ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

PENSIERO COMPUTAZIONALE E CODING NELLA SCUOLA PRIMARIA: IL PROGETTO COGITO

Elaborato in
PROGRAMMAZIONE AD OGGETTI

Relatore
Prof. ALESSANDRO RICCI

Co-relatore
Dott. ANGELO CROATTI

Presentata da
LAURA TARSITANO

Anno Accademico 2016 – 2017
II Sessione di Laurea

PAROLE CHIAVE

Pensiero Computazionale

Costruzionismo

Seymour Papert

Coding

Progetto COGITO

Ai miei genitori, a Nicoletta.
Il mio posto sicuro nel mondo

Indice

Abstract

Lo studio del pensiero computazionale è prezioso per ogni studente e non solo per gli appassionati e programmatori di mestiere. Così come non tutti imparano a scrivere per diventare scrittori, né imparano a contare per diventare matematici, l'insegnamento dei principi dell'informatica e del pensiero computazionale non obbliga a diventare informatici. Questi si possono quindi applicare alle altre discipline e ai comportamenti quotidiani.

Il computer diventa lo strumento primario con cui applicare questo cambiamento, viene visto come mezzo di costruzione e modellazione e non più soltanto come mezzo per gestire informazioni. Il maggior promotore di questo fu Seymour Papert attraverso la sua scuola costruzionista, che si basa sul principio che la conoscenza non è solo trasmessa ma è frutto di esperienza e costruzioni. L'errore non viene più visto come un qualcosa di negativo, ma anzi come uno strumento di crescita e di costruzione della propria conoscenza.

Su questi punti, grazie anche alla creazione del linguaggio di programmazione *Scratch*, utilizzato da milioni di utenti in tutto il mondo, continua oggi il lavoro di Mitch Resnick e del gruppo di ricerca *Lifelong Kindergarten* presso il MIT. Grazie anche alle iniziative di movimenti e correnti nazionali e internazionali il coding e il pensiero computazionale stanno entrando a pieno regime nelle scuola di tutto il mondo, in alcuni paesi Europei è già stato introdotto come materia all'interno del curriculm scolastico.

Il progetto COGITO sperimenta l'introduzione di queste competenze digitali all'interno del Terzo Circolo Didattico di Cesena, grazie alla collaborazione di docenti e insegnanti si cerca di trasmettere non solo i principi del pensiero computazionale ma anche come questi ultimi possano essere integrati con le discipline scolastiche. Il progetto è valutato in termini di commenti e valutazioni da parte delle insegnanti.

Introduzione

Lo studio della *computer science*, pensiero computazionale e programmazione è prezioso per ogni studente e non solo per gli appassionati e programmatori di mestiere. Diversi pionieri (Seymour Papert, Alan Key, etc.) hanno proposto *framework* concettuali e linguaggi in grado di rendere più efficace ed efficiente il processo di apprendimento di tali tematiche: l'insegnamento dell'informatica non consiste solo nel formare migliori sviluppatori software, è invece necessario introdurre nuovi percorsi di educazione all'informatica. Così come non tutti imparano a scrivere per diventare poeti, né tanto meno imparano a contare per diventare matematici, approcciarsi ai concetti di computazione e programmazione non sottintende di diventare programmatori di mestiere.

Nel 2006, Jeanette Wing definisce il pensiero computazionale come: "il processo necessario per la formulazione e soluzione di problemi in forme comprensibili da agenti in grado di processare informazioni". Il pensiero computazionale si propone di applicare le nozioni acquisite in discipline eterogenee (arte, musica, etc.) e nella quotidianità. A tale scopo il computer diventa un mezzo canalizzatore delle conoscenze dello studente con il fine di migliorarne e arricchirne l'apprendimento. Così come la macchina da scrivere ha preso piede a seguito dell'alfabetizzazione di scrittori e lettori, la diffusione di questo mezzo richiede l'alfabetizzazione computazionale (*computational literacy*)

Secondo il costruzionismo di Seymour Papert il computer costituisce un nuovo mezzo di apprendimento tramite cui gli studenti diventano creatori del proprio processo di crescita: il ruolo del computer è come quello della creta con cui costruire una scultura. Il computer non è solo un macchina con cui gestire informazioni, ma uno strumento per modellare, costruire, apprendere e scoprire. Il linguaggio LOGO creato e ideato dallo stesso Papert permette di applicare questi concetti costruendo, manipolando, sbagliando, ricostruendo e testando. L'errore non è più visto in ottica negativa, ma come un aspetto costruttivo del processo di apprendimento. Sbagliare significa esplorare, studiare il malfunzionamento di un programma e individuare soluzioni alternative al problema.

Questi sono solo alcuni dei punti cardine della scuola costruzionista di Papert, principi su cui, ad oggi, si basano i lavori di Mitch Resnick e del gruppo di

ricerca *Lifelong Kindergarten* presso il *Massachusetts Institute of Technology (MIT)*. Lifelong Kindergarten è il gruppo fondatore del linguaggio Scratch utilizzato oggi non solo da milioni di utenti in tutto il mondo, ma anche nelle scuole di ogni ordine e grado per introdurre i concetti di pensiero computazionale e coding.

Correnti internazionali e nazionali si stanno muovendo per introdurre questi concetti all'interno dei sistemi scolastici. Alcuni paesi europei hanno già introdotto il *coding* nel proprio curriculm scolastico, anche l'Italia ha riconosciuto l'importanza dell'alfabetizzazione digitale nella riforma della *Buona Scuola*. In questo contesto nazionale, il progetto COGITO sperimenta l'introduzione delle competenze digitali nel Terzo Circolo Didattico di Cesena. Lo scopo non è solo quello di trasmettere i concetti base dell'informatica ("learn to code"), ma anche e soprattutto quello di mettere a regime questi ultimi con il pensiero pedagogico, uno strumento utile per l'apprendimento di qualsiasi disciplina scolastica ("code to learn").

Il Capitolo 1, dopo aver illustrato alcune delle molteplici definizioni di pensiero computazionale, presenta la figura di Seymour Papert e del suo costruzionismo e continua con la figura di Mitch Resnick, suo successore e creatore dell'ambiente Scratch. Il Capitolo 2 si concentra sull'importanza del pensiero computazionale all'interno delle scuole primarie presentando le organizzazioni e i movimenti nazionali e internazionali che stanno già attuando azioni per fare in modo che questo accada. Il Capitolo 3 entra nel vivo del progetto COGITO descrivendo il contributo sperimentale di questa tesi che mette in sinergia il pensiero computazionale con le altre discipline scolastiche prendendo spunto dalle teorie di Seymour Papert. Il Capitolo 4 presenta la valutazione del progetto sulla base dell'esperienza degli insegnanti che vi hanno preso parte. Infine, sono mostrate le conclusioni e gli sviluppi futuri dei contributi proposti.

Capitolo 1

Pensiero computazionale e coding

Negli ultimi anni abbiamo assistito a un fenomeno alquanto simpatico. Le nostre teste (soprattutto quelle di chi è all'interno del sistema scolastico, universitario e di chi ha figli) sono state bombardate da notizie, articoli, progetti e iniziative sul pensiero computazionale, sul coding e su come questo debba essere inserito nella scuola. Ma cos'è il pensiero computazione?

Quando si vuole conoscere il significato di qualcosa l'abitudine è quella di andare a cercare il suo significato su internet. Se si prova a digitare "pensiero computazionale" sul nostro motore di ricerca verremmo inondati da pagine web piene di termini quali "coding", "programmare", "computational thinking", "la buona scuola", "Programma il Futuro" e tanto tanto altro ancora.

Proviamo a fare un po' di ordine e a capire cos'è il pensiero computazionale e a cosa effettivamente può servire, a noi, alla scuola e alla società intera.

1.1 Cos'è il pensiero computazionale? - Definizioni

Il primo a parlare di pensiero computazionale fu **Seymour Papert** nel 1980 nel suo libro *Mindstorms* [1] e nel 1996 parlando del linguaggio LOGO, linguaggio sviluppato da lui stesso per insegnare la programmazione ai bambini.

A portare di nuovo all'attenzione della società il concetto di pensiero computazionale fu **Jeannette Wing** in un articolo del 2006 *Computational Thinking* [2] la cui definizione è quella oggi maggiormente utilizzata, ovvero il pensiero computazionale è quel *"processo mentale che sta alla base della formulazione dei problemi e delle loro soluzioni così che le soluzioni siano rappre-*

sentate in una forma che può essere implementata in maniera efficace da un elaboratore di informazioni sia esso umano o artificiale"

Computing At School nella guida per gli insegnanti [3, p. 6] definisce il pensiero computazionale come un *"processo cognitivo e di pensiero che coinvolge logiche e ragionamenti attraverso i quali i problemi sono risolti e gli artefatti, le procedure e i sistemi compresi." Le abilità di riflessione computazionale si "riferiscono a capacità di pensare e risolvere problemi in tutto il curriculum e nella vita in generale. Il pensiero computazionale può essere applicato a una vasta gamma di manufatti."*

Bisogna individuare sin dal principio concetti, pratiche e prospettive che possono essere oggetto di insegnamento/apprendimento, studio e valutazione come suggerito in [4, pp. 26,27] Si parte dai **concetti** che sono alla base dei linguaggi di informatica:

- Sequenza: il risultato finale è influenzato dall'ordine in cui vengono scritte le istruzioni
- Condizione: il programma in base al verificarsi di possibili casi deve prendere delle decisioni
- Ripetizione: un programma deve poter eseguire le stesse istruzioni per un certo numero di volte, magari non conosciuto a priori
- Evento: un programma può far partire certe istruzioni solo nel momento in cui accade qualcosa
- Parallelismo: più istruzioni possono essere eseguite contemporaneamente sapendo gestire le complicazioni che questo genera
- Operatori: a volte è necessario scrivere espressioni logiche e matematiche che verranno poi risolte dal computer
- Dati: analizzare, memorizzare e rappresentare informazioni

Si passa poi alle **pratiche**: "modi di lavorare, pensare e approcciare i problemi"

- Incrementali e iterativi: è molto più semplice lavorare in modo incrementale su un progetto, testandolo e modulandolo, piuttosto che tentare di realizzare un progetto perfetto al primo colpo;
- Testing e debugging: è impossibile determinare se un programma informatico funzioni al primo colpo. Per questo è utile testarlo, provarlo alla ricerca di eventuali errori e cercare soluzioni per correggerli;

- Riuso e mixing: è molto più comodo riutilizzare del codice già scritto o scritto da altri piuttosto che scrivere ogni volta un programma da zero.
- Attenzione all'efficienza, calcolabilità e complessità: "programmando si impara a individuare una strategia di soluzione che porta a un risultato[...], sperabilmente il migliore e usando meno risorse possibili." [4, pag.28]

Metodi [3, pp. 7,8]

