza di Scratch non consente di creare animazioni, in compenso esporta i programmi in codice Logo.

LibreLogo è un *plugin*, componente aggiuntivo, di *LibreOffice*, *suite* di applicativi per produttività personale molto simile a *MicrosoftOffice*, ma completamente gratuito perché si tratta di un *free software*. La possibilità di utilizzare *Logo* all'interno di un *word processor* è molto interessante poiché permette di creare immagini integrate nel documento tramite la scrittura dei comandi, senza che vengano importate.

La creazione di questo *plugin* si deve all'ungherese Németh László, che non solo ha integrato in *LibreOffice Logo*, ma lo ha implementato grazie alla scrittura del programma con il linguaggio *Pyhton*, il quale è adatto all'integrazione con molti altri tipi di *software* e linguaggi. Grazie a questo *plugin* si può ottenere un prodotto grafico nel solito *file* e nello stesso formato in cui viene scritto, cioè in ODT, quello standard di *LibreOffice*.

Da questa analisi, quindi, emerge l'esistenza di due grandi famiglie di programmi in linguaggio *Logo*, cioè quelle strutturate a blocchi e quelle basate sulla

10 https://turtleart.org/

scrittura del testo. Questa suddivisione fra gli educatori, ovvero fra coloro che ritengono più utile l'uso di un codice visuale e quelli che privilegiano quello testuale: da questa diatriba sono nate numerose ricerche e sperimentazioni per decretare quale di questi due linguaggi sia il più adatto in campo educativo .

Una sperimentazione ben articolata a riguardo è stata condotta da Weintrop e Wilensky nel 2015, dalla quale hanno realizzato due diversi articoli, uno dove si valutano oggettivamente le differenze di comprensione del codice con i sue metodi, l'altro che analizza invece le percezioni soggettive degli studenti a riguardo. L'analisi parte dallo studio delle valutazioni e delle opinioni di 90 studenti di scuola secondaria che hanno frequentato un corso di 10 settimane, nel quale si creava l'opportunità di sperimentare sia l'uso di programmi a blocchi sia quelli testuali.

Il primo articolo di Weintrop e Wilensky (2015) sofferma la sua attenzione su come vi sia una maggiore comprensione di alcuni costrutti *software* specifici da parte degli studenti che utilizzano programmi a blocchi. Dall'altro canto i risultati non permettono di constatare se gli alunni siano poi in grado di comprendere la logica di un algoritmo. Gli autori riconoscono la necessità di approfondire ulteriormente la questione.

Nel secondo articolo si sottolinea, invece, il punto di vista dello studente partendo da tre domande fondamentali (Weintrop e Wilensky, 2015A, pagina):

- Gli studenti ritengono che la programmazione a blocchi sia più semplice di quella testuale e, se pensano questo, perché?
- Quali sono le differenze che gli studenti percepiscono tra la programmazione a blocchi e quella testuale?
- Quali sono i principali svantaggi che vengono percepiti dagli studenti nella programmazione a blocchi?

Le evidenze mostrano come generalmente, e soprattutto nelle fasi iniziali, venga preferita la programmazione a blocchi poiché permette di manipolare

facilmente gli oggetti: i comandi sono sempre visibili ed individuabili, cosa che non avviene con l'uso di *software* testuali, che risultano essere difficili da utilizzare. Un dato interessante, tuttavia, è che gli studenti, via via che progrediscono nel lavoro, individuano in modo piuttosto preciso alcuni svantaggi nella programmazione a blocchi, mostrando una nitida consapevolezza dello strumento e sottolineando criticità e potenzialità.

La prima sensazione esternata dagli studenti riguarda la potenzialità della programmazione testuale, poiché con quest'ultima si ha la possibilità di realizzare codici e funzioni maggiormente complesse. I blocchi, quindi, sono percepiti e considerati meno potenti del codice testuale poiché le funzioni sono limitate e non permettono alcune ricorsività tipiche dei programmi *text-based*.

Un altro svantaggio emerge dalla lentezza con cui si programma con i blocchi rispetto a quello testuale: se si vogliono realizzare codici più complessi si hanno maggiori complicazioni e difficoltà con i *block-based software* rispetto a quelli *text-based*. I blocchi, di conseguenza, vengono percepiti come una limitazione nel momento in cui si vuole fare un salto di qualità nella programmazione poiché diventano più complessi da gestire e limitano le possibilità di creazione.

Gli studenti, infine, hanno sottolineato come il linguaggio a blocchi non sembra “abbastanza vero”, a conferma del fatto che ai giovani piace fare le cose “per davvero”.

In un altro studio (Lewis et al, 2010) viene messo direttamente a confronto *Scratch* con *Logo*, rilevando delle interessanti conclusioni: l'esperienza riguarda studenti di 10-12 anni che hanno partecipato ad un corso estivo di programmazione, articolato in 36 ore in 12 giorni. Gli studenti sono stati suddivisi in due gruppi, ciascuno dei quali è stato trattato sia con *Scratch* sia con *Logo*. Un gruppo (*Scratch-First*) è stato iniziato alla programmazione con *Scratch* per poi passare a *Logo*, mentre l'altro (*Logo-First*) ha subito il processo opposto.

L'autrice mette alla prova le seguenti ipotesi sui vantaggi di Scratch:

1. Maggiore facilità nella programmazione.

2. . Maggiore confidenza nelle competenze acquisite e maggiore propensione a continuare lo studio della programmazione.

3. Maggiore capacità di comprensione di costrutti specifici quali i cicli e le istruzioni condizionali.

Il risultato imprevisto, per il quale questo lavoro è ampiamente citato, consiste nel fatto che il livello di confidenza nelle proprie capacità di programmazione è risultato significativamente superiore negli studenti che avevano seguito il percorso con Logo.

Quindi, secondo questo studio, l'uso di *Logo* mostra segni di supporto all'attività di sviluppo della confidenza nella materia e della comprensione delle funzionalità cicliche del programma, stimolando la curiosità verso la programmazione.

Altri studi si sono concentrati sulla bontà dei prodotti della comunità di *Scratch* (Aivaloglou e Hermans, 2016A), in particolare per quanto riguarda i *code smells*, cioè i frammenti di "codice maleodorante" (Formiconi, 2016). L'espressione riguarda il codice di scarsa qualità, ovvero quello, ad esempio, presenta sezioni troppo lunghe, clonate inutilmente o non utilizzate; caratteristiche che manifestano una limitata comprensione dei metodi di programmazione: la produzione di sequenze non eseguite o non funzionanti non servono alla creazione del prodotto desiderato e aumentano il rischio di introdurre errori.

La ricerca di Aivaloglou e Hermans serve per chiarire se i linguaggi a blocchi incorrano maggiormente nel rischio di produrre codice obsoleto: per verificare questa tesi sono stati analizzati l'influenza dei *code smells* in un gruppo di 61 scolari di età compresa tra i 12 e i 14 anni. I ragazzi sono stati suddivisi in tre gruppi per verificare come reagivano con tre tipologie differenti di codice: pulito, cioè privo di ridondanze

o errori, eccessivamente lungo e, infine, quello duplicato indebitamente.

Ciascun gruppo ha lavorato con una tipologia ben precisa di codice e, nonostante la poca significatività dello studio causata dal basso campione raccolto, emerge chiaramente come gli studenti che si sono trovati di fronte a codici indebitamente lunghi o clonati abbiano riscontrato maggiori difficoltà di comprensione.

Dalle analisi emerge che il gruppo che ha lavorato con codice eccessivamente lungo ha trovato difficoltà nel capire come lo *script* funzioni, mentre quello che ha trovato forme di *code* clonato indebitamente o inutilmente ha faticato ad individuare l'errore e a correggerlo poiché risulta essere troppo complesso modificare il codice per ottenere i risultati sperati.

In un altro studio condotto dagli stessi autori nello stesso anno, *How Kids Code and How We Know: An Exploratory Study on the Scratch Repository* (2016), viene analizzato un più vasto campione di lavori al fine di definire le caratteristiche di *Scratch* e dell'effettiva bontà del codice. In questo studio vengono analizzati 250˙000 lavori direttamente dal sito http://scratch.mit.edu, che conta 14 milioni di progetti al suo interno, quindi circa un 2%del totale.

La vastità del campione e il rigore con cui è stata svolta l'indagine permette di avere dei risultati significativi: gli autori, quindi, hanno impostato il loro lavoro alla ricerca di *scripts* eccessivamente lunghi, duplicati o inutilizzati -i *code smells*- che sono indice di cattiva qualità del codice. Poiché questa analisi presenta un così vasto campione, la bontà dell'uso di programmi a blocchi nell'apprendimento sarebbe dimostrata se i dati riguardanti la presenza dei *code smells* risultasse essere poco significativa.

Dai risultati ottenuti, in prima battuta emerge che la maggioranza dei programmi analizzati sono molto piccoli, quindi i loro autori non hanno realizzato codici articolati al fine di creare prodotti eccessivamente complessi. Questo fatto non

depone a favore dei fautori dell'uso dei programmi a blocchi poiché si sottolinea come la maggior parte dei fruitori di *Scratch* si attestino in una fascia di utenti di base e non avanzati, per cui consegue un modesto livello di astrazione.

I risultati analizzati, inoltre, hanno evidenziato una significativa presenza di *code smells* tra i progetti realizzati dagli utenti di *Scratch*: questo fatto genera numerose perplessità e spunti di riflessione sull'effettiva utilità dei linguaggi visuali rispetto a quelli testuali. La significativa presenza di codici ripetuti ed obsoleti non depone a favore della programmazione a blocchi perché sottolinea come vi sia una difficoltà di fondo nell'utenza di tale tipologia di programmi.

Ai fini del progetto da me ideato per questa Tesi di Laurea e dalle varie considerazioni scaturite dall'analisi dei risultati delle sperimentazioni, ricerche e studi precedentemente citati, ho ritenuto opportuno utilizzare un programma a blocchi soprattutto per scelte, sostanzialmente, didattiche.

In accordo con la didattica del CLIL, infatti, è sconsigliabile attuare una pratica di scrittura in lingua straniera almeno fino alla terza primaria: nelle prime due classi si dovrebbe prediligere un approccio puramente orale. In accordo con questa metodologia ho scelto la programmazione a blocchi, in modo tale da acconsentire ad una prima acquisizione orale dei comandi di direzionalità in lingua inglese, senza però perdere l'occasione di aprire la strada ad un futuro uso di programmi *text-based.*

La classe in cui è stata condotta questa sperimentazione, inoltre, è composta da ventiquattro alunni ed accoglie al suo interno cinque alunni BES, di cui uno certificato con Legge Quadro 104/1992 e quattro sospetti DSA in attesa di certificazione con Legge 170/2010, quindi è risultato sin da subito poco ottimale far partire il progetto con un approccio testuale in un contesto così variegato: il rischio sarebbe stato quello di lasciare indietro i bambini più fragili.

Gli alunni con DSA, solitamente, hanno numerose difficoltà nell'acquisizione della lingua straniera, specialmente se scritta (Lo Sapio, 2011), per questo è

preferibile mantenere un approccio orale o, come in questo caso, è bene fornire le parole già scritte in inglese. Questa pratica argina l'errore, limitando la frustrazione dell'alunno, implementando la sua autostima e fiducia nelle sue capacità e riducendo, di fatto, il rischio di un possibile rifiuto di eseguire il compito richiesto.

L'uso dei programmi a blocchi, quindi, è sembrata essere la scelta obbligata al fine di realizzare un progetto inclusivo per tutte le varie tipologie di alunno presenti nel gruppo-classe, nonostante vi fosse il rischio dei sopracitati *code smells* (Aivaloglou e Hermans, 2016) e non fosse il più auspicabile per un'acquisizione profonda della conoscenza dei vari comandi (Lewis et al, 2010).

Nonostante la scelta da me effettuata per questo anno scolastico, ritengo che questo progetto dovrebbe proseguire in un'ottica di verticalità per la terza primaria: in accordo con il CLIL, a partire da questa classe sarà possibile introdurre la programmazione testuale. Nelle sezione di conclusioni di questa tesi riporterò un possibile sviluppo per la classe successiva, in un'ottica di continuità e di implementazione del *coding* verso forme più complesse e più matematiche.

CAPITOLO 3: PROGETTO DIDATTICO PER UNA SECONDA PRIMARIA

Voglio vedere bambini che programmano
i computer e non viceversa!
Seymour Papert

3.1 STRUTTURA DEL PROGETTO- Durante l'ultima annualità del Corso di Laurea in Scienze della Formazione Primaria, il professor Andreas Robert Formiconi ha effettuato delle video lezioni per il Laboratorio di Tecnologie Didattiche, nel quale ha presentato un *software* piuttosto interessante: *LibreLogo*. Questo programma permette di sfruttare il linguaggio *Logo* per effettuare una programmazione *text-based*, attraverso la quale è possibile avere una prima esperienza di *coding*. Questo strumento, inoltre, permette di creare figure geometriche o immagini articolate tramite dei semplici comandi di direzionalità: queste lezioni sono state fondamentali per la scelta dell'argomento per la Tesi di Laurea.

Il Corso di Laurea in Scienze della Formazione Primaria, inoltre, prevede per l'ultima annualità di tirocinio duecentodieci ore di pratica diretta, di cui ottanta alla scuola d'infanzia e centotrenta a primaria, durante le quali è necessario produrre un video per il progetto MARC (Modellamento, Azione, Riflessione, Condivisione), nel quale le tirocinanti devono riprendere una loro attività didattica svolta nella classe accogliente.

Per soddisfare al meglio queste richieste è stato creato un prodotto organico che unisse il tirocinio diretto, il progetto MARC e la Tesi di Laurea, in modo tale da concludere il percorso di Studi con una riflessione generale che coinvolgesse tutti gli

aspetti vissuti durante i cinque anni di Corso.

Prima di iniziare ad illustrare il progetto è necessario descrivere il contesto entro cui è stato inserito, sia a livello macroscopico, ovvero di Istituto, sia a livello microscopico, cioè di classe. Lo studio del contesto è fondamentale per un'azione efficace, altrimenti potrebbe insorgere il rischio di progettare ed attuare attività interessanti, ma non direttamente assimilabili dal gruppo-classe (Calvani, 2007).

Durante l'ultima annualità del Corso di Studi, il tirocinio diretto si è svolto presso l'Istituto Comprensivo Nord di Prato, nel plesso "Gian Paolo Meucci", situato in via Giovanni Marradi n° 2, zona Santa Lucia. Il quartiere in cui si inserisce il plesso è di tipo prettamente residenziale ed è situato esattamente nella strada parallela al viale Galileo Galilei, una delle principali arterie della città.