- Decomposizione: *capacità di scomporre un problema in sotto-problemi più semplici, risolvere questi separatamente e trovare la soluzione complessiva unendo i risultati parziali.* Cruciale per affrontare la complessità di un problema. Le situazioni nuove (i sotto-problemi) si comprendono meglio e i sistemi di grandi dimensioni sono anche più facili da progettare
- Astrazione: *capacità di ridurre la complessità nascondendo dettagli irrilevanti e focalizzando sugli elementi principali per il problema che si deve risolvere o sistema da costruire.* Rende più comprensibile l'artefatto riducendo i dettagli inutili. E' la capacità di nascondere informazioni rendendo il problema più semplice, mantenendo comunque quelle più importanti
- Generalizzazione: *capacità di adattare la soluzione identificata per un problema specifico per risolvere tutti i problemi relativi ad una classe/insieme/categoria* Identifica modelli, somiglianze e connessioni. Grazie a questi si possono risolvere nuovi problemi sfruttando le soluzioni precedenti. Gli algoritmi che risolvono alcuni tipi di problema possono essere utilizzati per risolvere problemi dell'intera classe. Ogni volta che si incontra un problema di quella classe si può applicare la soluzione generale

In [4, pp. 28,29] sono illustrate delle nuove **prospettive** che si aprono utilizzando il pensiero computazionale, cioè nuovi modi di vedere il mondo e sé stessi:

- Esprimere se stessi
- Connettersi
- Farsi domande
- Saper gestire la complessità e i problemi difficili

Un informatico per risolvere un problema combina tutte queste azioni per arrivare ad una soluzione. Queste abilità possono anche essere utilizzate per risolvere problemi in qualsiasi ambito della vita, non solo con il calcolatore.

1.1.1 Programmazione e Coding

La programmazione è quell'attività che parte da un problema per arrivare alla soluzione tramite un programma eseguibile da un computer. L'informatico, nel processo di risoluzione di un problema deve quindi analizzarlo, trovare un algoritmo di risoluzione, verificare che questo sia corretto attraverso dei test, eventualmente sistemarlo e poi implementarlo, tradurlo, in un linguaggio di programmazione comprensibile dal calcolatore. All'inizio si racchiudeva tutto questo processo nel termine *programmazione*. Man mano che l'informatica si è evoluta è stato necessario fare una distinzione tra *programming* e *coding*. Con il primo termine si intende tutto il processo descritto sopra, con il termine *coding* invece, soltanto l'effettiva azione di scrivere codice.

Negli ultimi anni, il termine *coding* è stato riportato alla luce grazie allo sforzo di voler portare l'insegnamento del pensiero computazionale all'interno della scuola.

1.2 La Scuola Costruzionista di Seymour Papert

Seymour Papert, matematico, filosofo, pedagogista e informatico è conosciuto principalmente per essere il padre del *costruzionismo*, una declinazione del costruttivismo, e per aver sviluppato il linguaggio LOGO, linguaggio di programmazione a scopo didattico come supporto all'apprendimento.

Seymour Papert nasce a Pretoria, in Sudafrica nel 1928, dopo aver studiato matematica si trasferisce nel 1954 a Cambridge per svolgere attività di ricerca sempre in ambito matematico. Dopo molto girare si trasferisce negli USA nel 1967 per fermarsi al MIT e fondare un laboratorio di intelligenza Artificiale con Marvin Minky. Ha lavorato a teorie dell'apprendimento avvicinandosi al pensiero di Jean Piaget in un periodo di studio a Ginevra. Proprio dal costruttivismo di Piaget, Papert elabora il "costruzionismo" una teoria dell'apprendimento basata su un approccio multidisciplinare.

1.2.1 Il ruolo del computer

La visione costruzionista vede la scuola come luogo di costruzione e non di trasmissione della conoscenza. Con Papert il costruttivismo si affianca alle tecnologie e agli strumenti informatici. Il computer può diventare un ottimo strumento con cui fare scuola, diventando un vero e proprio strumento di apprendimento permettendo agli studenti di esplorare nuovi modelli, costruire le proprie conoscenze attivamente. Il controllo è di chi apprende. [5, p. 47]

C'è un passaggio da una visione dell'informatica come materia da insegnare e computer come strumento per insegnare a una visione in cui l'informatica è comunicazione, linguaggio di moderazione di sistemi complessi. Utilizzare il computer come strumento di apprendimento vuol dire avere come obiettivo quello di un "apprendimento attivo", secondo il famoso slogan di Papert "dovrebbe essere il bambino a programmare il computer e non il computer a programmare il bambino."[..] "I bambini devono cambiare il loro status di 'consumatori' di informazioni in quello di 'produttori' di conoscenza." Il senso di un computer in un ambiente di apprendimento è quello di laboratorio personale dove compiere esperienze significative.
Il termine costruttivismo si riconduce agli studi di Jean Piaget, secondo cui le conoscenze non possono essere semplicemente trasmesse da un individuo ad un altro. Il costruzionismo ha in più la connotazione di "set da costruzioni", da intendersi come il set dei LEGO.
Il costruzionismo si basa sul fatto che i bambini devono scoprire da soli le conoscenze di cui hanno bisogno.

Papert considera sbagliato il modello istruzionista, che considera la mente del bambino come un vaso da riempire. Nota invece, come sia buona prassi istruire le persone in base al lavoro che devono svolgere, il modello costruzionista si pone in quest'ottica e le attività dei bambini sono apprendere, pensare e così via; eppure non si fa un minimo accenno a queste cose, si parla piuttosto di numeri, grammatica e rivoluzione francese sperando che le cose importanti emergano da sole. "Rimane il paradosso: perché non insegniamo loro a pensare, imparare e giocare¿' Il computer non è quindi solo un mezzo tecnologico ma impatta sul modo con cui può avvenire l'apprendimento, è lo strumento per abbattere il muro tra cultura umanistica e scientifica e costruire un ambiente personale. "Un ambiente come quello del computer può invece restituire al bambino la possibilità di apprendere spontaneamente e di acquistare fiducia e consapevolezza delle proprie possibilità". Il computer non deve stare nel laboratorio informatico, il computer deve stare sul banco, deve diventare da poter manipolare per conoscere, scoprire e costruire.
"Il ruolo del computer è quello della creta con cui costruire una scultura".[10] Non è una macchina di informazioni o per gestire informazioni, è una macchina per eseguire i progetti, per costruire. Rende possibile azioni didattiche non possibili da strumenti tradizionali, come esplorare le strutture di un atomo, simulare effetti clima e molte altre; rende concreto e personale il formale.

C'è quindi un passaggio da una cultura del calcolo a una della modellazione e simulazione.

La conoscenza è vista come prodotto di costruzione attiva da parte del soggetto. La conoscenza non è trasmessa ma costruita con chi apprende nella propria mente, dell'interazione fra l'esperienza che sta vivendo ed esperienze

precedenti.

1.2.2 Bricolage

Papert sostiene che esistono due tipi di apprendimento che influenzano l'approccio ai problemi e il modo di lavorare con il computer. Distingue fra coloro che hanno l'abitudine di pianificare un piano di lavoro, i (*planners*) e coloro che invece procedono per tentativi ed errori (*bricoleurs*).Il modo di apprendere del *bricoleur*, che Papert definisce anche *tinkerer* è molto simile a quello dei bambini ed è costituito da un avvicinarsi gradualmente alla meta procedendo per gradi. Marvin Minsky, collaboratore di Papert nel laboratorio di Intelligenza Artificiale di Cambridge enuncia il cosiddetto *principio di Papert:* "alcuni fra gli stadi più cruciali dello sviluppo mentale sono basati non sulla semplice acquisizione di nuove abilità,bensì sull'acquisizione di nuovi metodi amministrativi per usare ciò che già si conosce[...] poiché una mente non può crescere molto se si limita soltanto ad accumulare conoscenze, ma deve inventarsi anche i modi per poter sfruttare al meglio le conoscenze già possedute"[6]. Papert sostiene che i principi del *bricolage* sono: usare le cose di cui si dispone, improvvisare, adattarsi.

1.2.3 LOGO

Il linguaggio LOGO, ideato da Papert a scopi didattici è il "linguaggio del costruttivismo", creato circa trent'anni fa per "affidare il comando delle operazioni" all'allievo si avvicina molto alla teoria del *bricolage*. Si possono utilizzare sia i comandi messi a disposizione dal linguaggio stesso, sia quelli creati e personalizzati dall'utente: "in ciò si può intravedere, accanto all'espandibilità dell'ambiente, un bricoleur che, oltre ad usare gli arnesi già messi a disposizione, utilizza quelli che via via ha creato"[5, p. 60] Programmare in LOGO consente di provare, testare, ricostruire quello che non va e riprovarlo, consentendo di imparare a sistemare un ragionamento piuttosto che scartarlo tutto in blocco come "sbagliato". È, come lo chiamerebbe Papert, un "apprendimento sporco" indicando quell'apprendimento che avviene senza seguire un percorso strutturato o costruito in anticipo, ma provando, testando, scoprendo strade man mano che ci si avvicina alla soluzione.

Programmare in LOGO è simile all'attività del bricoleur, in quanto piuttosto che scrivere un'applicazione che "gira al primo colpo" viene testata e scoperta man mano che si aggiungono istruzioni. Se non funziona, parte il meccanismo di ricerca e individuazione dell'errore per risolverlo: (*debugging*). Si possono quindi avere delle soluzioni che si costruiscono incrementalmente, con l'idea di unire i risultati parziali per giungere all'obiettivo finale. Nel-

la stesura di un'applicazione LOGO, "non ci si aspetta che tutto funzioni al primo colpo. Non si giudica con i soliti "esatto, hai preso un buon voto" e "sbagliato, hai preso un brutto voto." Invece ci si pone la domanda: "come posso aggiustarlo?"

1.2.4 L'importanza dell'errore

Quando si commette un errore questo può essere letto anche in chiave positiva, infatti l'errore permette di comprendere un determinato passaggio in modo migliore e quindi di capirlo meglio e superarlo. L'errore è molto utile per arrivare alla soluzione, anche proprio dal punto di vista della cooperazione. Si può chiedere al compagno o all'amico il perché di quell'errore e confrontandosi risolverlo.

Papert nel percorso che lo ha portato a realizzare LOGO getta le basi per la comprensione del concetto di errore. Secondo Papert, la distinzione giusto/-sbagliato è controproducente infatti molti bambini sono bloccati nel processo dell'apprendimento perché hanno un modello in cui o "si è capito" o "non si è capito". Il bambino davanti a un errore tende a rimuoverlo e a dimenticarlo, mentre la correzione degli errori fa parte del processo di comprensione del programma; il bambino che programma è incoraggiato a studiare il *bug*, non a cancellarlo in fretta dalla sua memoria.

L'errore era considerato come un elemento negativo, un inciampo nella strada che porta verso la conclusione con successo dell'unità didattica in questione; commettere un errore significava non poter andare avanti nella lettura del testo o della verifica di ciò che doveva essere appreso e costringeva necessariamente a guardarsi gli argomenti, a tornare indietro e a ripetere il percorso fornendo, spesso a caso, una risposta diversa da quella, sbagliata, in precedenza.[1] Mettere in evidenza l'errore come un aspetto negativo produce, è stato detto, produzione di energia e di interesse nel bambino, che ha bisogno di non venire rimproverato per l'errore commesso, ma va piuttosto messo nella condizione di controllare l'errore e di stabilire con lui un rapporto, in un certo senso,amichevole.[7]

In LOGO, l'errore è una parte costitutiva del processo di apprendimento e di costruzione della conoscenza; l'errore, di solito descritto come qualcosa di negativo, in logo diventa opportunità di crescita e apprendimento. Gli errori aiutano perché guidano la scoperta di ciò che accaduto, cosa non è andato e attraverso la comprensione di questo si riesce a sistemare. Più di qualunque altra, l'esperienza della programmazione aiuta in questa filosofia.
Nell'ambiente LOGO, i bambini imparano che anche l'insegnante è un allievo e ciascuno di noi apprende dagli errori [1]. Una caratterista del lavoro col calcolatore è che si possono presentare delle situazioni nuove sia al bambino

che all'insegnante dove quindi l'insegnante non deve fare finta di non sapere. Condividendo il problema e cercando insieme la soluzione permettere di far imparare al bambino che non deve fare quello che "il maestro dice", ma quello che "il maestro fa". Un ambiente scolastico costruttivo è un ambiente nel quale l'insegnante apprende insieme al bambino. L'alunno attraverso l'errore può valutare da solo i propri progressi. A livello operativo l'impatto è che gli studenti vorrebbero affrontare il problema tutto in una volta. La scomposizione del problema invece, permette di facilitare la risoluzione del problema e anche l'identificazione e risoluzione di eventuali errori

1.2.5 Costruzionismo

Papert, riconosce una sorta di debito intellettuale verso Piaget, con il quale ha lavorato per quattro anni a Ginevra e verso le teorie pedagogiche di Maria Montessori: entrambi lo hanno fatto riflettere sul pensiero dei bambini e sul rispetto che occorre avere nei confronti dei loro errori e del loro modo di pensare.[5, p. 73] Papert insiste su alcuni aspetti del costruzionismo, che possono essere così riassunti sotto forma di principi:[8]

- Il protagonismo dello studente, attraverso il quale viene incoraggiato lo sviluppo delle abilità cognitive e metacognitive dell'allievo e non viene insegnato alcunché
- L'"inversione epistemologica" - che sostituisce l'"imparare per usare" con l'"usare per imparare"
- La rivalutazione del pensiero operatorio concreto su quello formale logico-deduttivo. sono le esperienze concrete e casuali che, secondo Papert, hanno originato anche le teorizzazioni di tipo scientifico.
- L'apprendimento "sintonico"
- I "micromondi" sono "palestre cognitive", nelle quali si ricercano incessantemente problemi e soluzioni emergenti attraverso il *problem finding* e il *problem solving*
- "apprendere riflettendo sui propri errori'"

L'obiettivo di Papert quando ha pensato a LOGO, non è stato quello di formare generazioni di "programmatori di computer", ma quello utilizzare il computer come ambiente per "imparare a imparare" e la programmazione come attività per esprimere se stessi attraverso artefatti cognitivi. Il costruzionismo di Papert condivide alcuni punti dell'attivismo di Dewey (learning-by-doing) e del costruttivismo di Piaget (learning-by-making) ma, a partire da questi,

elabora una nuova filosofia educativa. Con l'attivismo di Dewey, Papert condivide la continuità tra progettazione e attualizzazione di un progetto mentre rispetto al costruttivismo di Piaget aggiunge qualcosa in più e si dice convinto che la costruzione da parte del bambino, è più efficace se non è mentale, ma se viene supportata da una parallela costruzione reale, effettiva del proprio modo di ragionare [5]. Piaget afferma che la conoscenza non viene trasmessa, ma costruita da chi apprende grazie all'interazione tra esperienza e conoscenze già possedute. Papert con il costruzionismo va oltre, "si apprende meglio quando si è coinvolti nella costruzione di un artefatto che il soggetto ritiene importante e significativo." [4, p. 24] Il bambino avverte il controllo del computer.

1.2.6 Micromondo

LOGO è il linguaggio dei cosiddetti *"micromondi"* ovvero una riproduzione, effettuata su computer, del comportamento di un sistema reale. Sono piccoli universi, realtà delimitate, luoghi "sicuri" da esplorare in cui è possibile creare applicazioni sotto forma di percorsi di apprendimento, animazioni, simulazioni geometriche etc. [9]

"Come micromondo, LOGO incoraggia un apprendimento per scoperta,non direttivo, sulla base di soluzioni di natura euristica, in un ambiente in cui tutte le idee vengono prodotte dall'allievo nel momento in cui servono, dove viene costruita un'intima relazione con l'ambiente di sviluppo e dove vengono vissute esperienze logico-intuitive in contatto non tanto con le idee dell'informatica, ma con quelle della matematica, della linguistica e, in sintesi, di un modello, intellettuale e personale, di costruzione della conoscenza e, quindi, dell'apprendimento". [5, p. 81]

La scuola quindi da ambiente che trasferisce cultura diventa, luogo di costruzione e non di trasmissione della conoscenza, valorizzando la creatività dei bambini con strumenti più vicini al loro modo di apprendere e di pensare. "Il computer è come al creta da cui è possibile costruire una scultura: è materiale per costruire"[10]

"In questo modo, l'informatica consente di mettere in luce, grazie al valore cognitivo della progettazione del software, la dimensione creativa dell'uso del computer."[5, p. 83]

1.2.7 La geometria della tartaruga

La "geometria della tartaruga", è un'area che Papert e i suoi colleghi hanno creato come un'area formale della matematica più significativa per i bambini. Nella geometria della tartaruga il computer ha un uso completamente diverso rispetto al tradizionale uso nella didattica. Viene utilizzato come mezzo espres-

sivo. Invece di porsi il problema "come insegnare la matematica scolastica", si pone come "ricostruire la matematica" o più in generale come ricostruire la conoscenza in modo tale che non sia necessario un grande sforzo per insegnare. La geometria della tartaruga si basa su tre principi: [1]

- Principio di continuità: il primo passo per comprendere un concetto è quello di integrarlo con conoscenze pregresse
- Principio di potenza: l'ambiente di apprendimento deve consentire a chi apprende di capire concetti carichi di significato, che non avrebbe mai pensato prima
- Principio di risonanza culturale: ciò che viene appreso deve avere senso all'interno dell'ambiente sociale nel quale ci troviamo.

1.3 L'eredità di Papert: Mitch Resnick e il MIT Lifelong Kindergarten

Mitchel Resnick[1] si può dire che sia il successore di Papert, un suo allievo che oggi lavora basandosi sui principi lasciatici in eredità.
Ha conseguito una laurea in fisica presso l'università di Princeton (1978) e MS e PhD in computer science presso il MIT(1988, 1992). Ha fondato il progetto Computer Clubhouse, una rete di 100 centri di apprendimento post-scolastico riservato a giovani delle comunità a basso reddito o con difficoltà. Ora è un professore della Ricerca di apprendimento presso il MIT Media Lab, sviluppa nuove tecnologie e attività per impegnare persone (soprattutto ai bambini) in esperienze creative di apprendimento.

Il suo gruppo di ricerca è il *Lifelong Kindergarten*, sviluppatore del software di programmazione Scratch utilizzata da milioni di persone in tutto il mondo.

Scratch è usato in molte scuole per introdurre i concetti di computazione ai bambini e ai giovani e secondo Resnick imparare a programmare è una parte essenziale dell'educazione. E' vero che il numero dei posti di lavoro per programmatori e computer scientist sta crescendo rapidamente, ma Resnick vede dei motivi più profondi e più ampi dell'imparare a programmare. Fa un'analogia tra lettura e scrittura. Quando si impara a leggere e a scrivere si aprono molte opportunità di imparare nuove cose. Quando si impara a leggere, allora si potrà leggere per imparare. Imparare a programmare è la stessa cosa. Se si impara a programmare, allora si potrà programmare per imparare ("learn to code, code to learn") [11]. Si impara sicuramente come funziona un

[1] `https://www.media.mit.edu/people/mres/overview/`

computer, ma questo è solo un punto di partenza. Imparando a programmare si imparano molte altre cose: Si imparano strategie di apprendimento per risolvere problemi, progettare progetti e comunicare idee. Non sono abilità utili solo per i programmatori ma per tutti. Il coding è visto come estensione della capacità di scrivere e creare. Si insegna ai bambini a scrivere non perché diventino scrittori, a fare calcoli non per diventare matematici, analogamente non si insegna il coding perché diventino informatici. Il coding permette di scrivere e creare nuovi tipi di cose. Nel Media Lab si sviluppano nuove tecnologie e strategie proprio per supportare l'"apprendimento creativo": il *Creative Learning*[2]. L'approccio si basa su quattro principi, le 4 'P'

- *Project* (Progetti): si lavora e si apprende meglio quando si lavora su progetti piuttosto che su esercizi singoli.
- *Peers* (Collaborare tra pari): Quando si collabora e si scambiano idee, si lavora meglio
- *Passion* (Passione): Se si lavora su qualcosa che piace e che appassiona si lavora meglio, ci si concentra di più e si è determinati nel portare a termine il lavoro
- *Play* (Giocare): L'apprendimento può essere giocoso: sperimentare nuove cose, testando i limiti provando e riprovando

Il Media Lab applica questi principi all'interno dei loro lavori e l'obiettivo è quello di consentire a tutti il *"Creative Learning"*

[2] `http://learn.media.mit.edu/creative-learning`

Capitolo 2

Pensiero Computazionale e Coding nella Scuola Primaria

Viviamo ormai, in una società dell'informazione. La tecnologia ci circonda e non possiamo negarlo. La conoscenza però delle scienze tecnologiche e informatiche che stanno alla base di queste tecnologie sono per la maggior parte oscure. "A scuola si impara a comprendere il mondo che ci circonda: non è pensabile non imparare anche i principi di quella scienza che sta avendo un così grande impatto sulla società" [4, p.30].

In realtà oltre a questo aspetto socio-economico c'è un aspetto sicuramente più importante (soprattutto per chi sta svolgendo questo lavoro) ovvero quello che il pensiero computazionale è utile per migliorare il modo di pensare. Alan Perlis fu uno dei primi a sostenere che gli studenti universitari dovessero imparare a programmare e conoscere i principi della programmazione per capire in "termini computazionali" le altre discipline. Un altro promotore di questa visione fu Seymour Papert, di cui si è largamente parlato nel primo capitolo.