Il plesso è composto da dodici classi, ognuna delle quali è dotata di un computer portatile e di una LIM: questa fornitura tecnologica è stata acquistata grazie ai fondi ricevuti dal PON 2014-2020 per il progetto *Atelier Creativi*, che ha permesso l'acquisto di sei *BeeBot*, ovvero delle piccole api robotiche programmabili con alcuni pulsanti sul dorso che rimandano al linguaggio *Logo*, comprensive di tabelloni e percorsi forniti dalla casa produttrice, e sono state ampiamente utilizzate nel progetto. All'interno del plesso, inoltre, è stato allestito un *Atelier Digitale*, una grande aula che ospita un ambiente adatto al lavoro di tipo laboratoriale, in cui si trovano undici *laptop*, sei computer portatili, otto fissi e un *active table*: questo spazio è risultato essere quello ideale per il progetto.

La classe in cui è stato svolto il progetto è composta da ventiquattro alunni, di cui cinque identificati come BES (Bisogni Educativi Speciali): uno di questi è certificato con Legge-quadro 104/1992 poiché è cardiopatico, ha problemi motori legati alla parte destra del corpo con una conseguente lateralizzazione a sinistra ed è colpito da medio ritardo cognitivo. I restanti alunni BES sono in attesa di certificazione come DSA (Disturbo Specifico dell'Apprendimento), quindi per loro è

stato sviluppato un PDP (Piano Didattico Personalizzato) in cui vengono attuate pratiche compensative e dispensative all'interno della didattica di classe.

All'interno del gruppo-classe non sono presenti particolari difficoltà di integrazione o di collaborazione tra gli alunni poiché la politica adottata dalle titolari di cattedra è improntata all'inclusione di tutti i bambini. Nessun bambino, infatti, svolge attività al di fuori dal gruppo classe: in particolare l'alunno certificato con legge 104/1992 resta sempre con i suoi compagni e non viene isolato come può talvolta succedere in altre realtà scolastiche.

Tutti i bambini, inoltre, sono lasciati liberi di giocare ed interagire tra di loro, quindi nessuno è obbligato a formare gruppi-gioco prestabiliti, come a volte può succedere in altre realtà scolastiche in cui si argina in questo modo il problema del bambino con *deficit*. Nei precedenti anni di tirocinio, infatti, è stato possibile assistere all'applicazione di questa strategia: in una classe avevano messo a punto questo metodo per non isolare un alunno certificato con Legge-quadro 104/1992 con grave ritardo cognitivo, per cui alcuni bambini, a rotazione, dovevano necessariamente giocare con lui per non lasciarlo mai solo.

Quest'ultima imposizione è discutibile poiché non è inclusiva (Lo Sapio, 2011), per cui la scelta adottata dalle docenti di classe ha portato a favorire la spontaneità del gioco dei bambini, incoraggiando l'alunno con difficoltà a costruire autonomamente le relazioni. In questo modo non solo il bambino con *deficit* non è mai solo o isolato e riesce sempre ad interagire con tutti, ma si è sviluppata nei suoi compagni una buona capacità relazionale. Il contesto da cui parte il progetto è, quindi, piuttosto variegato, ma sfruttabile per costruire gruppi di lavoro facendo leva sulle buone capacità relazionali che questi bambini possiedono.

Sulla base delle Indicazioni Nazionali per il curricolo alla scuola dell'infanzia e primaria, le indicazioni della Comunità Europea e la legge 107/2015, è stato ideato un progetto che coinvolgesse più discipline al suo interno, quindi più obiettivi e

traguardi di apprendimento. Partendo dalla geografia e l'educazione fisica, i bambini esploreranno la matematica e la geometria attraverso la tecnologia e l'inglese per giungere a toccare alcuni obiettivi di arte.

In accordo, inoltre, con il FOAM 2014-2020, declinato nella legge 107/2015, detta della "buona scuola", sono stati stanziati fondi per progetti in materia dell'uso delle nuove tecnologie e della programmazione. Questa nuova attenzione ha portato l'Istituto a promuovere progetti in questo campo, istituendo un "Team Digitale" composto da docenti che hanno ricevuto una formazione specifica nell'uso di nuovi mezzi, incluse le *BeeBot*.

È evidente il collegamento esistente tra la recente normativa e le Indicazioni Nazionali per il Curricolo del 2012, in particolare per quanto riguarda le discipline di tecnologia e matematica. All'interno delle Indicazioni Nazionali, per quanto riguarda la sezione di tecnologia, si trova il suggerimento di introdurre alcuni linguaggi di programmazione particolarmente semplici e versatili, che possano permettere lo sviluppo sia del gusto per l'ideazione e la realizzazione di progetti come siti *web* interattivi, esercizi, giochi, programmi di utilità, sia della comprensione del rapporto tra codice sorgente e risultato visibile.

Queste indicazioni sono fondamentali poiché sono direttamente riconducibili al *coding* e al linguaggio *Logo*, specialmente per quanto concerne l'uso di *software* di programmazione per realizzare prodotti che illustrino il rapporto tra codice e risultato ottenuto. Grazie a *Logo* si possono produrre semplici rappresentazioni grafiche utilizzando strumenti multimediali: questo traguardo è anche un obiettivo di apprendimento delle Indicazioni Nazionali, non solo per la sezione di tecnologia, ma anche di arte, motivo per cui tale strumento permette di creare una coerenza e una struttura interna più forte.

Nelle Indicazioni Nazionali per il Curricolo all'interno della disciplina della matematica si ritrovano suggerimenti anche per la costruzione del pensiero

matematico, un tipo di acquisizione fondamentale per i futuri apprendimenti. Questo processo non può risolversi in un'acquisizione immediata, ma fa parte di un percorso lungo e progressivo nel quale si intrecciano vari concetti, abilità, competenze e atteggiamenti, che si consolidano e si sviluppano nel tempo a più riprese.

Grazie al *coding* si veicola un nuovo modo di pensare tramite il pensiero computazionale, quindi quest'ultimo prende parte al processo di formazione del pensiero matematico. Il pensiero computazionale arricchisce il pensiero matematico grazie a delle nuove capacità: si sviluppa un modo diverso di pensare e risolvere i problemi, permettendo l'accesso a nuove competenze ed abilità fondamentali.

Lo strumento principe per veicolare queste abilità è, in accordo con Papert (1980), l'uso del computer, che pone il bambino in diretto contatto con la necessità di risolvere costantemente problemi per portare a termine le richieste, in un'ottica di interrogazione euristica non consapevole. In questo modo i bambini tendono a sviluppare capacità naturali di *problem solving* e *problem finding*, che sono fondamentali per gli apprendimenti futuri, sia perché sono alla base della risoluzione di problemi matematici, sia perché permettono di accedere a livelli di conoscenza più profondi grazie alla spontaneità delle domande nate durante il processo di programmazione.

Grazie all'insegnamento del *coding*, infatti, vengono promosse nuove capacità legate alla risoluzione di problemi, tipiche della pratica matematica e, all'interno delle Indicazioni, si ritrova la necessità di collocare quest'ultimi in contesti e situazioni autentiche, in modo tale da evitare esercizi ripetitivi e, soprattutto, prevedibili. Nelle Indicazioni Nazionali si ritiene che, grazie alla stimolazione dell'insegnante, l'alunno possa imparare a superare le situazioni problematiche, per cui è un obiettivo che si lega strettamente alle finalità di questo progetto.

Oltre alle competenze scaturite direttamente dal linguaggio della programmazione, si possono rintracciare nelle Indicazioni Nazionali altri obiettivi di

apprendimento veicolati dall'uso di *Logo*. Questo linguaggio sfrutta gli indicatori topologici per far muovere la *BeeBot*, quindi è possibile rintracciare come obiettivo di apprendimento l'esecuzione di un semplice percorso partendo dalla descrizione verbale o da un disegno. I percorsi possono essere descritti a voce, ma anche rappresentati graficamente in modo da poter ricordare cosa si sia fatto o dare istruzioni a qualcuno.

All'interno della normativa si ritrova come obiettivo di apprendimento anche la capacità di comunicare la posizione di oggetti nello spazio fisico usando termini di direzionalità adeguati. In questo modo si può legare l'uso del *coding* ad un ulteriore obiettivo di apprendimento descritto nelle Indicazioni Nazionali: la percezione della propria posizione nello spazio. Questa acquisizione è fondamentale non solo per la matematica, riconducendo ai concetti geometrici, ma anche per la geografia poiché è il primo passo per lo sviluppo dell'orientamento del bambino.

Ai fini del progetto, inoltre, sono stati rintracciati nella normativa anche ulteriori obiettivi di apprendimento legati alla geometria, cioè il riconoscimento, la denominazione e la descrizione di alcune figure geometriche, in particolare il quadrato e il rettangolo. Di queste figure l'alunno dovrà descrivere il numero di lati e distinguerle le une dalle altre, ovvero riconoscere che il quadrato ha quattro lati uguali e che il rettangolo, invece, ne ha a due a due congruenti.

In accordo con le recenti normative in materia di insegnamento della lingua straniera, questo progetto promuove un approccio CLIL alla didattica. Il termine CLIL, introdotto da David Marsh e Anne Maljers nel 1994, è l'acronimo di *Content and Language Integrated Learning*, apprendimento integrato di contenuti disciplinari in lingua straniera veicolare, quindi promuove l'insegnamento dell'inglese tramite le discipline tradizionali evitando un approccio di tipo prettamente grammaticale alla lingua straniera.

Questo metodo si basa sul fatto che la lingua straniera venga appresa non

grazie allo studio sistematico della sua grammatica, ma dalla necessità di dover comunicare. Veicolando l'insegnamento delle discipline attraverso la lingua straniera, i bambini possono apprendere strutture, forme lessicali e nozioni che andranno a costruire, nel lungo termine, apprendimenti più profondi e ben inseriti nel loro contesto.

Questa metodologia si sta diffondendo in maniera capillare in Europa, come testimoniano il Rapporto *Eurydice Keydata on Languages at school in Europe* (2012) e la Raccomandazione della Commissione Europea *Rethinking Education* (2012), nei quali la competenza linguistica è definita una dimensione chiave per la modernizzazione dei sistemi di istruzione europei, quindi la metodologia CLIL viene vista come il motore del rinnovamento e del miglioramento dei curricoli scolastici[11].

Le Indicazioni Nazionali per il Curricolo per l'insegnamento dell'inglese, inoltre, suggeriscono l'uso di questo approccio, proponendo, infatti, di creare situazioni in cui la lingua straniera sia utilizzata al posto dell'italiano per veicolare apprendimenti collegati ad ambiti disciplinari diversi.

Nel testo delle Indicazioni Nazionali, inoltre, si ritrova tra gli obiettivi di apprendimento da acquisire al termine della classe terza la capacità di svolgere i compiti assegnati secondo i comandi dati in lingua straniera. Ai fini del progetto, infatti, si promuove questa capacità poiché è necessario conoscere, per l'uso delle *BeeBot* prima e del computer dopo, i comandi di direzionalità in lingua inglese. Di conseguenza è promosso anche l'obiettivo di comprendere vocaboli ed istruzioni pronunciati chiaramente e lentamente: i bambini dovranno, infatti, capire i comandi in lingua inglese ed eseguirli per poter portare a termine il compito.

Per quanto riguarda la disciplina della geografia, in questo progetto sono stati individuati degli obiettivi di apprendimento declinati dalle Indicazioni Nazionali relativi a questa disciplina. Considerando che per le prime classi della scuola

11 http://www.indire.it/progetto/clil-content-and-language-integrated-learning/

primaria è auspicabile adottare un approccio attivo all'ambiente circostante che sia legato alle scienze motorie, è stato individuato come obiettivo di apprendimento la capacità di muoversi consapevolmente nello spazio circostante, orientandosi attraverso punti di riferimento ed utilizzando indicatori topologici come avanti, dietro, sinistra e destra.

Da questi suggerimenti, si possono individuare anche per educazione fisica degli obiettivi di apprendimento, nei quali si presuppone che l'alunno acquisisca consapevolezza di sé attraverso la percezione del proprio corpo. Dal movimento, infatti, l'alunno riuscirà ad accedere ad una maggiore padronanza degli schemi motori e posturali che si possano costantemente adeguare alle variabili spaziali.

Infine per quanto riguarda gli obiettivi di apprendimento di arte e immagine, sono stati individuati nelle Indicazioni Nazionali lo sviluppo delle capacità creative dell'alunno attraverso l'utilizzo di codici e linguaggi espressivi. Grazie all'uso di *TurtleArt*, l'alunno può sperimentare strumenti e tecniche diverse per realizzare prodotti grafici, sviluppando nuove capacità creative.

Per quanto riguarda l'attuazione del progetto, è stato ideato in modo tale da essere suddiviso in tre fasi fondamentali:

- Attivazione delle pre-conoscenze: è la prima fase del progetto in cui si vanno a ricercare le conoscenze acquisite negli anni precedenti che sono legate ai percorsi e alla direzionalità. Si attiva sin da subito un approccio CLIL abbinato al TPR (*Total Physical Response*) poiché i comandi di direzionalità vengono presentati in lingua straniera e sono associati al movimento del corpo. Durante questa fase del progetto è molto importante che gli alunni acquisiscano il linguaggio specifico: dovranno di fatto essere in grado di sfruttare e comprendere al meglio i comandi poiché saranno utilizzati per tutta la durata del progetto.
- Incontro con le *BeeBot*: è la seconda fase del progetto in cui vengono ripresi i

comandi di direzionalità per associarli al movimento non del proprio corpo, ma a quello di un oggetto esterno. In questo modo si veicola la conoscenza dal corpo all'esterno per poter acquisire un nuovo punto di vista e riuscire ad effettuare i percorsi. Saranno presentati, inoltre, anche il quadrato e il rettangolo grazie alla programmazione dell'ape robotica. Questa è la fase più delicata di tutto il progetto: il successo di una buona programmazione sarà ottenuto in base alla capacità di astrazione che gli alunni riusciranno ad acquisire tramite *Bee*.

- Incontro con il computer: è l'ultima fase del progetto in cui si verificano gli apprendimenti acquisiti dagli alunni e la loro effettiva capacità di programmazione. Grazie all'uso di *TurtleArt*, prima gli alunni vengono lasciati liberi di esplorare il programma, poi viene verificato se riescono a realizzare autonomamente un quadrato ed un rettangolo.

Nella prima fase legata al gioco motorio vi saranno dedicate due lezioni di un'ora e mezzo, che si concentreranno unicamente sull'acquisizione dei movimenti fondamentali in lingua inglese (FORWARD-avanti; BACK-indietro; RIGHT-destra; LEFT-sinistra). Nel corso di questi incontri sono previste delle attività motorie che saranno successivamente trasposte sul quaderno: in questo modo sarà possibile migliorare la comprensione dei concetti appena appresi.