Da un'indagine di European Schoolnet (2014) riferita ai paesi del G20, risulta che in 13 paesi europei si è introdotta o si sta per introdurre l'informatica nella scuola dell'obbligo. In 7 paesi europei, la programmazione è già diventata parte integrante del curriculum scolastico, per esempio Estonia, Grecia e Inghilterra.[1]

Riconosciuta l'importanza del pensiero computazionale, da qualche anno sono nati movimenti nazionali e internazionali che vogliono portare le attività di pensiero computazionale e coding nelle scuole primarie. L'obiettivo è quello di far avere ad ogni studente la possibilità di approcciarsi a questo mondo e imparare l'informatica, mettendo a disposizione di insegnanti e alunni dei percorsi in grado di farlo. Esempi da citare sono "code.org" negli Stati Uniti

[1] Midoro, Vittorio, ed. La scuola ai tempi del digitale. Istruzioni per costruire una scuola nuova. Franco Angeli, 2016. P.63

e la Gran Bretagna che ha addirittura inserito il coding all'interno del suo curriculum scolastico al pari quindi delle altre discipline. Anche l'Italia si sta muovendo grazie soprattutto alla Riforma della Buona Scuola dove uno dei suoi punti dice: "sviluppo delle competenze digitali degli studenti, con particolare riguardo al pensiero computazionale". Non solo le istituzioni si stanno muovendo ma esistono anche movimenti privati e volontari che propongono attività di questo tipo anche extrascolastiche.

2.1 Perché la scuola primaria

Per quanto possa essere difficile e dispendioso, far partire l'alfabetizzazione informatica fin dalle scuole primarie è sicuramente il modo migliore per iniziare. Come esposto in [12, pp. 97,98], difficile e dispendioso perché sicuramente bisogna prima di tutto trovare un accordo comune sul "cosa è l'informatica" e inoltre troppi insegnanti dovrebbero fare corsi di specializzazione sul pensiero computazionale. Sarebbe necessario introdurre un esperto di computer science in almeno ogni scuola (ipotesi comunque difficile e dispendiosa).

I vantaggi però sono molteplici: iniziando così presto i vantaggi dell'imparare il coding possono essere sfruttati per la vita nell'educazione. È il motivo per cui si inizia a scrivere e a leggere così presto e non perché sono abilità lavorative, ma perché offrono vantaggi per la vita intera. Inoltre la scuola primaria garantisce che tutti abbiano accesso all'istruzione informatica. "Acquisire il pensiero computazionale può essere utile per migliorare il nostro modo di pensare" [4, p. 31]. Per "insegnare" a un computer a risolvere un problema, bisogna aver capito il problema stesso e le sue strategie,‘ i computer, almeno per il momento, sono estremamente stupidi, è il programmatore che impartisce istruzioni sul suo comportamento e queste istruzioni devono essere estremamente dettagliate e precise.

2.1.1 Strumento di crescita ed espressione personale

Il motivo per cui ha senso parlare di computer e informatica nella scuola primaria è che è una tecnologia formidabile per l'apprendimento. Il pensiero computazionale può essere uno strumento di crescita e di espressione personale.

Esprimere sé stessi "scrivendo" la tecnologia. Saper leggere fa imparare nuove nozioni, capire concetti, saper leggere la tecnologia vuol dire subirla, viverla passivamente, ma saper anche scrivere permette di esprimere sé stessi attraverso storie, poesie, racconti di stati d'animo e saper "scrivere la tecnologia" vuol dire riuscire ad esprimersi e a raccontarsi. Essere utilizzatori attivi.

È uno strumento di crescita perché si sa che gli informatici non scrivono mai un programma che funzioni al primo colpo, servono prove, modifiche, test, correzioni di errori e ancora test prima di arrivare a un lavoro finale. È un lavoro incrementale. "Questo approccio può essere utile in molti ambiti della vita. Per esempio, per liberarsi dall'imperfezionismo [...], accettando gli errori e il fallimento come strumenti necessari per ottenere risultati" [4, p.35].

2.2 Iniziative nel mondo

Negli ultimi anni, tornando alla luce l'argomento pensiero computazionale, sono emerse diverse proposte e iniziative riguardanti questo argomento. Si possono evidenziare sostanzialmente due approcci differenti: il primo approccio è di tipo *problem solving*. Ovvero al bambino vengono proposti dei problemi da risolvere di difficoltà incrementale che deve risolvere scrivendo un programma. Vengono superate sfide sempre più difficili con concetti che vengono introdotti in modo strutturato e graduale. I linguaggi utilizzati per la maggiore sono quelli visuali, di solito formati da blocchi che devono essere collegati tra di loro per decidere il comportamento del programma.

Il secondo approccio è centrato sulla creatività. Ai ragazzi non vengono presentati problemi via via più difficili, ma viene lasciato a disposizione tutto l'ambiente di lavoro dando spazio alla loro creatività per la realizzazione di un progetto. In questo si imbattono nei concetti del pensiero computazionale e saranno motivati ad impararli perché ne avranno bisogno per far funzionare il loro programma. È un approccio meno strutturato, un approccio per scoperta e per errori. [4, pp. 88,89]

Code.org

Code.org[2] è un progetto statunitense "no profit" nato per diffondere il pensiero computazionale e il coding in tutto il mondo. È sostenuto dal governo americano e dalle maggiori aziende leader nel settore di tutto il mondo, sulla pagina web è possibile trovare video e tutorial di personaggi famosi che hanno aderito all'iniziativa.

Fornisce corsi dedicati sia agli insegnati che agli studenti dalle scuole primarie alle scuole superiori, ha un approccio orientato al problem solving, utilizzando sia linguaggi visuali che testuali. Le attività sono strutturate e graduali, bisogna superare problemi via via più difficili per andare avanti nelle attività. In alcune lezioni i ragazzi sono lasciati liberi di esprimersi avendo a disposizione tutti i blocchi e i concetti imparati durante i percorsi. Le attività sono

[2] `https://code.org/`

anche intervallate con attività *unplugged*, ovvero senza l'utilizzo del dispositivo elettronico, approfondendo i concetti su carta, o tramite altre attività che non prevedano l'utilizzo del computer.

Code.org promuove eventi anche a grande risonanza mediatica come *Hour of code*: di solito svolta durante una settimana di dicembre con lo scopo che ogni bambino durante quella settimana svolga almeno un'ora di coding.

2.2.1 Computing At School

Il Regno Unito è uno di quei paesi europei che già prevede l'informatica all'interno dei sui curriculum scolastici. Dal 2014 è stato introdotto il "computing" a sostituzione delle *ICT (Information and communications technology).* per dare ai giovani in Inghilterra le competenze fondamentali, le conoscenze computazionali di cui avranno bisogno per resto della loro vita. Come si evince da [13] questo rappresenta la continuità e il cambiamento, la sfida e l'opportunità. Da alle scuole la possibilità di riesaminare e valorizzare gli approcci attuali per fornire un curriculum ancora più emozionante e rigoroso che affronti le sfide e opportunità offerte dal mondo tecnologicamente ricco in cui viviamo. Il pensiero computazionale fornisce approfondimenti su molte aree del curriculum e influenza il lavoro all'avanguardia di un'ampia gamma di discipline.

L'obiettivo di Computing At School[3] nel Regno unito è quello di fornire supporto e orientamento a tutti coloro che sono coinvolti nell'insegnamento e nelle scuole con un significativo focus sul tema Computer Science.

2.3 Iniziative in Italia

2.3.1 La Riforma della Buona Scuola

Anche l'Italia si sta muovendo in merito alle attività di pensiero computazionale e coding all'interno della realtà scolastica. Il 16 Luglio 2015 è entrata in vigore la legge della"Buona Scuola": la riforma del sistema nazionale di istruzione.

Uno dei punti riguarda proprio il pensiero computazionale e il coding, le cosiddette competenze digitali. Lo scopo è di dare *"più spazio all'educazione ai corretti stili di vita, alla cittadinanza attiva, all'educazione ambientale, e si guarda al domani attraverso lo sviluppo delle competenze digitali degli studenti (pensiero computazionale, utilizzo critico e consapevole dei social network e*

[3] `https://www.computingatschool.org.uk/`

dei media).[4]

Riportando quanto scritto nella riforma al comma 56[5]

"Al fine di sviluppare e di migliorare le competenze digitali degli studenti e di rendere la tecnologia digitale uno strumento didattico di costruzione delle competenze in generale, il Ministero dell'istruzione, dell'università' e della ricerca adotta il Piano nazionale per la scuola digitale, in sinergia con la programmazione europea e regionale e con il Progetto strategico nazionale per la banda ultralarga".

Continua al comma 58 con:

"Il Piano nazionale per la scuola digitale persegue i seguenti obiettivi:

a *Realizzazione di attività volte allo sviluppo delle competenze digitali degli studenti, anche attraverso la collaborazione con università, associazioni, organismi del terzo settore e imprese, nel rispetto dell'obiettivo di cui al comma 7, lettera h);*

b *Potenziamento degli strumenti didattici e laboratoriali necessari a migliorare la formazione e i processi di innovazione delle istituzioni scolastiche;*

c *Adozione di strumenti organizzativi e tecnologici per favorire la governance, la trasparenza e la condivisione di dati, nonché lo scambio di informazioni tra dirigenti, docenti e studenti e tra istituzioni scolastiche ed educative e articolazioni amministrative del Ministero dell'istruzione, dell'università e della ricerca;*

d *Formazione dei docenti per l'innovazione didattica e sviluppo della cultura digitale per l'insegnamento, l'apprendimento e la formazione delle competenze lavorative, cognitive e sociali degli studenti;*

e *Formazione dei direttori dei servizi generali e amministrativi, degli assistenti amministrativi e degli assistenti tecnici per l'innovazione digitale nell'amministrazione;*

f *Potenziamento delle infrastrutture di rete, sentita la Conferenza unificata di cui all'articolo 8 del decreto legislativo 28 agosto 1997, n. 281, e successive modificazioni, con particolare riferimento alla connettività' nelle scuole;*

[4] `http://hubmiur.pubblica.istruzione.it/web/ministero/focus090715`(Settembre 2017)

[5] `http://www.gazzettaufficiale.it/eli/id/2015/07/15/15G00122/sg`(Settembre 2017)

g *Valorizzazione delle migliori esperienze delle istituzioni scolastiche anche attraverso la promozione di una rete nazionale di centri di ricerca e di formazione;*

h *Definizione dei criteri e delle finalità per l'adozione di testi didattici in formato digitale e per la produzione e la diffusione di opere e materiali per la didattica, anche prodotti autonomamente dagli istituti scolastici."*

2.3.2 Programma il futuro

Il MIUR nell'ambito del programma della Buona Scuola, in collaborazione con il CINI (Consorzio Interuniversitario Nazionale per l'Informatica) ha avviato l'iniziata Programma il Futuro[6] con lo scopo di portare nelle scuole i concetti di base dell'informatica grazie a strumenti semplici e divertenti. Si occupa di tradurre le attività di Code.org e fornisce supporto agli insegnanti tramite percorsi strutturati, videolezioni sui concetti di base dell'informatica, forum di discussioni e momenti di formazione. Non è necessaria nessuna conoscenza tecnica o scientifica per prendere parte a questa iniziativa. La partecipazione prevede due percorsi: uno base, che prevede la partecipazione all'"Ora del Codice" svolgendo attività di avviamento al pensiero computazionale in un'ora, e l'altro avanzata che consiste nel far seguire percorsi più strutturati in base alla fascia d'età. Entrambe le modalità possono essere svolte sia con *lezioni tecnologiche* tramite l'utilizzo di dispositivi elettronici, sia con *lezioni tradizionali* senza l'utilizzo e supporto di alcun dispositivo elettronico.

2.4 Strumenti proposti

2.4.1 Scratch

Scratch[7] è un progetto del *Lifelong Kindergarten group* guidato dal prof. Mitch Resnick. È un linguaggio di programmazione visuale completo (questo vuol dire che si potrebbe scrivere qualsiasi programma) object-oriented, molto facile da usare. Sono presenti dei blocchi che possono essere connessi tra di loro per definire il comportamento del programma. Oggi Scratch compie 10 anni. È stato lanciato nel 2007 con la prima versione e nel 2013 con la versione 2.0. Da allora, come dice lo stesso Resnick in un articolo sulla rivista *HelloWorld (Issue 2)*, Scratch ha attraversato diverse interazioni e sperimentato una crescita esponenziale nel numero di utenti utilizzatori diventando un sostegno per chi usa il coding all'interno delle aule scolastiche e non solo.

[6] `https://programmailfuturo.it/`
[7] `https://scratch.mit.edu/`

Per la creazione di Scratch sono stati ispirati dai lavori fatti nei dopo scuola nelle "Computer Clubhouse" da bambini che volevano creare le loro storie interattive senza avere però gli strumenti giusti a disposizione. Scratch è stato comunque ispirato dai lavori e dalle idee di Seymour Papert di cui Resnick fu allievo. Con l'avvento di Scratch 2.0 anche la community ha acquistato più valore e potenza portando a circa 22 milioni di progetti condivisi da tutto il mondo. Ad oggi Scratch è lo strumento più utilizzato anche per attività di coding e pensiero computazionale all'interno delle scuole in tutti i livelli, dall'infanzia alle secondarie di secondo grado.

La prossima tappa sarà Snap 3.0 come anticipato da Resnick nell'intervista alla rivista HelloWorld. Le aree del miglioramento riguarderanno il mondo fisico, per facilitare il collegamento con l'hardware esterno, il collegamento con il "mondo esterno" facilitando il collegamento di Scratch ai servizi online e soprattutto non irrilevante il supporto ai dispositivi mobile che ora manca.

Capitolo 3

Progetto COGITO

COGITO ha creato una sinergia tra pensiero computazionale e scuole coinvolgendo la visione di Papert come punto di partenza sull'impatto dell'informatica nella scuola primaria. Basandosi sul pensiero costruzionista di Papert, COGITO ha portato allo sviluppo di ambienti digitali progettati per favorire l'esplorazione e apprendimento di concetti e competenze interdisciplinari

3.1 Descrizione progetto

Il progetto COGITO nasce nel 2015 con l'idea di sperimentare le attività di pensiero computazionale e coding con alcune classi del Terzo Circolo Didattico di Cesena. L'obiettivo non è avere delle attività fini al coding, ma integrare tali attività con le più scolastiche, coniugando il "Pensiero Computazionale" con il "pensiero psico-pedagogico" proprio della nostra Scuola.

L'obiettivo del progetto Cogito è l'insegnamento dei concetti e delle tecniche fondanti della programmazione (coding) identificabili nel cosiddetto pensiero computazionale. Attività non fini a loro stesse ma a supporto dello sviluppo delle capacità di ragionamento logico, della risoluzione di problemi, dell'ideazione e della creazione di artefatti informatici digitali, e di sostegno all'apprendimento di altre materie.

3.1.1 Cos'è, da chi e quando è nato

Il progetto è nato in un contesto di iniziative internazionale (per esempio il sito "Code.org") e nazionali (a partire dal progetto "Programma il Futuro" promosso dal MIUR). L'obiettivo di queste iniziative è l'insegnamento dei fondamenti dell'informatica e della programmazione in tutti i livelli delle scuole, sin dall'infanzia. Questi movimenti sono spesso motivati da prospettive lavorative. Ad esempio dalla mancata capacità dell'università di produrre un

numero di laureati sufficiente per coprire il fabbisogno lavorativo presente e futuro. Esiste però una visione diversa: non solo le conoscenze legate al pensiero computazionale sono spendibili in ambito lavorativo, ma queste, se propriamente applicate su più livelli scolastici, potenziano le capacità creative del problem solving.

Seguendo questa visione, il progetto COGITO, pur riconoscendo l'importanza del mondo lavorativo, rivede la programmazione e il coding come importanti strumenti a supporto dell'apprendimento come lo sono la scrittura e la capacità di fare calcoli; strumenti che facilitano il creare, il pensare e l'immaginare dello studente, e quindi strumenti al servizio delle altre discipline. Il riferimento culturale di COGITO è il gruppo *Lifelong Kindergarten* presso il MIT Media Lab, nella figura del prof. Seymour Papert, che ha iniziato queste attività negli anni settanta (Capitolo 1) e oggi portate avanti da suoi allievi, uno fra tutti il prof. Mitchel Resnick, attuale responsabile del gruppo.

Come riportato sul sito del gruppo [1], la loro "missione" è sviluppare tecnologie che, nel medesimo spirito dei blocchi e dei colori a dita usati nelle scuole dell'infanzia, permettano di espandere ciò che gli alunni e le persone possono progettare, creare e imparare. Resnick riassume in modo efficace questa visione: ***"learn to code, code to learn"*** . Imparare a programmare è lo strumento stesso per imparare. Resnick vede ragioni più profonde e più ampie nell'imparare a programmare, non solo quelle legate al mondo del lavoro. Queste abilità non sono utili solo per gli informatici, ma per tutti, indipendentemente dagli studi, dagli interessi e dall'occupazione.

Questo modo di pensare ha un impatto significativo sulla progettazione e sullo sviluppo delle attività relative al pensiero computazionale nei contesti scolastici. Se lo scopo è quello di insegnare i concetti prettamente informatici del pensiero computazionale fine a se stesso, allora questo può essere fatto al di fuori della scuola. Inquadrando invece queste attività a supporto dell'apprendimento si apre un visione più ampia di cooperazione tra materie e contenuti computazionali. Una cooperazione dove gli strumenti tecnologici e il pensiero computazionale intrecciati alle discipline scolastiche diventano un innovativo strumento di insegnamento.

3.1.2 Strumenti utilizzati - Snap

"Snap! è Scheme vestito da Scratch" - Bryan Harvey

La piattaforma utilizzata per lo svolgimento delle attività in COGITO è *"Snap!"*: un ambiente di sviluppo per un linguaggio a blocchi, open-source ed eseguibile direttamente su browser. Creato da Jens Mönig e Brian Harvey

[1] `https://www.media.mit.edu/groups/lifelong-kindergarten/overview/` (2017)

dell'Università della California a Berkeley[2], è un ambiente di programmazione gratuito di tipo visuale che permette lo sviluppo semplificato di artefatti tramite l'interconnessione di blocchi logici, la cui successione definisce il comportamento del programma. La prima versione fu rilasciata nel 2011 quando venne usato per un corso introduttivo di Computer Science per studenti non informatici: "The Beauty and Joy of Computing", questo il nome del corso.

Snap! si ispira a Scratch e si rivolge sia a studenti alle prime armi che a quelli più esperti [14], includendo e ampliando caratteristiche di Scratch e di Scheme (un linguaggio di programmazione funzionale, un dialetto del Lisp[3] di cui mantiene tutte le caratteristiche). Nel Dicembre 2014, 100 scuole superiori di New York City introdussero il corso "The Beauty and Joy of Computing" come nuovo corso di AP utilizzando proprio Snap![4]

Snap! importa da Scratch:

- Interfaccia *drag-and-drop*
- Facili strumenti di animazione
- Metafore visuali per cicli, condizioni, etc.

Snap! importa da Scheme elementi *first-class*:

- Procedure
- Liste
- Oggetti
- "Continuazioni"

La principale differenza tra Scratch e Snap! è suggerita da *BYOB (Build Your Own Blocks)* (predecessore di Snap!): la possibilità di "creare nuovi blocchi" sintattici e semantici a complessità crescente, definendone la categoria di appartenenza, il nome, il tipo, e la lista dei parametri. Questa funzionalità ha consentito agli studenti del progetto COGITO di diventare "creatori di mondi" strutturati e organizzati sulla base degli obiettivi prefissati, e rendendo trasparenti allo studente concetti complessi e non funzionali ai fini didattici.

[2] `http://www.i-programmer.info/news/98-languages/8628-visual-language-snap-version-40-release.html`, Accesso 20 Agosto 2017

[3] Lisp (List Processor) è una famiglia di linguaggi di programmazione con implementazioni sia compilate sia interpretate, associata nel passato ai progetti di intelligenza artificiale. È stato anche il primo linguaggio a facilitare uno stile di programmazione funzionale.

[4] `https://en.wikipedia.org/wiki/Snap!_(programming_language)`,Accesso 20 Agosto 2017

3.1.3 Li2Lab

Le attività sono state svolte nell'aula TEAL denominata *Li2Lab (Little Turtle Lab)*, composta da banchi componibili, facili da spostare e adattare in base alle esigenze didattiche. La classe è stata organizzata tipicamente in gruppi da due, favorendo il lavoro in coppia degli studenti. In Li2Lab sono presenti circa una ventina di tablet, gli strumenti utilizzati dai bambini per lo svolgimento delle attività. Ogni gruppo aveva a disposizione un tablet per lavorare, stimolando così la cooperazione e il lavoro di gruppo.

COGITO rivede il computer nelle aule come strumento di modellazione. Il passaggio del computer da semplice calcolatore a strumento di modellazione consente di sviluppare linguaggi adatti alla costruzione di micromondi. Gli alunni e gli insegnanti diventano modellatori di realtà artificiali, riducendo la complessità del reale ad un modello con un'adeguata complessità da affrontare. Si crea il circolo virtuoso del "learn to code, code to learn", percorso che integra computer, pensiero computazionale, coding e apprendimento. Quest'ultimo è un circolo virtuoso poiché attraverso il computer è possibile rinnovare gli ambienti d'apprendimento e l'insegnamento di competenze multidisciplinari utili a loro per la comprensione e apprendimento dell'informatica stessa.