Durante la seconda fase, la più importante, i bambini dovranno uscire dal loro corpo per immedesimarsi nelle *BeeBot*: saranno dedicate sei lezioni di un'ora e mezzo ad acquisire questa capacità. Le prime due lezioni serviranno a comprendere che i comandi *right-left* indicano, in realtà, *turn right* e *turn left*: non uno spostamento quindi, ma una rotazione. Le successive lezioni serviranno ad acquisire la capacità di immedesimarsi nell'apina e percorrere strade attraverso i comandi imparati fino ad ora: già a partire dal terzo incontro si avrà un approccio in cui i bambini tenteranno di formulare correttamente percorsi da un punto di partenza (P) a

quello di arrivo (A).

Tramite l'uso delle *BeeBot* sarà verificata la correttezza dei comandi e della programmazione effettuata grazie a dei test fatti in gruppi: per gestire la classe e l'apprendimento degli alunni è stato scelto l'approccio del *cooperative learning*. Questa strategia didattica promuove il lavoro di gruppo al fine di raggiungere un obiettivo comune, quindi i bambini, per portare a termine le richieste, dovranno collaborare tra di loro (Calvani, 2011).

La quarta lezione prevista per questa fase promuove la programmazione delle *BeeBot*: dopo essere stati divisi in quattro gruppi da sei elementi ciascuno, su un pannello verrà disegnato un percorso e i bambini dovranno farlo ripercorrere all'ape robotica inserendo i comandi corretti. Seguirà una fase di verifica della correttezza dei comandi individuati in base alla quale sarà possibile capire l'effettivo livello di comprensione dei bambini.

Nella quinta lezione i bambini, che manterranno sempre la struttura a gruppi, dovranno inventare un percorso per le *BeeBot*: prima lo disegneranno, poi lo eseguiranno con i comandi corretti e, infine, verificheranno la correttezza della programmazione con l'apina.

Infine sarà necessaria una sesta lezione per mostrare i percorsi chiusi con l'ape e poter concludere introducendo il concetto di figura geometrica: un quadrato ed un rettangolo verranno disegnati su cartelloni di carta tramite i movimenti della *BeeBot* precedentemente programmata e dotata di un pennarello.

Nella terza fase si ha il passaggio mentale successivo: non vi saranno più i percorsi fisici, ma verranno usati unicamente quelli concettuali e disegnati sul computer. Nel corso delle due lezioni dedicate alla scoperta e all'uso di *TurtleArt* verranno prodotti disegni che raffigureranno sia i quadrilateri che verranno richiesti loro, sia un loro prodotto grafico ideato e realizzato in linguaggio *Logo*.

In questo modo il progetto si concluderà con l'acquisizione da parte dei discenti

di un nuovo modo di utilizzare i computer: saranno in grado di usare un *software* di programmazione a blocchi che permetterà loro di disegnare non solo delle figure geometriche, ma anche rappresentazioni più complesse.

Tramite queste nozioni, in un'ottica di approccio CLIL dell'insegnamento della lingua inglese, gli alunni conosceranno i termini stranieri legati alla direzionalità come *forward*-avanti; *back*-indietro; *right*-destra; *left*-sinistra, ma anche riguardanti il mondo animale *bee*-ape, *turtle*-tartaruga, *snake*-serpente. Alla fine del progetto i bambini avranno anche acquisito la capacità di immedesimarsi in corpi esterni e descrivere brevi percorsi.

All'interno di questo *corpus* di conoscenze, verranno, inoltre, gettate le basi per la costruzione di una forma mentale di tipo computazionale: questa tipologia di pensiero è fluida, non cristallizzata, complementare i processi mnemonici e implementa i processi di *problem solving* e *problem finding*.

3.1.1 ATTIVAZIONE DELLE PRE-CONOSCENZE PER IMPARARE MUOVENDOSI NELLO SPAZIO: ATTIVITÀ MOTORIA CON APPROCCIO TPR- Sin dall'avvento del cognitivismo è nota l'importanza delle pre-conoscenze nel processo di apprendimento degli allievi, in particolare grazie a Ausubel (1960) e la sua teoria degli *advance organizer*: questo termine si riferisce ad ogni tipo di strumento che offra, in forma comprensibile, un'anticipazione di quelli che saranno i punti essenziali da acquisire, mobilitando le conoscenze dello studente.

In accordo con questo principio, è stata ideata una prima fase di attivazione delle pre-conoscenze degli allievi, i quali avevano già avuto una prima esperienza con i percorsi durante l'anno scolastico 2015/2016. Le docenti, grazie ad un'attività

motoria, avevano chiesto ai discenti di muoversi nel verso di alcune frecce, rappresentate su delle *flashcards* che venivano mostrate loro. Le frecce indicavano le quattro direzioni fondamentali -destra, sinistra, avanti e indietro- che sono il fondamento per l'attività di programmazione in linguaggio *Logo*.

Partendo da questa esperienza, è necessario riprendere le conoscenze acquisite durante questo lavoro tramite un'attività motoria, per questo sono stati realizzati su dei fogli quattro grandi frecce con sotto scritto il loro significato in inglese. La prima attività del progetto, quindi, è stata dedicata ai giochi motori in cui sono stati effettuati dei percorsi: i comandi sono stati impartiti oralmente in lingua inglese e i bambini dovevano eseguirli correttamente.

I quattro comandi, che sono stati rappresentati su dei cartelli, sono:

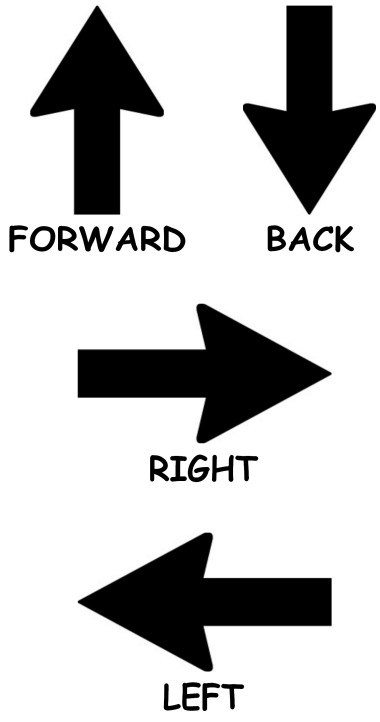

Illustrazione 1: Esemplificazione delle flashcards utilizzate

- FORWARD per indicare un passo in avanti, è rappresentato da una freccia rivolta verso l'alto (↑);
- BACK per indicare un passo indietro, è rappresentato da una freccia rivolta verso il basso (↓);
- RIGHT per un passo a destra, è rappresentato da una freccia rivolta verso destra (→);
- LEFT per un passo a sinistra, è rappresentato da una freccia rivolta verso sinistra (←).

Questa tipologia di approccio si basa sui principi del TPR, ovvero *Total Physical Response*, una metodologia di ampio uso all'interno dell'insegnamento della lingua

inglese, la quale prevede una serie di movimenti collegati a dei termini stranieri. Il TPR è una strategia didattica molto utile: riduce il rischio di frustrazione e aiuta gli alunni BES ad imparare sfruttando l'apprendimento cinestetico poiché viene percepito come gioco motorio e non come insegnamento.

In accordo con questa metodologia, è stato messo in atto un gioco motorio in cui metà classe riportava sul quaderno i percorsi fatti dal gruppo che eseguiva i comandi, in modo tale da interiorizzare le parole e i gesti associati ai termini in lingua inglese, mentre l'altra metà effettuava i percorsi dettati dall'insegnante. Essendo necessaria la scelta di un quaderno per la trasposizione didattica dei percorsi, è stato ritenuto il più adeguato quello di geografia.

Il primo fattore che ha suggerito questa scelta è che questa disciplina, in accordo con le Indicazioni Nazionali per il Curricolo, prevede come obiettivi di orientamento la capacità di muoversi consapevolmente nello spazio circostante, utilizzando gli indicatori topologici che, come già detto in precedenza, sono stati scelti per questa attività.

Il secondo motivo che ha portato a compiere questa scelta è di tipo pratico: la tipologia di quadrettatura della pagina del quaderno di geografia. I quaderni di geografia per la classe seconda, infatti, sono quadrettati con una misura di 1 cm per 1 cm. Considerata l'importanza dell'associazione del movimento corporeo alla rappresentazione grafica, tale quaderno sembra essere il più adatto a rappresentare il passo dei bambini come un quadretto della sua pagina.

La trasposizione sul quaderno di ciò che è stato fatto con il corpo è fondamentale poiché permette di creare solide basi conoscitive per il lavoro successivo. Grazie alla riproduzione grafica, infatti, si interiorizza e si apprende con maggiore precisione e facilità, quindi è un'operazione inscindibile dal lavoro che sarà fatto successivamente.

In data 31 gennaio 2017 è iniziato ufficialmente il progetto ideato con un

incontro durato poco meno di due ore. Dopo la seconda ricreazione la docente ha parlato ai bambini di un percorso piuttosto lungo ed interessante, che avrebbe portato a conoscere nuovi strumenti e persino ad utilizzare i computer. È stato raccontato loro che questo progetto sarebbe stato molto interessante poiché tutto si basava su dei movimenti, che sarebbero stati associati a dei comandi in lingua inglese. È stato spiegato loro che questi comandi erano delle semplici indicazioni spaziali corrispondenti ad un passo: in questo modo è stata subito collegata l'unità di misura non convenzionale ai movimenti del corpo e alla precisione dei comandi necessaria durante la pratica del *coding*.

All'interno dell'*Atelier Digitale* i bambini sono stati disposti su tre file, formando di conseguenza dei gruppi di circa sette/otto bambini ciascuna. Sono stati spiegati i comandi in lingua inglese con le frecce realizzate, collegandole fin da subito ai movimenti, in modo tale da entrare sin da subito in una didattica TPR. I bambini erano molto curiosi e ben disposti a giocare seguendo le istruzioni, ma le nozioni da apprendere erano molto alte e il tempo stimato per questa attività, circa mezz'ora, non è stato sufficiente per far acquisir loro i termini di base.

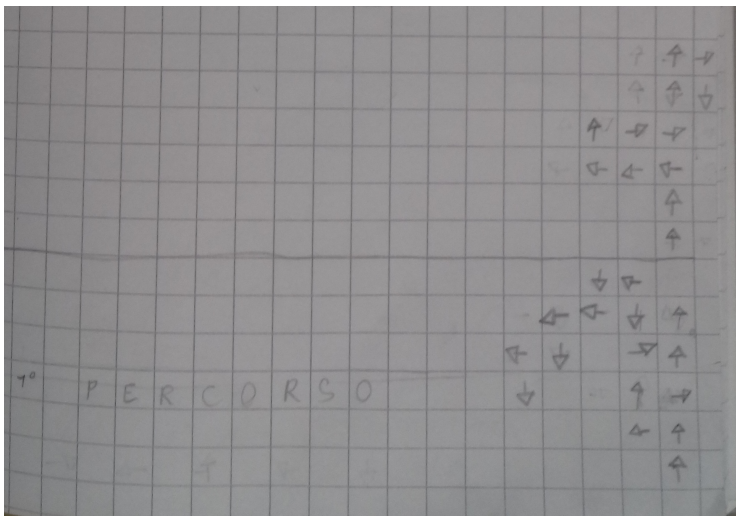

Illustrazione 2: Esempio di elaborazione dei percorsi del primo incontro

È stato mostrato loro il cartello con la freccia e pronunciato il comando in lingua inglese, ma nonostante l'uso delle rappresentazioni grafiche, i bambini erano molto insicuri e si sono confusi più di una volta sul da farsi. Per aiutarli la maestra Gaia ha pensato di attaccare ad

una lavagna metallica i quattro comandi in modo tale da formare una legenda che sarebbe stata indicata al momento della richiesta: in questo modo sperava di aiutare maggiormente i bambini nella comprensione delle quattro direzioni. Nonostante questo, i bambini hanno continuato a sbagliare i comandi, anche se con una minor frequenza.

Esaurito il tempo stimato per questa attività, la docente ha provato ad andare ugualmente avanti con il progetto: è stato proposto un nuovo gioco motorio simile a "*Snake*", il famosissimo videogioco per cellulari. In questo gioco è stata mantenuta la divisione in gruppi, sono stati disposti due a sedere ai banchi, mentre uno è rimasto in piedi in fila indiana per formare un serpente. È stato spiegato loro che sarebbero stati impartiti i comandi al serpente e che ciascuno di loro doveva stare attento a non sbagliare altrimenti sarebbe stato eliminato. Gli altri bambini che erano a sedere avrebbero rappresentato il percorso effettuato sul quaderno con le frecce corrette.

I bambini si sono dimostrati subito entusiasti di questo gioco perché era maggiormente competitivo e permetteva di testare le proprie capacità: la motivazione del gruppo classe, in questo modo, era aumentata significativamente. Nonostante questo le difficoltà non hanno tardato a ripresentarsi più diffuse e più gravi di prima: inizialmente nessuno aveva capito come trasporre il percorso sul quaderno perché non riuscivano ad immedesimarsi nei loro compagni che si spostavano.

È stata proposta l'attività per più volte nella speranza che riuscissero a comprendere meglio questo processo, ma era evidente che i bambini avevano bisogno di più tempo per assimilare le nozioni. La lezione è stata conclusa tornando in classe: è stato preso atto della necessità di approfondire questa prima attività con ulteriori incontri.

La maestra Annalisa ha proposto di riprendere il discorso con un ulteriore approfondimento per collegare il comando con il movimento: ha fatto disegnare una legenda sul quaderno di geografia, facendo riportare accanto ad ogni freccia anche

l'iniziale del termine in inglese (F ↑; B ↓; R →; L ←). In questo modo i bambini hanno iniziato a rappresentare i percorsi non graficamente, ma con le lettere e solo successivamente con il disegno. Questa attività è stata fondamentale per riuscire a far assimilare in modo completo i comandi in lingua inglese.

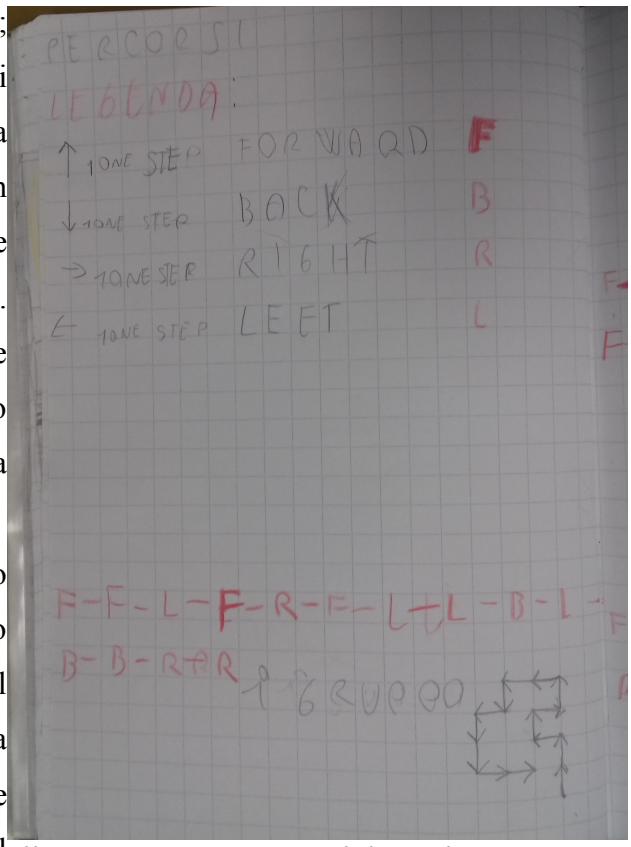

Illustrazione 3: Esempio di legenda e percorsi proposti dalla maestra Annalisa

Il secondo incontro è stato svolto il 3 febbraio 2017, è durato circa un'ora e mezzo, nel corso del quale l'attenzione dei bambini è stata focalizzata sulla rappresentazione della freccia lungo il perimetro del quadretto piuttosto che disporla al centro di esso, in modo tale da aumentare la precisione del disegno e della descrizione del comando.

I bambini sono tornati nell'*Atelier Digitale*, dove la docente ha ripetuto le attività già svolte durante il primo incontro, verificando che le difficoltà emerse precedentemente erano quasi totalmente arginate. Dopo una prima attività di movimento effettuata dai bambini disposti su tre file, sono stati proposti nuovi percorsi con il gioco "*Snake*" che sono stati portati a termine con una maggior precisione e un minor numero di errori.

Dopo questo incontro è stato deciso di continuare ad implementare le nozioni di base ripetendo, per una terza volta, una lezione simile alle precedenti, ma che

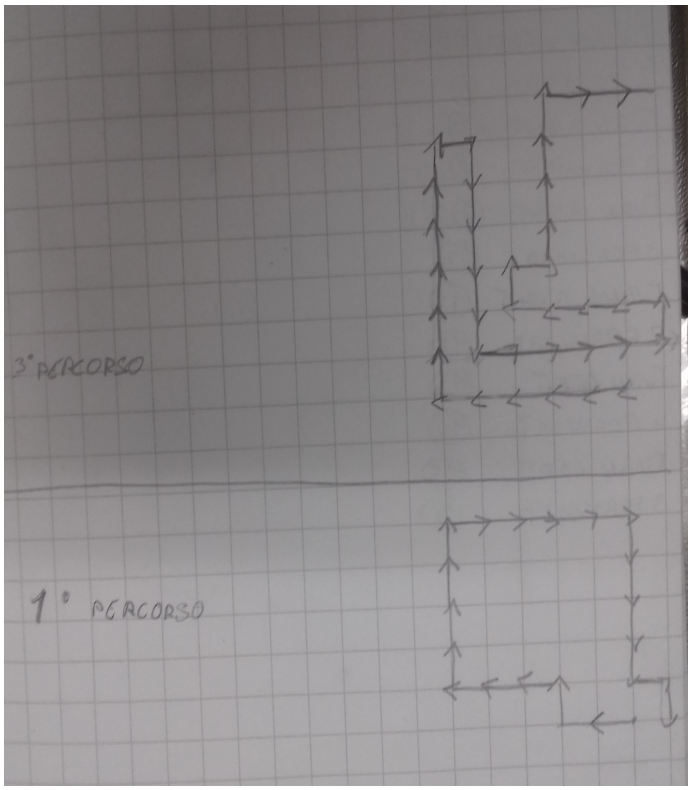

Illustrazione 4: Esempio di elaborazione di percorsi del secondo incontro

introducesse anche il concetto di abbinare i comandi a dei numeri che ne indicasse una loro ripetizione per un preciso quantitativo di volte.

3.1.2 PRIMO INCONTRO CON *BEEBOT*: IMPARARE ATTRAVERSO I PERCORSI- Dopo aver acquisito i termini di direzionalità e la loro funzionalità, si entra nella seconda fase del progetto. Una volta che i bambini si sono appropriati dei comandi di base del linguaggio *Logo* -FORWARD, RIGHT, LEFT, BACK-, si ha avuto un nuovo passaggio mentale legato alla spazialità: non dovevano apprende unicamente con il proprio corpo, ma con uno strumento esterno che permesse loro di dislocare il movimento. Questa capacità è risultata essere piuttosto complicata da acquisire e si lega alla competenza di pensare e programmare movimenti per altri oggetti al fine di

ottenere il percorso desiderato.

Un'ulteriore capacità da acquisire tramite questo strumento è legato all'uso del comando RIGHT e LEFT: nella programmazione in linguaggio *Logo* quest'ultimi non indicano uno spostamento, ma una rotazione. Per questo motivo i bambini hanno dovuto apprendere e comprendere che, in realtà, questi due comandi significano TURN RIGHT e TURN LEFT, cioè “gira a destra” e “gira a sinistra”. Non è stato possibile inserire questi due comandi sin dall'inizio con questa accezione di significato poiché prima di questo passo, infatti, era necessario verificare e consolidare la distinzione tra la destra e la sinistra.

Lo strumento più adatto per raggiungere questa competenza è stato l'uso della *BeeBot*, un'ape robotica che può essere programmata grazie all'uso di alcuni pulsanti sul suo dorso: in questo modo sono state stimolate le abilità dei bambini ad immedesimarsi in un oggetto esterno, ma contemporaneamente hanno imparato a scandire nel corretto ordine i comandi.

Illustrazione 5: Il dorso di una BeeBot

Ogni pulsante si riferisce ad un preciso comando, ma la peculiarità di quest'ultimo è che non è più universale poiché l'ape si muove a seconda di dove si trova. Questi pulsanti sono:

- ↑ indica il comando FORWARD, la *BeeBot* si muove un passo avanti di fronte a sé;
- ↓ indica il comando BACK, il robot si muove un passo indietro a sé;

- ← indica il comando LEFT, l'ape ruota a sinistra con un angolo di rotazione di novanta gradi;
- → indica il comando RIGHT, *Bee* ruota a destra di novanta gradi rispetto alla sua posizione;
- X indica il comando ERASE, i comandi precedentemente inseriti nell'apina vengono cancellati;
- II indica il comando PAUSE, premendolo la *BeeBot* interrompe il suo percorso, ma lo mantiene in memoria ed è pronta a continuarlo dal punto in cui si è fermata.
- GO è il comando che indica la conclusione della programmazione del robot, quindi, premendolo, l'ape inizia a muoversi.

Le attività legate a questo robot sono molteplici, quindi sono state tenute varie lezioni con questo strumento. L'idea iniziale è stata quella di creare dei cartelloni da mettere sul pavimento, ma successivamente sono stati forniti dalla scuola dei pannelli già quadrettati dell'unità di passo di *Bee*, che erano stati acquistati assieme all'ape robotica: questi mezzi sono stati fondamentali per fornire al bambino uno strumento assimilabile al suo quaderno. Il quadrato così rappresentato, inoltre, permette di ottenere un'unità di misura non convenzionale e il passo del robot è stato associato alla quadrettatura del quaderno. Di fatti in questo modo si gettano le basi per la comprensione delle unità

Illustrazione 6: Esempio di tabellone utilizzabile con le BeeBot

di misura poiché si ha avuto una diretta corrispondenza tra il quadrato del cartellone con il quadretto del quaderno: quest'ultimo è stato considerato un mezzo per misurare lo spostamento di *Bee*.

Nella terza lezione del 7 febbraio 2017, durata circa un'ora e mezzo, i bambini sono ritornati nell'*Atelier Digitale* della scuola ripetendo per la terza volta, in modo piuttosto sintetico e rapido, le attività svolte in precedenza. In aggiunta alle ormai usuali attività, è stato inserito all'interno del comando il valore numerico in inglese del movimento, in modo tale da fa muovere i bambini più passi per volta.

Il timore emerso durante questa attività è stato quello di complicare ulteriormente il compito e di non ricevere, di conseguenza, un'adeguata risposta, ma la docente ha potuto constatare che la velocità di esecuzione, la comprensione, il collegamento tra i comandi, le azioni e i disegni fossero adeguate alle aspettative.

Terminata l'attività, in accordo con la maestra Annalisa, è stata presentata la *BeeBoot* ai bambini. È stata posizionata un'apina al centro della lavagna metallica con i comandi disegnati sopra per mostrare la somiglianza tra il dorso della *BeeBoot* e gli indicatori di posizione che erano stati imparati nelle lezioni precedenti. Dopo questa operazione la docente ha mostrato loro un percorso con la *BeeBot*, che era stato già effettuato con il gioco motorio, riportando i comandi così come erano stati da loro scritti: è stato evidente sin da subito che non si muoveva esattamente come si aspettavano.

I bambini hanno imparato, grazie ad alcuni tentativi ed errori, che la freccia LEFT e la freccia RIGHT non indicavano il proseguire a destra o a sinistra dell'ape: questi comandi rappresentavano una rotazione. Per questo motivo è stato introdotto il termine TURN seguito da LEFT o RIGHT per indicare la rotazione a destra o a sinistra della *BeeBot*. Contestualmente i bambini hanno subito capito che non bastava premere un pulsante di rotazione per far muovere l'apina, ma che servisse anche la freccia FORWARD per far proseguire *Bee* sul suo percorso.

A conclusione della lezione, i bambini, grazie l'aiuto della docente, hanno provato a programmare la *BeeBot* con il percorso già effettuato in precedenza: è stato complicato per gli alunni immedesimarsi nell'apina, più volte hanno dovuto ripercorrere il tragitto fisicamente per individuare i giusti comandi, ma alla fine sono riusciti a portare a termine il compito assegnato.

Il 13 febbraio 2017 è stato condotto il quarto incontro ed è durato circa un'ora e mezzo. Considerando che erano stati introdotti dei nuovi comandi, la docente ha pensato di presentare loro una nuova tipologia di esercizio che potesse implementare le nuove acquisizioni, in modo da evitare situazioni analoghe al primo incontro che potessero creare un clima negativo alla classe e per l'apprendimento. In classe è stata distribuita una fotocopia quadrettata, sulla quale erano raffigurati un punto di partenza, uno di arrivo ed alcuni elementi grafici.

La docente ha spiegato loro che avrebbe dettato una sequenza di comandi che loro dovevano prima trascrivere sul quaderno, poi raffigurarli sulla fotocopia.

In questo modo si è voluto verificare se i bambini avessero compreso il comando TURN LEFT e TURN RIGHT e se riuscissero a rappresentare il percorso corretto. I comandi dettati sono stati:

- TURN RIGHT
- FOUR (4) FORWARD
- TURN LEFT
- ONE (1) FORWARD
- TURN LEFT
- TWO (2) FORWARD
- TURN RIGHT
- THREE (3) FORWARD
- TURN RIGHT

- THREE (3) FORWARD
- TURN RIGHT
- TWO (2) FORWARD
- TURN LEFT
- FOUR (4) FORWARD
- TURN LEFT
- THREE (3) FORWARD

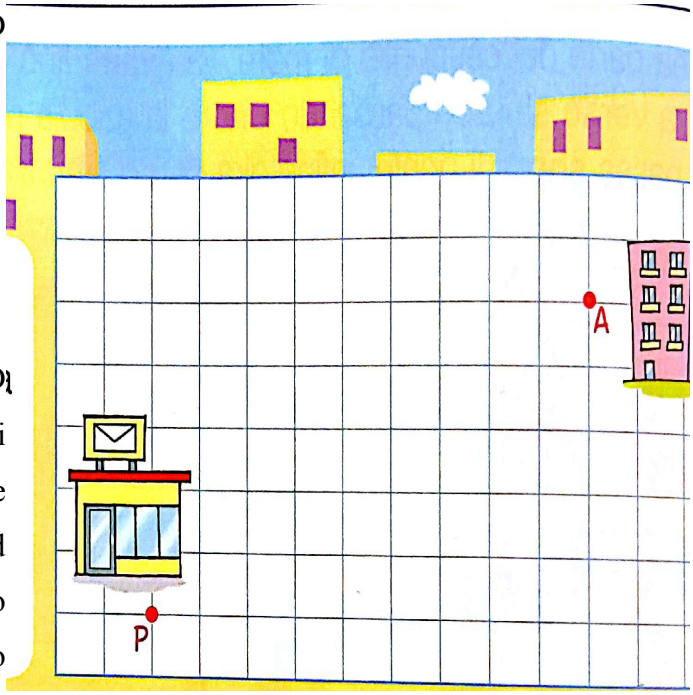

Illustrazione 8: Prima fotocopia fornita per svolgere l'esercizio

Questa sequenza di comandi è risultata essere troppo lunga per i bambini ed ha prodotto un percorso troppo breve poiché, oltre al comando di movimento, era necessario considerare anche il cambio di direzione. È sembrato subito evidente che bisognasse aggiustare il tiro e ridurre il numero dei comandi per permettere ai bambini di poter comprendere meglio ed eseguire i percorsi in modo adeguato.

Contestualmente a questa scelta, è stato ridotto drasticamente il numero dei comandi, scendendo a sei, ed è stata distribuita un'ulteriore fotocopia per eseguire il secondo esercizio. Seguendo le indicazioni usate precedentemente, i bambini hanno trascritto prima i comandi dettati, poi li hanno riportati sulla fotocopia. I comandi dettati sono stati:

- TURN LEFT
- SEVEN (7) FORWARD
- TURN LEFT
- THREE (3) FORWARD

- TURN RIGHT
- FOUR (4) FORWARD

A differenza di quanto è accaduto nel primo esercizio, i bambini hanno mostrato di avere minori difficoltà ad eseguirlo, facendo pochissimi errori e correggendosi autonomamente.

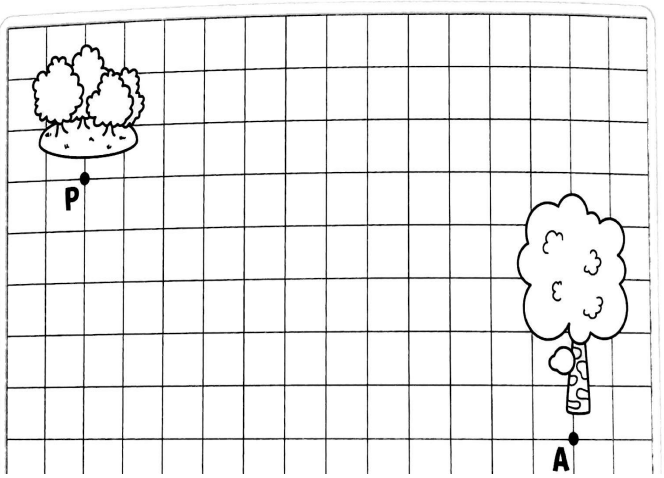

Illustrazione 9: Seconda fotocopia fornita per eseguire l'esercizio

Il quinto incontro si è tenuto il 17 febbraio 2017, in una lezione di un'ora e mezza circa. Durante questo incontro è stato deciso di lasciare maggiore autonomia ai bambini, che, lavorando in otto gruppi, hanno dovuto elaborare un percorso. Il compito, infatti, consisteva nel progettare percorsi che partissero da un punto P ad un punto A: una pedina blu indicava la partenza (P) e una gialla per l'arrivo (A).