3.2 Primo anno

Per il primo anno le attività si sono svolte da Ottobre 2015 fino a Maggio 2016, con 4 classi (due seconde e due terze), con una media di 2 ore per classe ogni due settimane. Le attività sono state suddivise in due periodi, in cui sono stati usati strumenti diversi:

1. Da Settembre a Dicembre, dove sono stati introdotti i concetti di base del Pensiero Computazionale e del coding mediante un insieme selezionato di percorsi tratti dal sito "code.org" e dal sito di "Programma il Futuro". Ciò a portato i bambini a svolgere attività di programmazione sia su carta(unplugged) che su tablet(plugged) per arrivare alla risoluzione dei problemi proposti nei percorsi. Questi problemi tipicamente sono impostati come un gioco dove bisogna specificare il comportamento di un personaggio per fargli raggiungere l'obiettivo prefissato.

2. Da Gennaio a Maggio si è passati ad una fase più costruttiva, con la programmazione di **cartoline animate** sulla piattaforma Snap!.
   La cartolina animata è stata la chiave narrativa con cui condurre i bambini alla creazione dei programmi di varia natura, dalle più semplici di ripasso dei concetti visti sul pensiero computazionale, alle più complesse

e tematiche, in cui si cercava di creare collegamenti con argomenti visti in altre discipline.

3.2.1 Considerazioni primo anno

Un aspetto comune alle attività sviluppate sia nella prima sia nella seconda parte è stato quello di cercare di mantenere un collegamento continuo fra reale e virtuale, fra il mondo creato mediante coding e gli elementi e attività nel mondo reale e fisico. Ad esempio: le prime cartoline animate avevano, come immagine dello sfondo e dei personaggi che la animavano, disegni fatti su carta dai bambini stessi opportunamente scannerizzate. Le ultime avevano la mappa di Cesena, sulla quale volava un aliante, andando a visitare luoghi reali scelti dalle insegnanti, che i bimbi conoscevano avendoli visitati fisicamente e discussi.

In generale i bambini hanno mostrato generale entusiasmo e risposta positiva alle attività proposte. Si sono mossi nella creazione dei "mondi" a volte in modo guidato e a volte dando libero sfogo alla loro fantasia.

E' emersa chiaramente l'importanza di trovare le chiavi narrative appropriate con cui introdurre i concetti e guidare i bambini nelle attività. E' emersa l'utilità e importanza del lavoro in gruppo, che ha permesso di far prevalere logiche più collaborative e di condivisione rispetto a quelle competitive.

Sono comunque emerse alcune criticità, oggetto di riflessione e discussione fra docenti e insegnanti in prospettiva delle attività future. Esempi:

- La necessità di scegliere in modo opportuno i gruppi
- La non completa efficacia dei mezzi tecnologici utilizzati. in particolare l'uso di Snap! su tablet (in alternativa a Scratch, per il quale non esiste ancora una versione funzionante su dispositivi mobile) ha mostrato limiti vari per un suo utilizzo da parte dei bambini
- La corposità delle classi rispetto al numero dei docenti presenti in aula a seguire le attività
- La necessità di adottare ritmi più contenuti, sia per quanto riguarda i concetti che per quanto riguarda i tempi, cercando di differenziare il percorso tra secondo e terze, avendo maggior riguardo, soprattutto con le seconde, nell'adottare un ritmo più lento toccando un minor numero di contenuti

3.3 Secondo anno

Nel secondo anno di attività ci si è concentrati nel proseguire le attività iniziate nel primo anno avendo due obiettivi principali:

1. Consolidare, interiorizzare ed estendere le competenze basilari relative al pensiero computazionale introdotte durante il primo anno In particolare sono stati ripresi i seguenti concetti:

   - Algoritmo, programma, sequenza di istruzioni
   - Costrutti di ripetizione e di selezione
   - Evento e scambio di messaggi
   - Variabili

   è stato inoltre introdotto:

   - Il concetto di procedura, quindi di suddivisione top-down di un programma in procedure, come tecnica di decomposizione di un problema in sottoproblemi

   L'ambiente di programmazione utilizzato è stato, come per il primo anno, Snap!.

2. Attuare sin dal principio l'integrazione con le altre discipline scolastiche attraverso la costruzione incrementale di programmi identificati come micromondi. aventi come oggetto concetti e contenuti delle suddette materie e/o di più materie.

   La costruzione dei micromondi è stata l'asse portante sul quale si è sviluppata l'intera attività del secondo anno, con l'obiettivo non solo di riprendere i concetti ed elementi del pensiero computazionale visti al punto (1), ma anche utilizzare e manipolare concetti e competenze di studio delle altre materie.

   L'obiettivo è quello di far maturare nei bambini il concetto che il computer, così come il pensiero computazionale e il coding, è uno strumento estremamente flessibile con cui creare artefatti digitali di qualsiasi genere, non solo per risolvere problemi, ma anche per creare, progettare e apprendere.

   La costruzione dei micromondi è incrementale, integrando fasi di progettazione guidata a fasi di tipo esplorativo dove i bambini provano, testano, creano, esplorano in completa autonomia.

Un altro obiettivo che si è cercato di perseguire è stato quello di mettere in piedi sistemi di valutazione di cui si discuterà più avanti.

La maggiore novità introdotta in questo secondo anno è stata l'introduzione di un percorso parallelo a quello svolto in Li2Lab: il "percorso in aula".

Nel primo anno gli studenti hanno svolto attività di pensiero computazionale una volta ogni due settimane. A causa di queste e altre questioni tempistiche, nelle nuove lezioni i concetti non sono stati sempre ripresi e approfonditi, rischiando di creare un buco tra le lezioni. Si è pensato così di affiancare un percorso da svolgere in aula nella settimana successiva alla lezione del Li2Lab con lo scopo di riprendere, approfondire e consolidare i concetti visti nelle lezioni in Li2Lab. Le attività del secondo anno sono state strutturate alternando settimanalmente una lezione in Li2Lab e una lezione in aula.

3.3.1 Pianificazione

Le attività del secondo anno si sono svolte da Novembre 2016 a Maggio 2017. Le classi coinvolte erano le stesse del primo anno ora diventate due terze e due quarte.

Il lavoro di pianificazione insieme alle insegnanti è stato il punto di forza di questo progetto. L'unione delle loro conoscenze pedagogiche con quelle informatiche ha fatto sì che il percorso venisse strutturato coprendo tutti gli aspetti, non solo quindi quelli strettamente legati al mondo informatico. È stato inoltre possibile costruire un percorso che andasse di pari passo con le discipline scolastiche e affrontare in Li2Lab gli stessi argomenti che i bambini affrontavano in aula con le loro insegnanti. L'obiettivo che non è mai stato perso di vista è sempre stato "learn to code, code to learn", infatti attraverso il coding i bambini non hanno solo imparato cos'è una variabile, un ciclo for o un costrutto di selezione, hanno sperimentato come questi concetti siano utili anche per risolvere un problema di matematica o un'espressione, per tradurre delle parole o molto semplicemente per creare un videogioco.

Prima dell'inizio delle attività si sono svolti diversi incontri con le insegnanti aventi lo scopo di pianificare le attività del nuovo anno e individuare gli obiettivi da raggiungere (descritti ampiamente nel paragrafo 3.3).

Micromondo

Il concetto di micromondo utilizzato all'interno del progetto COGITO è il concetto introdotto da Papert in [9] I micromondi sono dei programmi che rappresentano un "mondo" che riguarda un certo argomento (matematica, storia, geografia,...). Coma la cartolina (utilizzata durante il primo anno), il micromondo ha degli attori (gli Sprite) e uno scenario (lo Stage).

Il micromondo è un mondo dove il contenuto può essere molto ricco, vario e soprattutto "dinamico" (al contrario della cartolina, che richiama staticità): a livello tecnico un micromondo è caratterizzato da una certa struttura fatta dagli Sprite e dallo Stage, da un certo comportamento complessivo degli Sprite e dello Stage, e quindi dalle interazioni che avvengono all'interno.

Il fatto che un micromondo appartenga a un certo ambito si riflette anche nell'insieme dei blocchi che vengono messi a disposizione (e poi, nel tempo, anche creati in base alle esigenze). I blocchi sono "orientati al dominio", hanno un significato in base al contesto e all'obiettivo del micromondo. Più in generale questo significa che ogni micromondo introduce un proprio **linguaggio**, composto dall'insieme dei blocchi (azioni) con cui si può costruire lo script degli attori. I blocchi sono presenti in misura minore, sono visibili solo quelli necessari al comportamento dello specifico micromondo. Non è detto che il linguaggio debba rimanere lo stesso, c'è la possibilità di introdurre nuovi blocchi e quindi di definire un comportamento personalizzato in base alle esigenze o alle richieste.

Snap! supporta la funzionalità di nascondere i blocchi e presentare quindi un micromondo con un linguaggio personalizzato. Questa possibilità è cruciale perché permette di definire il giusto livello di astrazione con cui costruire le conoscenze e con cui spiegare le cose.

3.3.2 Contributo dato

Come è stato già accennato una delle novità di questo secondo anno è stata l'introduzione di un percorso, parallelo a quello in Li2Lab, da svolgersi in aula nella settimana successiva a quella di laboratorio. L'idea di affiancare il tradizionale percorso in Li2Lab è nata dopo la richiesta delle insegnanti di rallentare con i concetti, perché i bambini dopo una settimana di pausa avevano bisogno di riprendere quelli già visti prima di poter andare avanti. Era necessaria una modalità che permettesse loro di assimilare la lezione precedente per poter andare avanti con quella successiva. Richiesta più che lecita, dato che la finalità è quella di puntare non sulla quantità delle attività svolte ma sull'effettiva comprensione di esse da parte degli alunni. Tuttavia avendo a disposizione solo due ore ogni due settimane non si poteva rallentare il lavoro, per timore di non riuscire a perseguire gli obiettivi prefissati.

Da qui l'idea di affiancare il percorso tradizionale del Li2Lab con un percorso parallelo in aula, dove proporre ai bambini attività di potenziamento e consolidamento. Le maestre si sono mostrate entusiaste rispetto alla proposta: un'opportunità per i bambini di non perdere il ritmo con la settimana di pausa tra una lezione e l'altra e un momento utile per chiarire tutti quei concetti non chiari o semplicemente per consolidarli e permettere ai bambini di farli propri.

Non era solo un'opportunità per i bambini, ma anche per le insegnanti che attraverso queste attività in aula avevano la possibilità di "rivedere" e assimilare contenuti utili anche per le loro normali attività o per iniziare a porre le basi per un percorso da svolgere in autonomia.

Struttura percorso in aula

L'idea di partenza prevedeva per ogni classe un percorso così strutturato:

- Lezione Li2Lab di introduzione a nuovi micromondi e concetti o continuazione del micromondo già iniziato

- La settimana successiva lezione in aula di potenziamento e approfondimento dei concetti/micromondi introdotti precedentemente

Nelle lezioni in aula sono stati trattati gli argomenti che durante le attività in Li2Lab i bambini avevano capito meno o ripresi quei concetti che richiedono più ore per essere assimilati.

Le attività in aula sono state svolte in diverse modalità a seconda delle esigenze della classe e degli argomenti affrontati. Inoltre sono state svolte lezioni "unplagged" (senza quindi l'utilizzo di dispositivi elettronici), sia "plugged" (grazie al supporto dei tablet).

Nel primo periodo dell'anno le attività in aula erano strutturate in tre momenti:

1. Ripasso dei concetti affrontati la settimana prima in Li2Lab: attraverso la proiezioni di slide venivano ripresi i concetti della settimana precedente e ripessati con la partecipazione attiva dei bambini.

2. Quiz: il momento più ludico della lezione era il quiz, dove i bambini, divisi in squadre, dovevano rispondere alle domande proposte. Le domande erano di varia natura: dalle risposte a crocette, agli script in cui indovinare l'output, dalle frasi con parole mancanti da completare ai giochi di ripasso.

3. Tablet per fare le attività: l'ultimo momento della lezione prevedeva l'esercitazione su mini-micromondi in cui i bambini attraverso esercizi e sfide via via di difficoltà maggiore avevano l'opportunità di esercitarsi, comprendere meglio e fare propri i concetti ripassati nelle attività precedenti.

3.3.3 Esempi del percorso fatto

Possiamo dire che l'anno è stato suddiviso in quattro parti dove sono stati portati avanti quattro micromondi principali.

- Tarta-Matematica - da Novembre a Dicembre
- Micromondo del Disegno - Gennaio e Febbraio
- Tarta-Parole - Marzo e Aprile
- Videogioco Pioggia nel Pineto - Maggio

Micromondo Matematica - TartaMatematica

Il micromondo TartaMatematica è stato il primo ad essere introdotto ed è relativo al mondo della matematica; permette di scrivere programmi che risolvono problemi. Il protagonista, lo Sprite, è una tartaruga e lo sfondo è una lavagna. La chiave narrativa con cui è stato presentato ai bambini è stata: "insegniamo alla tartaruga a risolvere i problemi di matematica, fatto questo, la tartaruga aiuterà noi a risolverli!"

La cosa importante è il linguaggio definito per il micromondo, presenta infatti solo i blocchi necessari alle sue azioni. Le azioni che la tartaruga dovrà compiere sono "scrivere sulla lavagna" e "risolvere operazioni".

Li2Lab: Il percorso è stato sviluppato incrementalmente in modo che attraverso piccoli step, i bambini scoprissero le funzionalità e il linguaggio proprio del micromondo.

Si è partiti introducendo il concetto di Stringa per poi far sperimentare ai bambini attraverso prove ed esempi come queste venivano effettivamente scritte dalla Tartaruga sulla Lavagna Dalle stringhe si è passati poi ai numeri sempre con lo stesso approccio, sperimentando e provando. Dai valori si è passati alle espressioni. La tartaruga non solo sa scrivere stringhe e numeri, ma sa fare anche i conti tramite la categoria degli operatori.

È in questo periodo, con il micromondo TartaMatematica che è stato ripreso il concetto di **variabile**: uno spazio dove poter memorizzare dei valori associato ad un certo nome. La metafora usata per i bambini è stata quella della scatola che ha un contenuto (valore) e un'etichetta (nome). In Li2Lab si scoprono i blocchi dedicati alle variabili, come crearla, assegnare un valore e cambiarlo.

La fase successiva è stata quella dell'implementazione del problema. L'obiettivo è quello di far vedere come un problema di matematica possa essere rappresentato e risolto tramite un programma. In questa fase è stato importante il concetto di **algoritmo** per rappresentare la sequenza di passi logici

Figura 3.1: Esempio del Micromondo TartaMartematica

necessari alla risoluzione e di come questi, scritti nell'ordine errato non portavano al risultato atteso. Si è partiti da un problema molto facile per poi passare a problemi via via più complessi proposti dalle maestre. La fase finale ha visto la creazione del testo del problema da parte dei bambini, per poi proporlo da risolvere tramite il Micromondo TartaMatematica (Figura 3.1).

Aula: Per quanta riguarda l'approfondimento in aula le prime lezioni sono state tutte strutturate secondo lo schema ripasso-quiz-approfondimento (con o senza tablet)

Gli argomenti ripassati e approfonditi sono stati relativi al micromondo matematica: il linguaggio utilizzato dalla tartaruga, gli operatori, le espressioni e soprattutto le variabili. Quest'ultimo è stato il concetto sul quale è stato necessario insistere e soffermarsi per qualche lezione (soprattutto con le classi terze). Attraverso la metafora della scatola sono state prima proposte attività unplugged dove i bambini immedesimatisi nella tartaruga hanno svolto il compito simulando il suo comportamento: per esempio per inizializzare il valore della scatola, bisognava mettere fisicamente il valore nella variabile, mentre per controllare il valore della variabile bisognava aprire la scatola e leggere il suo valore.

Il passo successivo è stato quello di passare all'uso dei tablet per la risoluzione dei problemi.

Micromondo Inglese - TartaTraduttrice

Obiettivo di questo micromondo è stato quello di considerare l'utilità degli algoritmi non solo in ambiti prettamente matematici. Viene introdotto il con-

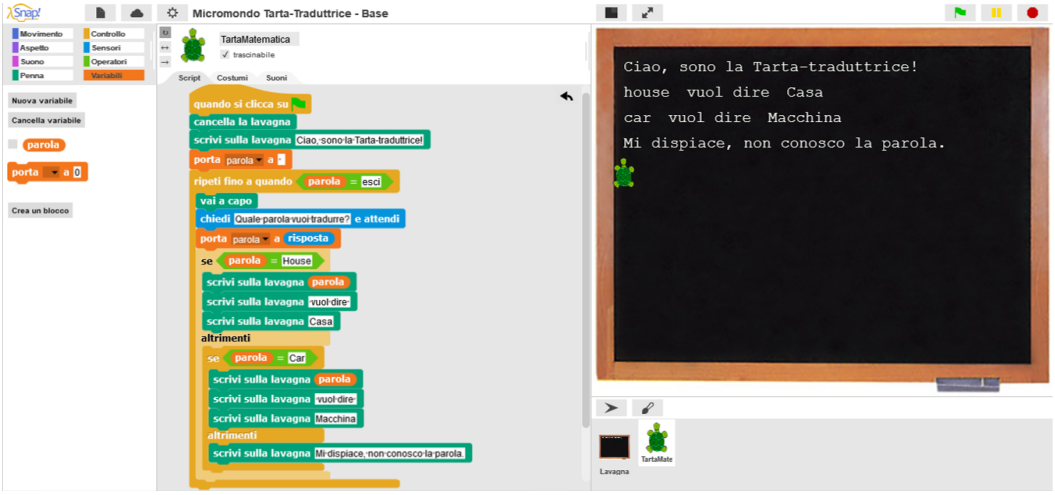


Figura 3.2: Esempio di Micromondo TartaTraduttrice

cetto di input da parte dell'utente e illustrati i blocchi utilizzati in Snap! per fare ciò. In questa fase viene anche ripreso il costrutto del 'se', la condizione.

La chiave narrativa per i bimbi è stata quella di voler costruire un micromondo in cui la tartaruga li aiutasse a tradurre le parole dall'inglese all'italiano.

Li2Lab: Il primo passo è stato quello di introdurre il blocco dell'**input** e capire il suo funzionamento: il blocco "chiedi" memorizza la parola inserita dall'utente in una variabile predefinita chiamata "risposta". Il secondo passo è stato quello di controllare la parola inserita grazie al blocco della condizione (il blocco del **'se'**): "Se si conosce la parola, questa viene tradotta, altrimenti esce un messaggio di errore". (Figura 3.2).

Aula: Le attività in aula hanno sempre seguito lo schema ripasso-quiz-potenziamento. Gli argomenti di ripasso sono stati i blocchi relativi all'input e al se È stato ripassato e consolidato l'algoritmo necessario alla TartaTraduttrice per tradurre una parola.

-chiedere la parola che si vuole tradurre
-controllare se si conosce la parola
–se la conosce, allora la traduce
–altrimenti scrive "mi dispiace"

Successivamente i bambini si sono cimentati nella traduzione di più parole e quindi nell'utilizzo di 'se' annidati.

Micromondo Italiano - TartaParole

Obiettivo di questo micromondo è stato quello di considerare l'utilità degli algoritmi non solo in ambiti prettamente matematici ma anche negli altri come quello letterario. È stato introdotto il concetto di **"Stringa"** e di come queste vengano memorizzate nella memoria del computer: una lettera per ogni spazio di memoria, ogni carattere ha la sua "celletta" e quindi una sua posizione.

Li2Lab: Le attività si sono concentrate sugli algoritmi per manipolare le stringhe grazie all'utilizzo dei blocchi a disposizione nella categoria operatori: *lunghezza di* A, *lettera in posizione* X *di* A, *unione di* A e B. I bambini hanno provato a scrivere una lettera per riga, a contare il numero di vocali presenti in una parola e a implementare l'algoritmo per scrivere una parola al contrario. (Figura 3.5)

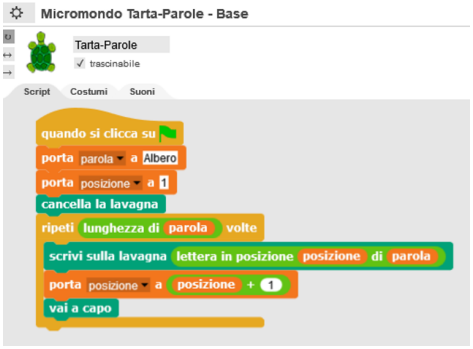


Figura 3.3: Una lettera per riga

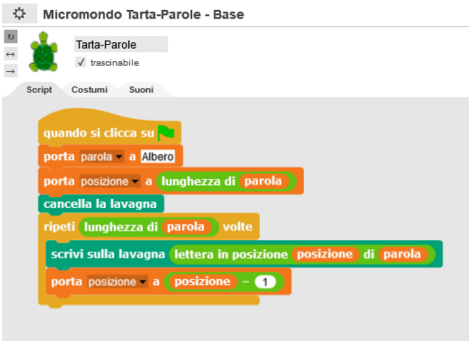


Figura 3.4: Parola al contrario

Figura 3.5: Esempi Micromondo TartaParole

Aula: Le attività in aula hanno sempre seguito lo schema ripasso-quiz-potenziamento. Gli argomenti di ripasso sono stati i blocchi della categoria operatori necessari a manipolare le stringhe questo attraverso degli script proiettati alla lavagna in cui i bambini dovevano indicare l'output finale.