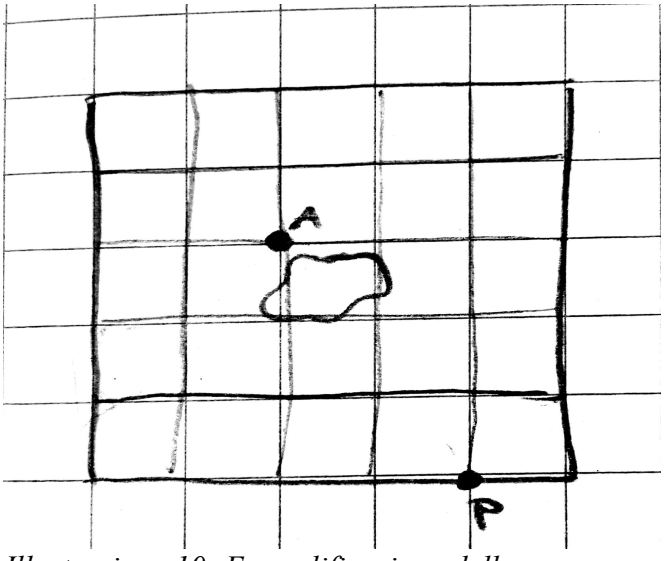

Illustrazione 10: Esemplificazione della disposizione dei punti di partenza (P) e arrivo (A) all'interno del tabellone usato

Data questa disposizione, i bambini hanno trascritto su un foglio la disposizione delle pedine e dell'area di lavoro per lasciare spazio agli altri attorno al tabellone. Dopo questa fase preliminare, i vari gruppi hanno progettato e individuato il percorso che la *BeeBot* doveva effettuare per portare a termine le richieste.

Dopo la fase di

progettazione, i gruppi, uno per volta, sono venuti a testare l'efficacia della loro programmazione attraverso la verifica con la *BeeBot*: nessuno è riuscito ad individuare, al primo tentativo, il percorso desiderato. Per questo motivo, a rotazione, i vari gruppi hanno provato più volte i percorsi per riuscire a raggiungere il punto di arrivo (A): programmavano la *Bee*, sbagliavano e tornavano a posto a correggere l'errore. Alla fine della lezione, grazie agli errori commessi, quelli visti dai loro compagni e l'aiuto dell'insegnante, tutti sono riusciti a completare il compito.

I bambini, tendenzialmente, commettevano tutti il solito tipo di errore: il comando TURN LEFT/RIGHT. Non comprendevano bene, infatti, come mai l'ape iniziasse a ruotare su sé stessa invece di proseguire nella direzione indicata e continuavano, nonostante i suggerimenti, ad inserire più volte il comando di rotazione al posto di quello di avanzamento. Compreso l'errore i gruppi hanno superato con facilità l'esercizio.

Durante l'incontro è stato richiesto anche un altro compito che ripercorre, a livello mentale, al contrario questa operazione: da un percorso stabilito e disegnato con del nastro adesivo di carta, i bambini dovevano ricavare i comandi per far muovere la *BeeBot*.

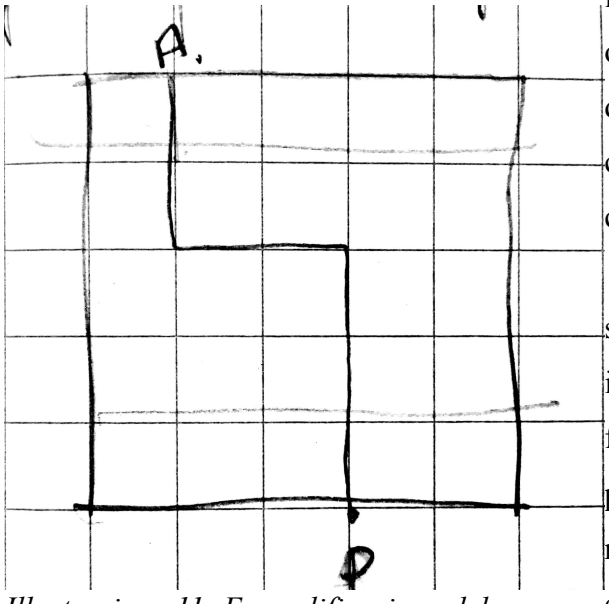

Illustrazione 11: Esemplificazione del percorso disegnato con lo scotch di carta sul tabellone usato.

È stato dato del tempo per poter scrivere e pensare i codici per eseguire il percorso, dopo di che è seguita una fase di verifica della correttezza dei percorsi effettuati. Tutti i gruppi sono riusciti ad individuare il percorso corretto, inoltre l'errore nato dal TURN LEFT/RIGHT è risultato

essere quasi totalmente scomparso. Il vedere l'errore nel percorso dell'apina ha permesso loro di migliorare e comprendere al meglio dove sbagliavano, specialmente grazie al confronto con la programmazione degli altri gruppi. L'errore, come rilevato da Papert, non è stato percepito con un'accezione negativa, ma è stato il punto di partenza per lo sviluppo della conoscenza e della risoluzione dei problemi, quindi non è insorta frustrazione.

Il 21 febbraio 2017 si è tenuto il sesto incontro, in cui è stata condotta una lezione di un'ora e mezzo. Durante questa lezione la docente ha ritenuto utile provare ad aumentare il livello di difficoltà e di astrazione dei bambini. Per stimolare le loro capacità di *problem solving*, è stata programmata una *BeeBot*, che ha svolto il percorso inserito: i vari gruppi avrebbero dovuto individuare i comandi che avevano permesso all'ape robotica di muoversi in quel modo.

I comandi inseriti nella *BeeBot* che i bambini dovevano riconoscere erano:

- THREE (3) FORWARD
- TURNRIGHT
- TWO (2) FORWARD
- TURN LEFT
- ONE (1) FORWARD
- GO

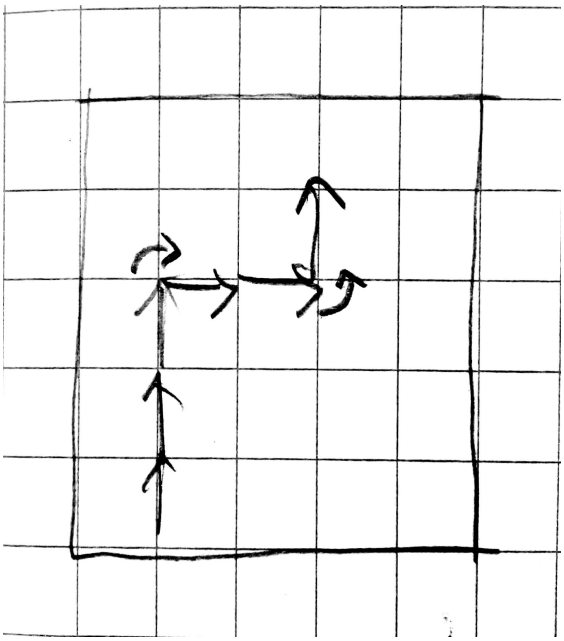

Illustrazione 12: Esemplificazione del percorso effettuato dalla BeeBot sul tabellone utilizzato

Tutti i gruppi sono stati in grado di individuare la corretta programmazione dell'ape nonostante il percorso non fosse stato rappresentato graficamente: tramite la memoria visiva e il *problem solving* i bambini hanno messo in campo la forma di intelligenza che la docente intendeva implementare,

ovvero il pensiero computazionale.

Per risolvere il problema della mancanza di stimoli grafici, infatti, i bambini hanno raffigurato il percorso su un foglio di carta e poi hanno trovato la programmazione adeguata. In questo modo hanno risolto il problema pratico e hanno potuto concentrarsi unicamente sulla risoluzione dell'esercizio.

Il 28 febbraio non è stata svolta la lezione come al solito poiché la scuola era impegnata nei festeggiamenti del Carnevale, tuttavia è interessante riportare la scelta delle docenti per quanto riguarda il tema delle maschere. Le maestre, infatti, hanno preparato assieme ai bambini dei travestimenti composti da dei cartelloni di cartoncino colorato a forma di freccia. Posizionandoli su delle magliette bianche, gli alunni rappresentavano uno specifico comando: in questo modo il *coding* è diventato il tema portante del Carnevale della classe.

Illustrazione 13: La maestra Morena impegnata nel percorso dettato dai bambini

Le docenti, inoltre, si erano travestite da api e, a turno, diventavano delle *BeeBot* che i bambini dovevano guidare ai fiori sparsi lungo la scuola: un gioco di ruolo a tema carnevalesco che integrava gli aspetti legati al progetto con la componente ludica del martedì grasso. È stato molto divertente vedere come i bambini si impegnavano per creare percorsi adeguati, giocando con le docenti, proponendo soluzioni e correggendosi in un'ottica di collaborazione attiva.

In data 3 marzo 2017 si è svolto il settimo incontro dedicato al progetto ed è

durato circa un'ora e mezzo. Durante questa lezione il gruppo classe è stato diviso in sei gruppi da quattro bambini ciascuno, fornendo a ciascuno di essi un tabellone di carta preparato appositamente, riquadrato con la misura esatta del passo della *BeeBot*.

Illustrazione 14: Il gruppo blu impegnato nella programmazione della BeeBot sul tabellone di carta

Il fine dell'incontro è stato quello di progettare, disegnando, un percorso a libera scelta per ciascun gruppo, ma a differenza delle volte scorse, le operazioni di *coding* e di *testing* sarebbero state fatte autonomamente poiché, grazie al numero delle api robotiche fornite dalla scuola, era possibile utilizzare un'apina a gruppo. Una volta raggiunto il risultato sperato sul disegno, i bambini avrebbero chiamato l'insegnante per mostrare il loro operato.

Durante l'incontro ciascun gruppo ha lavorato autonomamente, realizzando uno o più percorsi da mostrare alla docente, discutendo sul modo di risoluzione del problema e degli errori commessi. Per conferire maggiore significatività all'operato, i bambini riportavano prima su un foglio e, una volta rientrati in classe, sul quaderno i percorsi di gruppo progettati, in modo da documentare ciò che avevano fatto.

Nel corso dell'incontro alcuni gruppi hanno ideato percorsi chiusi che raffiguravano quadrati e rettangoli: da questa loro rappresentazione è stato deciso di proseguire nel progetto, passando alla presentazione di questi quadrilateri durante l'incontro successivo.

L'ottavo incontro si è svolto in data 3 marzo 2017 ed è durato un'ora e mezzo:

inizialmente gli alunni sono rimasti in classe, a conclusione della lezione sono andati nell'*Atelier Digitale*. La docente e gli alunni hanno pianificato un percorso che iniziasse e finisse nel solito punto di partenza: in questo modo sono stati individuati i codici per realizzare un rettangolo e un quadrato con le *BeeBot*. Attraverso una discussione, infatti, è stata individuata la programmazione per realizzare un "rettangolo stretto", ovvero:

- TWO (2) FORWARD
- TURN LEFT
- SIX (6) FORWARD
- TURN LEFT
- TWO (2) FORWARD
- TURN LEFT
- SIX (6) FORWARD

In questo modo i bambini hanno scoperto che per fare un rettangolo servivano due indicazioni uguali per ciascuna coppia di lati "verticali" e "orizzontali": nel loro caso, quelli verticali dovevano essere lunghi due passi, mentre quelli orizzontali di sei. Inoltre hanno capito che bisognava girare sempre nella stessa direzione per chiudere il percorso, altrimenti sarebbe restato aperto. Per realizzare il quadrato, invece, sono stati individuati i seguenti comandi:

- THREE (3) FORWARD
- TURN RIGHT
- THREE (3) FORWARD
- TURN RIGHT
- THREE (3) FORWARD
- TURN RIGHT
- THREE (3) FORWARD

Dopo aver individuato i seguenti comandi, la docente ha portato gli alunni nell'*Atelier Digitale* e sono stati usati dei pannelli realizzati con un foglio di carta da pacchi, riquadrati con la misura del passo della *BeeBot*. È stata applicata una penna sulla parte posteriore dell'ape ed è stata programmata come progettato in classe: la docente ha fatto partire la *BeeBot*, la quale, con un po' di fatica a causa del dislivello creato dalle mattonelle, ha disegnato la figura desiderata.

La stessa operazione è stata ripetuta anche per il quadrato ed è emersa un'osservazione interessante da parte di una bambina: "L'ape rispetta solo i quadretti". La *BeeBot*, infatti, segue il perimetro del quadretto, quindi, quando gira, lo fa seguendo l'angolo di quest'ultimo. Da questa semplice osservazione la bambina ha introdotto il concetto di angolo: i quadretti hanno una misura interna di 90° e l'ape gira su sé stessa di questa angolazione.

Il 7 marzo 2017 si è tenuto il nono incontro del progetto, che è durato circa un'ora. Durante questa lezione sono stati formati sei gruppi di lavoro per realizzare una serie di percorsi chiusi. Ogni gruppo ha progettato due percorsi, uno rappresentava un quadrato e l'altro un rettangolo, che sono stati verificati grazie alla possibilità di usare sei apine contemporaneamente.

I bambini, infatti, hanno potuto prima progettare, poi programmare l'apina senza dover passare fin da subito dalla verifica della docente. Come la volta precedente, quindi, la verifica è stata effettuata in modo autonomo e, una volta rappresentate le figure desiderate, la docente è stata chiamata per mostrare il loro operato.

Una volta rientrati in classe gli alunni hanno riportato i progetti di gruppo sul quaderno, raffigurando le figure, delimitandone la regione interna e quella esterna: in questo modo sono state fornite le prime nozioni sui quadrilateri, riguardanti il perimetro e l'area di queste ultime.

3.1.3 PRIMO INCONTRO CON IL COMPUTER PER LA VERIFICA DELLE CONOSCENZE: COSA SAPPIAMO FARE?- Giunti a questo punto si entra nella terza fase del progetto, quella che segna la conclusione del percorso. Per questa fase i bambini hanno avuto un primo approccio con i computer per poter scoprire ed imparare ad usare *TurtleArt*, il programma scelto per creare un collegamento tra il *coding* e le conoscenze apprese con le *BeeBot*. Come illustrato nel capitolo precedente, questo *software* prevede una tipologia di programmazione a blocchi e non *text-based*, quindi non necessita di immettere per iscritto i comandi, ma solo di cliccare sui vari mattoncini al fine di far muovere la tartaruga come le ape robotiche.

La scelta di questo programma è stata dettata per salvaguardare gli alunni più fragili della classe, ovvero i cinque bambini considerati BES: per questa tipologia di studenti, infatti, è bene non promuovere fin da subito un insegnamento scritto della lingua straniera, ma è necessario privilegiare un approccio orale a tale materia.

Tramite questo *software* non solo è stato possibile veicolare l'insegnamento della lingua inglese, ma ha potuto fornire un primo approccio alla programmazione tramite l'uso di blocchi logici facilmente assimilabili e comprensibili anche dai bambini con più difficoltà. Da un punto di vista grafico, inoltre, *TurtleArt* è stato disegnato come un *software* molto accattivante e colorato, quindi agli occhi di un bambino è potuto sembrare un semplice videogioco ma, in realtà, veicolava un forte aspetto educativo.

Come è stato già illustrato all'inizio del capitolo, inizialmente era stato pensato di dedicare solo due lezioni per questa fase del progetto, ma in realtà è sembrato doveroso approfondire ed estendere maggiormente questo aspetto. È risultato essere necessario, infatti, non solo ampliare la spiegazione delle funzionalità del programma, cosicché non si potesse incorrere nell'errore di sopravvalutare le capacità di astrazione dei bambini, ma anche mostrare nuovi comandi, permettendo alla fascia alta della classe di sviluppare capacità più elevate.

Il nono incontro si è svolto il 10 marzo 2017 ed è durato circa un'ora. In questa lezione è stato completato il lavoro di trasposizione dei percorsi fatti la volta precedente con la *BeeBot* sul quaderno di matematica, poi la classe si è spostata nell'*Atelier Digitale* per introdurre il passaggio successivo del progetto: l'uso del computer.

Durante questo incontro è stato illustrato, a piccoli gruppi, come funzioni il programma scelto per imparare a fare *coding*. Sull'*active table* sono stati illustrati i comandi ed il funzionamento di *TurtleArt*: i bambini erano molto curiosi ed entusiasti di poter usare finalmente il computer. Sono nate molte domande, ma vedendo come sono posti i comandi è stato sin da subito evidente che vi fossero numerose analogie tra le apine e *TurtleArt*, quindi non è stato complicato capire come funzionasse il programma. Concluso il giro di presentazione, i bambini sono tornati in classe.

Il 13 marzo 2017 si è svolta la decima lezione, durata due ore, in cui i bambini, nella prima parte, hanno liberamente esplorato i comandi e le funzionalità di *TurtleArt*. Gli alunni sono stati divisi in undici gruppi di due membri ciascuno poiché erano stati messi a disposizione un tale numero di *laptop* e, inoltre, lavorare a coppie avrebbe aiutato anche gli alunni più fragili per riuscire a concludere i compiti grazie alla metodologia del *peer-tutoring*. Questa tecnica ha permesso sia agli alunni più fragili, grazie al supporto di un compagno, sia a quelli più esperti di acquisire una maggiore consapevolezza metacognitiva. Tale strategia, inoltre, ha creato la possibilità di permette agli alunni di raggiungere, secondo le teorie di Vygotskij, un livello più elevato della loro zona di sviluppo prossimale (Schaffer, 2008).

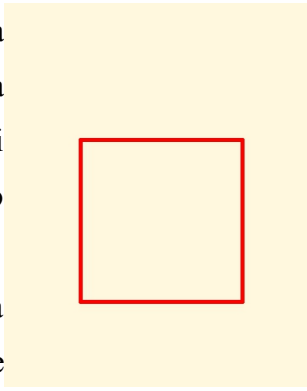

Illustrazione 15: un quadrato

La prima mezz'ora di lezione è stata dedicata alla scoperta dei comandi e del loro corretto uso, successivamente è stata aperta una discussione per scegliere i comandi per

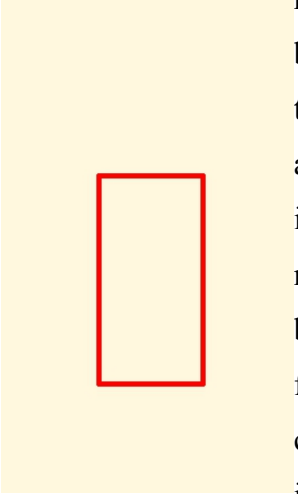

Illustrazione 16: un rettangolo "stretto"

realizzare un quadrato e, dopo, un rettangolo: il compito dei bambini è stato quello di programmare in modo corretto la tartaruga per ottenere il disegno richiesto. I bambini, assieme all'aiuto della docente, hanno cercato di ricordare i comandi inseriti nelle *BeeBot* per realizzare tali forme geometriche. È nata una discussione che ha coinvolto tutta la classe: i bambini hanno provato a percorrere i percorsi come se fossero la tartaruga ed hanno deciso quali fossero i comandi corretti da inserire nel programma affinché fossero realizzati i disegni.

Tutti i gruppi hanno realizzato il quadrato e il rettangolo senza problemi, anzi, alcuni di loro hanno scoperto che premendo più volte la sequenza di programmi era possibile far ripetere il movimento alla tartaruga. Questa scoperta è stata molto interessante poiché ha permesso di comprendere che questi strumenti possono avere in sé anche la funzione della ripetizione del comando.

Al termine della lezione i bambini hanno esplorato il programma senza vincoli, realizzando disegni intuitivi ed accendendo la loro fantasia. Durante l'esplorazione hanno usato anche nuovi comandi di cui non avevo mai parlato loro: si tratta della funzione ARC. Un bambino, infatti, ha iniziato a giocare con tutti i comandi e ha notato che uno di essi si chiamava ARC: ha associato la parola ad arco, ed ha scoperto che con questo codice si può creare una circonferenza.

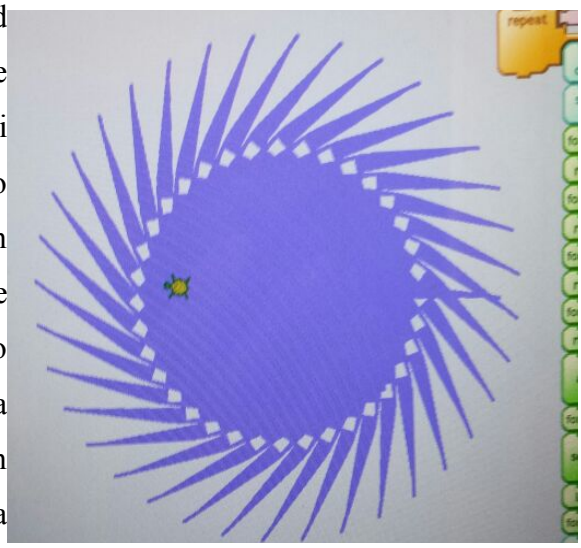

Illustrazione 17: esempio di esplorazione libera

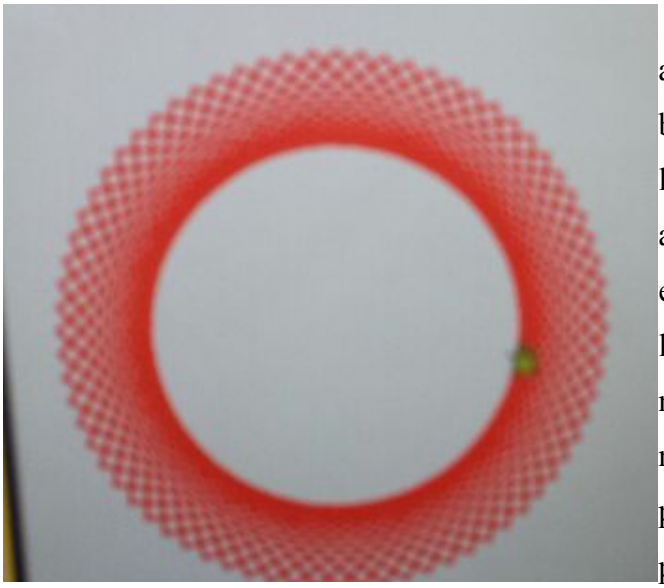

Illustrazione 18: Esempio di esplorazione libera

Al termine della lezione gli alunni sono rientrati in classe: i bambini erano entusiasti del lavoro svolto e delle scoperte che avevano conquistato. Il loro entusiasmo e fermento ha portato la docente a decidere di presentare nuovi comandi che, inizialmente, non dovevano essere inclusi nel progetto, cosicché i bambini potessero entrare in contatto con nuove funzionalità.

Il penultimo incontro è stato effettuato il 20 marzo 2017 ed è durato un'ora e mezza. Nel corso dell'incontro sono stati presentati altri comandi ai bambini: la funzione di START FILL/END FILL, ovvero per il riempimento dell'area, i vari colori della penna (PENCOLOR) e la funzione di REPEAT.

Con questi strumenti i bambini sono riusciti a cambiare il colore dello sfondo dell'area, creare nuovi disegni partendo dalle scale, passando alle finestre e poi chiudendo con un disegno ben più articolato che si basava sulla circonferenza. Alcuni di loro, infatti, hanno iniziato ad esplorare le varie funzioni, scoprendo di poter realizzare disegni molto complessi associando più funzioni assieme. Alcuni di loro, inoltre, hanno notato che, variando i numeri scritti accanto ai comandi era possibile realizzare disegni ancor più complessi. La riflessione a riguardo di questa scoperta è stata semplice: i bambini avevano intuito che quei numeri erano associati a come la tartaruga girasse su sé stessa (con RIGHT e

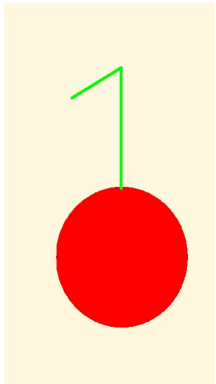

Illustrazione 19: una ciliegia realizzata durante l'ultimo incontro

LEFT) e di quanto potesse avanzare (con FORWARD e BACK).

Al termine della lezione sono rientrati in classe ed è stato assegnato un compito da svolgere per casa: i bambini avrebbero dovuto pensare ad un disegno ed al codice necessario per realizzarlo con *TurtleArt* nell'ultimo incontro.

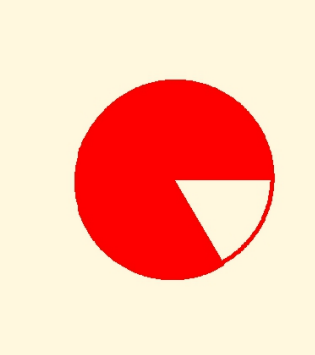

Illustrazione 20: una pizza

Nell'ultima lezione del 27 marzo 2017, durata circa un'ora, sono stati ripresi i computer e gli alunni hanno realizzato, grazie ai lavori svolti a casa dai bambini, i disegni progettati. Il risultato è stato molto interessante: un bambino ha realizzato una pizza a spicchi, un altro una ciliegia, un altro ancora un quadrato perfettamente riempito del colore arancione come desiderava. Non sempre hanno inserito il codice corretto e hanno dovuto lavorarci sopra per sistemarlo, ma, dopo svariati tentativi, sono riusciti nell'intento.

È doveroso sottolineare che i vari prodotti realizzati sono stati considerare di valore in base al livello dell'allievo: ad esempio, il bambino che ha realizzato il quadrato arancione è un alunno BES in attesa di certificazione, per cui il fatto che sia riuscito a portare a termine il suo intento senza l'aiuto dell'insegnante, è stato senza dubbio un grande successo, paragonabile ai suoi compagni che hanno realizzato la pizza o la ciliegia.

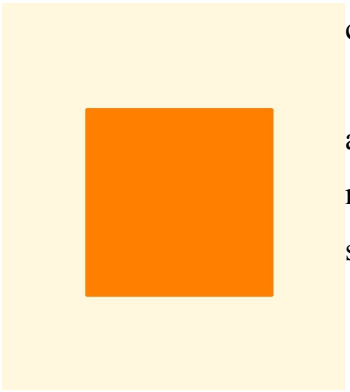

Illustrazione 21: un quadrato arancione

Al termine della lezione la docente ha accompagnato gli alunni in classe, segnando, in questo modo, la conclusione del progetto e del percorso della scoperta del *coding*.

Conclusioni

Ciò che rende la matematica ripugnante
ai bambini non è la difficoltà,
ma il procedere secondo un insensato
rituale dettato dai tempi di
un programma di studi prefissato.
Seymour Papert

Al termine del percorso è doveroso redigere un bilancio complessivo, in modo da individuare, criticamente, i punti di forza e di debolezza di questo progetto, cosicché si possa rendere una proposta più efficace.

Innanzitutto bisogna individuare i bambini che hanno riscontrato maggiori difficoltà nel processo e descriverne una loro evoluzione, per verificare che abbiano effettivamente raggiunto gli obiettivi prefissati. Tutti gli alunni, ad eccezione del bambino certificato con Legge-quadro 104/1992, sono riusciti ad eseguire tutte le tipologie di percorsi e realizzare i disegni che avevano ideato.

L'alunno certificato è, di fatto, rimasto molto indietro rispetto ai suoi compagni poiché ha avuto grossi problemi di salute: essendo cagionevole ed avendo le difese immunitarie molto deboli, il bambino ha potuto frequentare solo un paio di lezioni del progetto. Questa assenza forzata ha causato una grave lacuna poiché non ha permesso al bambino di partecipare in modo adeguato alle attività. Perdendo molte lezioni l'alunno, di fatto, non ha potuto acquisire le nozioni di base per i lavori successivi.

Per quanto riguarda gli altri alunni non vi sono stati grossi deficit da sottolineare: qualcuno tendeva a sbagliare a ruotare a destra o a sinistra, ma riconoscevano in fretta l'errore commesso, correggendolo di volta in volta.

In definitiva possiamo identificare tre fasce di acquisizione delle nozioni di *coding*:

- Una fascia alta, composta da tre bambini, i quali riuscivano a realizzare non solo le richieste della docente, ma anche disegni complessi ideati individualmente. Questi alunni sono riusciti a comprendere in modo approfondito le funzioni di REPEAT, ARC, START FILL e END FILL oltre ai comandi di base del linguaggio *Logo*, ovvero FORWARD, LEFT, RIGHT e BACK.
- Una fascia media, composta dalla maggior parte dei bambini, i quali hanno pienamente compreso e realizzato le richieste della docente, ma hanno avuto difficoltà a concepire disegni più complessi. Questi alunni hanno compreso molto bene i comandi di base del linguaggio *Logo*, ma non hanno pienamente acquisito il concetto legato alla ripetizione. I bambini appartenenti a questa categoria hanno realizzato disegni semplici, figure geometriche ripetute o colorate nella loro superficie interna.
- Una fascia bassa, composta da quattro bambini. Questi alunni hanno compreso i comandi basilari del linguaggio *Logo*, ma non inseriscono correttamente il codice all'interno del programma, puntando sulla ripetizione data dal premere più volte i blocchi realizzati. Questi bambini soddisfano le richieste della docente, ma non sono sempre in grado di comprendere come siano riusciti a produrre il risultato atteso. Nel disegno libero con il programma, infine, non hanno effettuato il compito richiesto poiché non hanno progettato a casa il disegno ed il codice da utilizzare. In questo modo il loro agire ha dimostrato superficialità e, quindi, mancanza di qualità nel

codice: nonostante i prodotti potessero definirsi interessanti, di fatto, li avevano realizzati tramite una ripetizione casuale di comandi e cliccando più volte su di essi. Questo comportamento non è da considerarsi adeguato all'apprendimento e colloca questi alunni nella fascia bassa di valutazione.

Per quanto riguarda le azioni e strategie didattiche adottate nel corso del progetto, si rintracciano delle negatività per quanto riguarda il lavoro a gruppi: la strategia del *cooperative learning* e del *peer tutoring* non ha dato i risultati sperati non tanto in termine di acquisizione delle nozioni, ma per quanto riguarda la cooperazione effettiva tra gli alunni. I bambini, infatti, non poche volte hanno litigato, interrotto la lezione e causato confusione, portando la docente a dover intervenire costantemente per ristabilire un ambiente favorevole per l'apprendimento.

Nonostante queste situazioni abbiano comportato maggiore confusione durante la lezione, si deve, tuttavia, ammettere che i bambini hanno imparato molto dall'osservazione e dalla discussione con i loro compagni, approfondendo in modo adeguato la comprensione dei comandi e movimenti da imparare. Per questo motivo ritengo che, nonostante tutto, queste strategie possano essere funzionali per i discenti: il clima rumoroso di questo laboratorio ha prodotto i risultati sperati, quindi sono tecniche che dovranno rimanere in un'esperienza futura.

Tutte le attività presentate nell'ultimo capitolo hanno, implicitamente, permesso lo sviluppo del pensiero computazionale poiché, grazie alla necessità di risolvere le situazioni problematiche nate dall'uso di questi *software*, i bambini si sono interrogati costantemente su cosa inserire per ottenere i disegni desiderati. Durante le attività prima con le *BeeBot* e poi con il computer, passando a sbirciare i lavori dei diversi gruppi, si poteva notare e sentire come i bambini discutessero su ciò che era stato fatto e su quello che dovevano ancora fare.

Questa discussione nata tra i membri dei vari gruppi, l'applicazione di strategie risolutive non convenzionali ed innovative, hanno creato situazioni in cui è stata

messa in gioco una capacità a volte sottovalutata nella scuola primaria: il pensiero computazionale. Grazie a questo assaggio di programmazione, infatti, i bambini hanno sperimentato una forma di pensiero nuova, che li ha portati a testare capacità di cui, fino ad allora, ignoravano l'esistenza. Gli alunni, infatti, hanno messo in gioco le loro capacità di *problem solving* che, con il progredire dell'esperienza di *coding*, ha portato a sviluppare delle strategie di *problem finding*: i bambini, dopo alcune lezioni con le *BeeBot*, hanno iniziato spontaneamente ad individuare i problemi prima che si presentassero, interrogandosi sin da subito sulla loro risoluzione.

Ovviamente non tutti i bambini sono riusciti a sviluppare completamente queste strategie, tuttavia l'osservazione dei compagni più esperti è stata un'esperienza da non sottovalutare: ricordando le tesi di Vygotskij, si può supporre che la loro Zona di Sviluppo Prossimale abbia avuto un *input* che, se stimolato nel tempo, porterà gli alunni ad acquisire questa capacità al massimo delle loro potenzialità.

Per quanto riguarda il numero delle lezioni e la loro articolazione, invece, dovranno essere strutturate in modo diverso, ma le tre fasi principali non dovranno variare. La struttura definitiva di questo progetto prevederà, come già teorizzato nei capitoli precedenti, una fase di attivazione delle pre-conoscenze in cui verrà proposta un'attività motoria con approccio TPR. Questa fase è quella preparatoria al lavoro successivo, per cui è il punto di partenza fondamentale di tutto il processo.

A differenza di quanto inizialmente teorizzato, non dovrà articolare l'apprendimento dei comandi in lingua inglese in un paio di lezioni, ma dovrà svilupparsi per almeno tre incontri. Il primo incontro dovrà coinvolgere gli alunni in un'attività motoria, prima attraverso una ripetizione dei comandi dettati per concludersi con un gioco ad eliminazione che è stato chiamato "*Snake*". Durante questo gioco sarà fondamentale far scrivere sul quaderno di geografia i movimenti compiuti dai compagni. A differenza di quanto precedentemente ipotizzato, i bambini non dovranno rappresentare le frecce, ma le lettere iniziali dei comandi. In questo

modo si crea un livello di apprendimento precedente al disegno della freccia, che non crei disturbo o conflitto con la comprensione dei comandi in lingua inglese.

Per i bambini, infatti, è un'astrazione di notevole livello rappresentare i percorsi con delle frecce nonostante siano raffigurate in una legenda, quindi è preferibile prima scrivere i comandi con le lettere e poi disegnare effettivamente il percorso.

Nel corso della seconda lezione i bambini dovranno rappresentare direttamente il percorso che viene loro dettato ed effettuato dai loro compagni. Si potranno aiutare con le lettere, ma dovranno cercare di sviluppare un più alto livello di astrazione. Nel corso di questo incontro verranno proposti anche percorsi già disegnati e che dovranno essere riscritti sotto forma di lettere, in modo da apprendere non solo il processo diretto, ma anche quello inverso.

Nel corso del terzo incontro verrà ripetuto il gioco motorio per verificare la completa comprensione dei comandi. Questa lezione sarà l'anello di congiunzione tra la prima fase e la seconda poiché potrà essere sfruttata per mostrare ai bambini le *BeeBot* e il loro funzionamento. Per questa fase saranno dedicate otto lezioni di approfondimento all'uso e alla creazione dei percorsi con le api robotiche, in vista dell'ultima parte del progetto, ovvero l'uso del computer.

In questa lezione verrà mostrato, quindi, il livello superiore di apprendimento che dovranno acquisire per proseguire nel progetto: i bambini scopriranno il concetto di rotazione a destra e a sinistra dell'ape, quindi la *BeeBot* effettua un cambio di direzione e non uno spostamento con questi due comandi.

La quarta lezione proseguirà alla scoperta della seconda fase del progetto, in cui saranno mostrati percorsi, che vadano dal punto di partenza (P) all'arrivo (A), disegnati su un tabellone e i bambini, divisi a gruppi, dovranno riscrivere nella forma corretta. In questa attività, quindi, gli alunni prenderanno confidenza sia con lo strumento da usare, la *BeeBot*, sia impareranno il concetto di rotazione a destra e a sinistra. I percorsi effettuati e i codici individuati verranno trascritti sul quaderno al

fine di migliorare la comprensione dell'attività e acquisire maggior consapevolezza del processo.

Nel quinto incontro verrà effettuato un consolidamento delle nozioni acquisite fino ad ora, in modo tale da implementare la confidenza sia con i termini in inglese, sia con le loro funzioni. Verrà svolta un'attività in aula in cui i comandi saranno dettati e gli alunni, dopo averli scritti, dovranno rappresentare su delle fotocopie il percorso dal punto di partenza (P) all'arrivo (A). La sequenza di comandi non dovrà superare sei elementi, altrimenti non solo il percorso risulterà essere di difficile comprensione, ma gli alunni potrebbero avere difficoltà a non commettere errori nella trascrittura.

Nella sesta lezione i bambini, divisi a gruppi, dovranno progettare un percorso che vada come i precedenti da P ad A e scrivere, di conseguenza, i comandi: prima lo disegneranno prendendo come punto di riferimento il tabellone e la sua area, poi dovranno individuare il codice da inserire e verificarlo con la docente. In questo incontro i bambini verificheranno le loro capacità, arginando gli errori grazie alla correzione dell'insegnante e al confronto con i loro compagni. Una volta individuata la giusta programmazione, i bambini riporteranno il codice ed il disegno del percorso sul quaderno.

Nel settimo incontro i percorsi non saranno tracciati graficamente, ma verranno mostrati due-tre volte tramite la *BeeBot*: in questo modo i bambini dovranno ricordare il percorso fatto, organizzarsi per trascriverlo e comprendere quali sono i comandi inseriti. È una tipologia di capacità molto elevata, ma necessaria per proseguire nelle attività. Al temine dell'incontro gli alunni dovranno trasporre sul quaderno i percorsi effettuati assieme alla relativa programmazione, in modo tale da rendere maggiormente significativa l'esperienza.

Nella nona lezione verranno forniti un tabellone riquadrato e una *BeeBot* a gruppo: durante questa attività i bambini saranno invitati a creare a gruppi dei

percorsi. A differenza degli altri incontri, in questo gli alunni non dovranno costruire un percorso da P ad A, ma lo inventeranno liberamente. Un'altra differenza, inoltre, è che i bambini non svolgeranno il *testing* e il *debugging* con la docente, ma lo faranno autonomamente e mostreranno i risultati solo una volta che saranno riusciti a realizzare ciò che avevano progettato.

Durante questa attività ogni gruppo riporterà i percorsi realizzati e ideati sul quaderno, in modo tale da potersi confrontare con i compagni. L'osservazione di vari percorsi, inoltre, potrebbe fornire degli spunti utili per illustrare le fasi successive del progetto, ovvero la presentazione del quadrato e del rettangolo: alcuni bambini, infatti, potrebbero ideare percorsi chiusi riconducibili a queste figure geometriche.

Il decimo incontro sarà strutturato in due momenti: inizialmente si terrà una discussione in aula in cui verrà effettuata una riflessione sui percorsi chiusi. Durante questo confronto i bambini verranno guidati per individuare un codice per realizzare un quadrato e un rettangolo: con il corpo, i disegni e un osservazione critica gli alunni giungeranno alla corretta programmazione. In un secondo momento l'insegnante, tramite un tabellone di carta e una *BeeBot* capace di scrivere, realizzerà il *coding* di queste due figure geometriche, in modo tale da creare un tabellone che riporti il lavoro di gruppo realizzato.

È fondamentale fornire ai bambini l'area in cui si dovrà realizzare il disegno, poiché questo dovrà restare all'interno del tabellone, quindi dovrà avere delle misure adeguate. Dopo l'uso della *BeeBot* i bambini scriveranno i codici accanto al tabellone, in modo tale da poter rappresentare in modo semplice e lineare la programmazione di queste figure geometriche. Nella lezione successiva, inoltre, gli alunni riporteranno sul quaderno le figure realizzate in classe e saranno guidati in una riflessione: in questo modo comprenderanno la differenza tra il perimetro e l'area dei quadrilateri. Questa attività li porterà ad acquisire nuove nozioni che potranno essere utili per le lezioni e gli apprendimenti successivi.

Nel dodicesimo incontro gli alunni dovranno replicare il codice per disegnare un quadrato e un rettangolo con le *BeeBot*, in modo tale che venga completamente assimilato il concetto di programmazione dei quadrilateri. Questa attività verrà svolta a gruppi e per ciascuno di essi verrà fornito un tabellone e un'ape robotica, in modo tale da poter effettuare le operazioni di *testing* e di *debugging* autonomamente.

Queste strategie permetteranno di ottenere un riscontro degli effettivi apprendimenti degli alunni in vista della successiva fase del progetto: l'uso del computer. Le lezioni dedicate a questa fase del progetto saranno quattro e prevederanno un lavoro simile a quanto effettivamente effettuato all'interno della proposta descritta.

La tredicesima lezione sarà incentrata sulla scoperta di *TurtleArt*: in questo incontro, infatti, i bambini potranno vedere il programma tramite un *active table* oppure su una LIM. In questo modo gli alunni potranno conoscere in modo più approfondito il *software* e utilizzarlo con maggiore sicurezza. Verranno illustrati anche comandi nuovi e che sono destinati alla fascia di apprendimento più alta, ovvero gli strumenti di START FILL, END FILL e REPEAT: questa anticipazione sarà utile per degli apprendimenti successivi in un'ottica di progettazione in continuità con la classe successiva.

Nel quattordicesimo incontro gli alunni avranno il loro primo e vero approccio all'uso del computer: dovranno accenderlo, accedere nella giusta partizione e lanciare *TurtleArt*. È preferibile impostare un lavoro che sia prettamente individuale, nel caso non sia possibile è bene cercare di mantenere una struttura a piccoli gruppi composti da due-tre elementi massimo. Durante questa lezione i bambini entreranno in contatto con il programma e potranno esplorarlo liberamente, affinché possano acquisire maggior sicurezza del mezzo. Saranno liberi di disegnare ciò che vogliono tramite un'attività libera e non vincolante.

La quindicesima lezione i bambini avranno un approccio più sistematico all'uso

del programma, quindi non dovranno esplorare liberamente il *software*, ma dovranno realizzare un quadrato e un rettangolo immettendo la giusta programmazione. Agli alunni che finiranno in anticipo, verrà chiesto loro di provare ad individuare il codice per riempire l'area dei quadrilateri, cosicché possa essere verificata la fascia di apprendimento a cui appartengono. In questo modo saranno verificate le reali acquisizioni degli alunni e potranno essere guidati, inoltre, ad usufruire in modo più completo e ricco di questo strumento. Al termine della lezione sarà richiesto di disegnare e pensare al codice di un disegno da riprodurre la volta successiva.

Bisognerà sottolineare il fatto che questo disegno, che sarà come una specie di logo, sia semplice e materialmente realizzabile dal bambino, quindi verranno invitati a non ideare prodotti eccessivamente complessi da riprodurre.

L'ultimo incontro sarà strutturato in due momenti: inizialmente i bambini si cimenteranno nella riproduzione del disegno pensato a casa, quindi attueranno strategie di *testing* e di *debugging* per ottenere il risultato desiderato. Dopo questa fase, la docente salverà il loro disegno su un dispositivo mobile per portarlo a stampare a colori ed incollarlo su una pagina del quaderno. In questo modo il bambino potrà scrivere, in un secondo momento, il codice utilizzato per ottenere il risultato che ha stampato.

È doveroso sottolineare il fatto che questa operazione era stata attuata anche all'interno del progetto ideato, ma il dispositivo mobile che è stato utilizzato ha eliminato gran parte dei file dei bambini, rendendo impossibile l'attuazione di tale operazione. Non è stata riproposta l'attività per non demotivare e demoralizzare gli alunni: sarebbero stati sicuramente dispiaciuti di un simile incidente e il clima che si sarebbe andato ad instaurare non sarebbe stato favorevole.

Oltre alle modifiche di questa proposta didattica, si deve sottolineare un ulteriore punto di sviluppo che dovrebbe essere inserito in questo progetto, ovvero la creazione di situazioni e storie legate prima all'ape *Bee*, poi a *Turtly* la tartaruga.

Grazie alla narrazione, infatti, si stimolerà maggiormente la fantasia dei bambini, quindi i livelli di motivazione e di coinvolgimento saliranno, con un conseguente aumento di partecipazione degli alunni nelle varie attività.

Questo progetto, inoltre, può essere esteso sia alle classi e al ciclo precedenti, sia a quelle successive. Il linguaggio *Logo* e il pensiero computazionale, infatti, può essere stimolato e sviluppato sin dall'ultimo anno della scuola dell'infanzia poiché i bambini, senza presentare computer o termini in inglese, sono in grado di riconoscere e sviluppare percorsi motori con o senza le *BeeBot*.

Anche durante il primo anno di scuola primaria, inoltre, sarà possibile implementare le conoscenze e le abilità legate all'apprendimento dei comandi topologici. I percorsi, infatti, possono già sfruttati per dare un primo approccio al *coding*, senza dover inserire, necessariamente, i termini in inglese.

Dalla terza classe fino alla quinta, inoltre, possono essere presentati nuove nozioni e capacità tramite l'uso del *software* in linguaggio *Logo* che superi la programmazione a blocchi e si diriga verso una testuale. Già da questa classe, infatti, è possibile presentare e richiedere agli alunni parole scritte in lingua straniera: in questo modo è possibile usare un'altra tipologia di *software* che utilizza il linguaggio *Logo* in una forma *text-based*, ovvero *LibreLogo*.

Come già accennato nel secondo capitolo, questo programma è un *plugin* della famosa *suite* da ufficio *LibreOffice*, tramite cui è possibile disegnare delle figure direttamente sul foglio di testo. I comandi sono analoghi a quelli di *TurtleArt*, quindi non sarebbe un salto mentale eccessivo per gli alunni che si troverebbero nelle condizioni di doverlo usare. Questo *software*, inoltre, permette di scrivere codici più complessi, realizzando disegni più articolati.

La scoperta del linguaggio *Logo* permette lo sviluppo di abilità matematiche e di *problem solving* e di *problem finding* fondamentali, che forniscono l'accesso ad ulteriori nozioni e abilità. A partire dalla classe terza, infatti, si potrà presentare il

concetto di angolo tramite questo programma, esplorando le possibilità a cui si può accedere. Sarà possibile, inoltre, concepire anche il concetto di rotazione poiché la tartaruga, muovendosi su sé stessa, compie un movimento rotatorio.

Questa nozione sarà fondamentale per la classe quarta: si potrà esplorare il concetto di poligono regolare poiché queste figure sono realizzate grazie alla rotazione della tartaruga. Per fare un pentagono, ad esempio, bisogna dividere l'angolo giro in cinque parti poiché è regolare e chiusa come figura: se la tartaruga non compiesse il giusto angolo di rotazione su sé stessa, non sarebbe possibile realizzare questo poligono.

Per il quinto anno potrebbero essere realizzati codici che permettano di ricavare perimetro e area delle figure geometriche: con la funzione IF e PRINT sarà possibile, infatti, ottenere queste due richieste immettendo semplicemente i dati. In questo modo i bambini non solo saranno stimolati all'apprendimento delle formule dell'area e del perimetro, ma potranno anche imparare nuove funzioni del programma che saranno utili negli anni successivi.

L'apprendimento per scoperta, quindi, permette di concepire l'errore in modo non convenzionale, accedendo a nozioni più profonde e sviluppabili in un'ottica verticale: grazie all'uso di un *software* di programmazione *text-based* sarà possibile acquisire a nuove competenze e possibilità di *coding* più vaste ed articolate.

Il *coding* è un veicolo per trasmettere e sviluppare abilità fluide e non mnemoniche, come può essere il pensiero computazionale: i bambini, come afferma lo stesso Papert (1980), sono capaci di porsi domande euristiche su come si possa risolvere un determinato problema, sviluppando autonomamente strategie efficaci di *problem solving*. Grazie a questa capacità vengono coinvolte anche strategie di *problem finding*: porsi domande aiuta a comprendere quali siano le strade da percorrere per ottenere i risultati sperati.

Il nostro mondo è in continua evoluzione e, come abbiamo già detto nel primo

paragrafo, non tutti hanno il medesimo accesso alle tecnologie, ma ognuno di noi ha l'esigenza di acquisire una *digital literacy* di base per poter essere cittadino attivo nel mondo. Ad oggi, inoltre, troppi alunni ancora restano esclusi dal mondo dell'apprendimento della matematica, sia a causa di una cattiva didattica della stessa sia per un sempre crescente numero di alunni con deficit di apprendimento.

A mio avviso questo ambiente di programmazione, veicolando apprendimenti di matematica senza doverla consapevolmente metterla in atto, è perfetto per tutte le tipologie di studenti: *Logo* non ha fallito, ma ha anticipato di svariati decenni una tipologia di didattica applicabile oggi con maggiore facilità. Questa didattica, quindi, deve essere sfruttata con consapevolezza affinché tutti gli alunni, specialmente quelli con maggiori fragilità, siano capaci di accedere agli apprendimenti, soprattutto in ambito matematico: la percezione dell'errore in una nuova ottica permetterebbe ai bambini di accedere ai concetti più profondi, creando un ricordo positivo legato alla matematica.

Questa didattica, in definitiva, porterebbe tutti ad accedere con maggior successo agli apprendimenti in campo matematico, limitando la frustrazione e permettendo anche a chi ha difficoltà di raggiungere gli obiettivi prefissati. Una nuova concezione dell'errore quindi, che permette a chiunque di apprendere con la stessa facilità degli altri e in un contesto più mite, che non crei situazioni di disagio o di frustrazione in tutti gli alunni. Il metodo descritto in questo lavoro è stato sviluppato in modo che ogni bambino, anche quello con più difficoltà, possa accedere al meraviglioso mondo della *Mathland* senza subire una trasmissione non solo passiva, ma anche improduttiva dei saperi.

BIBLIOGRAFIA

Aivaloglou E., Hermans F. (2016). *How Kids Code and How We Know: An Exploratory Study on the Scratch Repository*. Proceeding ICER '16 Proceedings of the 2016 ACM Conference on International Computing Education Research. Pagine 53-61.

Aivaloglou E., Hermans F. (2016A). *Do code smells hamper novice programming*. IEEE 24th International Conference on Program Comprehension (ICPC)

Ausubel, D.P. (1960). *The use of advance organizers in the learning and retention of meaningful verbal material*. Journal of Educational Psychology, 51, 267-272.

Barnes T., Price T. W. (2015). *Comparing textual and block interfaces in a novice programming environment*. Proceeding ICER '15 Proceedings of the eleventh annual International Conference on International Computing Education Research. Pagine 91-99.

Bocconi S. et al (2016). *Developing computational thinking in compulsory education* – ED. P. Kampylis e Y. Punie Science for Policy report by the Joint Research Centre (JRC), the European Commission's science and knowledge service.

Calvani A. (2001). *Educazione, comunicazione e nuovi media. Sfide pedagogiche e cyberspazio*, Torino: UTET libreria.

Calvani A. (2011). *Principi dell'istruzione e strategie per insegnare. Criteri per una didattica efficace*, Roma: Carrocci.

Calvani A., Fini A., Ranieri M. (2010). *La competenza digitale nella scuola. Modelli e strumenti per valutarla e svilupparla*. Trento: Erickson.

Calvani A., Fini A., Ranieri M. (2011). *Valutare la competenza digitale. Prove per la scuola primaria e secondaria*. Trento: Erickson.

Calvani, A. (a cura di) (2007). *Fondamenti di didattica. Teoria e prassi dei dispositivi formativi*, Roma: Carrocci.

Cambi F. (a cura di) (2010). *Media Education tra formazione e scuola. Principi, modelli, esperienze*, Pisa: ETS.

Chambers C., Scaffidi C. (2016). *Skill progression demonstrated by users in the Scratch animation environment*. Proceedings of the Conference on Computer Human Interaction (CHI). Pagine 1-39.

CSTA, ISTE (2011). *Computational Thinking in K-12 Education leadership toolkit.*

Formiconi A. R. (Versione parziale 0.4 del 9 settembre 2016). *Piccolo manuale di LibreLogo, La Geometria della Tartaruga*, licenza Creative Commons Attribuzione 2.5 Italia.

Giannakos M. N., Jaccheri L. (2013). *What Motivates Children to Become Creators of Digital Enriched Artifacts?* Proceeding C&C '13 Proceedings of the 9th ACM Conference on Creativity & Cognition, Pagine 104-113.

Gülbahar Y., Kalelioğlu F. (2014). *The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective.* Informatics in Education, 13. Pagine 33-50.

Gülbahar Y., Kalelioğlu F., Kukul V. (2016). *A Framework for Computational Thinking Based on a Systematic Research Review*. Baltic J. Modern Computing, Vol. 4, No. 3. Pagine 583-596.

Henderson H. (2003). *A to Z Computer scientists*. New York: Facts on File Inc.

Homer M., Noble J. (2014). *Combining tiled and textual views of code*. Proceeding VISSOFT '14 Proceedings of the 2014 Second IEEE Working Conference on Software Visualization. Pagine 1-10.

Lewis C. M. (2010), *How programming environment shapes perception, learning and goals: Logo vs. Scratch.* Proceeding SIGCSE '10 Proceedings of the 41st ACM technical symposium on Computer science education. Pagine 346-350.

Lewis C. M. et al (2014). *Children's perceptions of what counts as a programming language*. J. Comput. Sci. Coll., 29(4). Pagine 123-133.

Lo Sapio G. (2010). *Introduzione alla psicologia della disabilità*. Pisa: EDIZIONI ETS.

Lo Sapio G. (2012). *Manuale sulla disabilità*. Roma: Armando Editore

Lo Sapio G. (a cura di) (2011). *Manuale di psicologia dell'educazione*. Pisa: EDIZIONI ETS.

Mannila et al (2014). *Computational thinking in K-9 education.* In *Proceedings of the Working Group Reports of the 2014 on Innovation &Technology in Computer Science Education Conference*, ItiCSE-WGR 2014, 1-29, ACM, New York.

Matias J. N. et al (2016). *Skill Progression in Scratch Revisited*. Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, pagine 1486-1490.

Papert, S. A. (1960). *The lattices of logic and topology (PhD thesis)*. University of the Cambridge.

Papert, S. A.(1980). *Mindstorms. Children, Computers and Powerful Ideas*

Pieri M., Ranieri M. (2014). *Mobile learning. Dimensioni teoriche, modelli didattici, scenari applicativi*, Milano: UNICOPLI.

Postman, N. (1984). *La scomparsa dell'infanzia. Ecologia dell'età della vita*,

Ranieri M. (2011). *Le insidie dell'ovvio. Tecnologie educative e critica della retorica tecnocentrica*, Pisa: ETS.

Schaffer H. R. (2008). *I concetti fondamentali della psicologia dello sviluppo*, Milano: Raffaello Cortina Editore.

Sherin B. L. (2001). *A comparison of programming languages and algebraic*

notation as expressive languages for physics. Int. Journal of Computers for Mathematical Learning, 6. Pagine 1-61.

Weintrop D. et al (2016) *Defining Computational Thinking for Mathematics and Science Classrooms*. J Sci Educ Technol, 25. Pagine 27–147

Weintrop D., Wilensky U. (2015). *To block or not to block, that is the question: students' perceptions of blocks-based programming*. Proceeding IDC '15 Proceedings of the 14th International Conference on Interaction Design and Children. Pagine 199-208.

Weintrop D., Wilensky U. (2015A). *Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs*. Proceeding ICER '15 Proceedings of the eleventh annual International Conference on International Computing Education Research. Pagine 101-110.

Wing J. M. (2006). *Computational thinking.* In *Communications of the ACM*, 49 (3). Pagine 33-35.

SITOGRAFIA

http://dailypapert.com/about-seymour-papert/ (consultato in data 9 aprile 2017)

http://el.media.mit.edu/logo-foundation/services/ta.html (consultato in data 29 aprile 2017)

http://lascuola.it/nuovadidattica/it/home/contenuti/1382696203499/costruzionismo (consultato in data 9 aprile 2017)

http://nuovadidattica.lascuolaconvoi.it/teorie/costruzionismo/ (consultato in data 9 aprile 2017)

http://scratch.mit.edu (consultato in data 29 aprile 2017)

http://wiki.laptop.org/go/Turtle_Art (consultato in data 29 aprile 2017)

http://www.comune.jesi.an.it/jesicentro/TDC/DISPENSE/LOGO/storia.htm (consultato in data 9 aprile 2017)

http://www.di.unipi.it/~lagana/log1.html (consultato in data 29 aprile 2017)

http://www.indire.it/progetto/clil-content-and-language-integrated-learning (consultato in data 29 aprile 2017)

http://www.papert.org/ (consultato in data 9 aprile 2017)

http://www.python.it/ (consultabile in data 29 aprile 2017)

http://www.scuolaetecnologia.it/wp-content/uploads/2016/03/MIUR.AOODGEFID_0005403.16-03-2016_16-03-14_Avviso_atelier_creativi.pdf (consultato in data 13 maggio 2017)

http://www.telegraph.co.uk/obituaries/2016/08/11/seymour-papert-artificial-intelligence-guru--obituary/ (consultato in data 30 aprile 2017)

https://labuonascuola.gov.it/area/m/5113/ (consultato in data 9 aprile 2017)

https://turtleart.org/ (consultato in data 27 aprile 2017)

https://www.theguardian.com/education/2016/aug/03/seymour-papert-obituary
(consultato in data 30 aprile 2017)

NORMATIVA CONSULTATA

Indicazioni nazionali per il curricolo nella scuola dell'infanzia e del primo ciclo di istruzione, 2012 consultabile dal sito: http://www.indicazioninazionali.it/documenti_Indicazioni_nazionali/indicazioni_nazionali_infanzia_primo_ciclo.pdf (consultato in data 9 aprile 2017)

Legge n. 107 del 13 luglio 2015, riforma del sistema nazionale di istruzione e formazione e delega per il riordino delle disposizioni legislative vigenti, consultabile dal sito: http://www.istruzione.it/scuola_digitale/allegati/Materiali/pnsd-layout-30.10-WEB.pdf (consultato in data 12 maggio 2017)

Legge n. 170 del 8 ottobre 2010, nuove norme in materia di disturbi specifici di apprendimento in ambito scolastico, consultabile dal sito: http://www.gazzettaufficiale.it/gunewsletter/dettaglio.jsp?service=1&datagu=2010-10-18&task=dettaglio&numgu=244&redaz=010G0192&tmstp=1288002517919 (consultato in data 29 aprile 2017)

Legge-Quadro n. 104 del 5 febbraio 1992 per l'assistenza, l'integrazione sociale e i diritti delle persone handicappate, consultabile dal sito: http://www.gazzettaufficiale.it/eli/id/1992/02/17/092G0108/sg (consultato in data 29 aprile 2017)

PON (Programma Operativo Nazionale) 2014-2020 consultabile al sito: http://www.istruzione.it/allegati/2014/PON_14-20.pdf (consultato in data 9 aprile 2017)