Micromondo Disegno

Il micromondo del disegno è stato quello utilizzato come micromondo ludico, dove i bambini hanno dato libero sfogo alla loro fantasia (Figura 34). Lo stage era sempre la lavagna, e lo Sprite la tarturagache aveva a disposizione tutti i movimenti necessari per muoversi nello spazio e per disegnare. Giocando con gli algoritmi i bambini sono riusciti a creare figure geometriche anche molto complicate con effetti ottici spettacolari, grazie anche all'introduzione del cambio colore della penna.

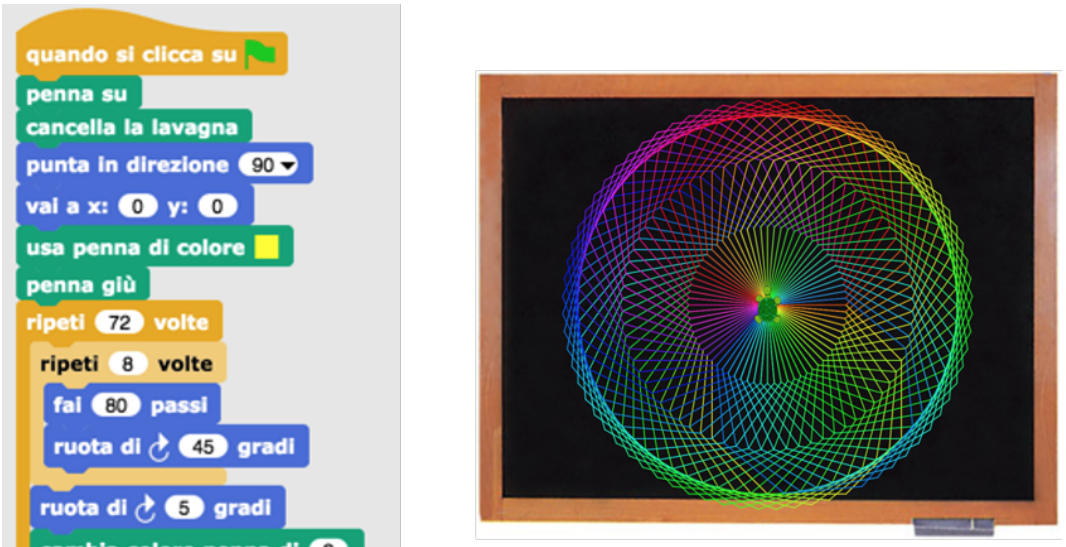


Figura 3.6: Esempio di Micromondo Disegno

Li2Lab: In Li2Lab il micromondo del disegno è stato generalmente utilizzato nella seconda parte di ogni lezione, nell'ultima mezz'ora di tempo. I bambini aspettavano con ansia questo momento per sfidarsi all'insegna del "Disegno più spettacolare".

Videogioco - Pioggia nel Pineto

Il Micromondo della Pioggia nel Pineto è un videogioco vero e proprio sviluppato nell'ultimo mese di attività, con l'obiettivo di riprendere tutti i concetti visti durante l'anno e di introdurre il concetto di **multi-sprite** e di **evento**.

Il tema del videogioco è stato la poesia di d'Annunzio la "Pioggia nel Pineto", che i bambini hanno studiato e approfondito nel corso dell'anno scolastico.

Il videogioco prevede la caduta dal cielo di alcune goccioline che portano con loro i versi della poesia, lo sfondo è un pineto disegnato a mano dai bambini e poi opportunamente digitalizzato per essere utilizzato sul tablet. Il gioco prevede che al tocco della gocciolina questa sparisca emettendo un suono e facendo guadagnare un punto. Ovviamente le funzionalità possono essere estese in base all'inventiva dei "costruttori".

Ogni gruppo ha quindi creato un videogioco personalizzato sia nella grafica che nelle funzionalità. La creazione è stata sviluppata in modo incrementale sempre procedendo per step distinguendo una fase iniziale di creazione e im-

plementazione guidata dai docenti e una fase finale di totale autonomia dove ogni gruppo ha aggiunto le funzionalità che ha ritenuto più opportune.

I passi per la creazione del micromondo sono stati:

- Posizionamento in alto della gocciolina
- Algoritmo per far cadere la goccia fino al fondo, trascinando con se un verso
- Aggiunta del suono quando la goccia tocca il fondo
- Aggiunta dell'evento *"Quando sono cliccato"* per far sì che al tocco la goccia scomparisse e/o cambiasse costume

Successivamente è stato impostato il comportamento della gocciolina che scende dal cielo portando con se i versi

Il Micromondo del videogioco è stato l'unico a non aver avuto un percorso separato in aula, ma anzi si procedeva allo sviluppo e creazione di esso senza distinzione tra lezione in aula e lezione in Li2Lab. (Figura 3.9)

Figura 3.7: Videogioco Terze

Figura 3.8: Videogioco Quarte

Figura 3.9: Esempi ambientazioni Videogioco "Pioggia nel Pineto"

3.3.4 Considerazioni secondo anno

Il focus su cui si è concentrato COGITO in questo secondo anno è stato quello di porre l'attenzione sull'importanza dell'integrazione tra il pensiero

computazionale e le discipline scolastiche. Il lavoro svolto durante l'anno ha permesso di mettere a regime le discipline con le attività di pensiero computazionale e coding. Le attività sono sempre state pensate e costruite non pensando al learn to code, ma al code to learn.

I bambini hanno mostrato entusiasmo e curiosità nello scoprire come i costrutti e i concetti visti in Li2Lab venivano ritrovati in aula con le insegnanti di matematica, italiano, geografia, arte,.. La risposta è stata molto positiva.

In questo secondo anno di attività hanno fatto propri molti concetti visti nell'anno precedente e li hanno interiorizzati a tal punto da riconoscerli all'interno delle discipline didattiche. Un esempio raccontato dall'insegnante di Matematica: durante una normale lezione di matematica la maestra assegna un problema, una bambina alza la mano ed esclama: "maestra qui dobbiamo usare il se!".

È emersa l'utilità delle chiavi narrative, i bambini ricordano meglio i concetti associandoli ad esempi già fatti e ripercorrendoli. È emersa anche quest'anno l'utilità e importanza del lavoro in gruppo, che ha permesso di far prevalere logiche più collaborative e di condivisione rispetto a quelle competitive. I gruppi sono sempre stati decisi dalle insegnanti, di solito con il pattern uno bravo e uno meno bravo, per spronare il meno bravo a fare meglio e ad apprendere dal più bravo. Si è notato come invece questa modalità sia stata in alcuni casi controproducente, il più bravo si sentiva rallentato dal meno bravo e procedeva in autonomia lasciando così il meno bravo senza far nulla e quindi demotivato. È emerso come il gruppo "a livello" sia invece ottimale per attività di questo tipo. Bambini con lo stesso livello di competenza insieme compongono un'eccellenza. Sicuramente questo sarà un aspetto da confermare con le attività del prossimo anno.

Capitolo 4

Progetto COGITO - Valutazione

È sicuramente di fondamentale importanza capire se le attività svolte durante questi due anni abbiano rilevanza, e individuare metriche per la valutazione dei risultati portati dal pensiero computazionale e coding a regime con le altre discipline scolastiche. La valutazione è un punto cruciale e importante, non solo da un punto di vista informatico del "code to learn", ma anche (e soprattutto vista la natura del progetto COGITO) del "learn to code", per capire quanto i concetti e i metodi del pensiero computazionale possano influire nell'apprendimento e nella comprensione dei concetti delle altre discipline.

Non è così facile però avere una valutazione oggettiva soprattutto del "code to learn". Sono aspetti che vanno maturati, fatti propri, che potrebbero avere dei risconti fra degli anni.

Per valutare il "learn to code" sono stati proposti ai bambini degli esercizi e delle sfide da svolgere in autonomia per capire quanto di quello affrontato e studiato fosse stato capito. Il riscontro è stato molto positivo.

I benefici del "code to learn" però non sono misurabili in modo semplice e soprattutto oggettivo. Servono infatti conoscenze e metodi pedagogici complessi, seguiti dall'analisi e da valutazioni su campioni di studenti che non hanno preso parte al progetto COGITO così da ottenere un metro di confronto rispetto al processo di risoluzione a diverse tipologie di problemi. Allo stato dell'arte abbiamo ritenuto importante far emergere l'importanza del progetto COGITO non solo dal punto di vista informatico ma anche e soprattutto da quello pedagogico. Per farlo è stato chiesto il supporto alle insegnanti che collaborano al progetto, chiedendo di rispondere alle domande di seguito riportate:

1. Perché COGITO è importante?

    - Per noi Cogito è importante perché risponde alla nostra ricerca di dare senso ai percorsi di apprendimento in un'ottica di "benesse-

re". Cogito rende possibile uno spazio di pensiero necessario ad ogni studente per riconoscersi e farsi conoscere, in cui sentirsi cioè competente. La proposta "difficile" diviene allettante perché crea compiti significanti e perciò autentici, che portano i bambini a sentirsi responsabili nello svolgimento delle azioni. Le attività, legate in un nodo narrativo, coinvolgono, stupiscono,toccando corde ludiche e creative .Dentro una situazione stimolante i bambini diventano protagonisti di un processo di cambiamento in cui non solo si rispettano i tempi personali ma l'errore cambia notevolmente valore, divenendo stimolo per riflessioni, aggiustamenti, possibilità diverse. Da ultimo riteniamo che Cogito faccia una grande scommessa, quella cioè di innovare radicandosi su di una solida tradizione psicopedagogica.

2. Quale valenza può avere per la didattica?

   - Sicuramente Cogito ha portato una profonda riflessione sull'uso della parola. Nelle attività di Cogito la parola tra i bambini diviene scambio di opinioni, idee, decisioni per definire e realizzare i percorsi rendendoli maggiormente abili ad argomentare le proprie ragioni in una visione di relazioni che esige rispetto e attenzione per l'altro, indispensabili anche per gettare semi di educazione alla cittadinanza. Inoltre la strategia del lavoro in coppia e/o di piccolo gruppo si affina senza precludere la possibilità di scambio attivo tra gruppi. Anche tra gli adulti coinvolti in Cogito, la parola diviene veicolo di scambio, confronto di professionalità, arricchimento e collaborazione reale nella costante progettazione in itinere.

3. Sono emersi miglioramenti nell'apprendimento delle materie tradizionali, contestualmente alla sperimentazione di COGITO?

   - Sinceramente non possiamo dire quanto Cogito abbia influenzato e influenzi tuttora i percorsi disciplinari. Sicuramente le strategie di gruppo sperimentate nell'aula Li2Lab sono state spesso richieste dagli alunni e riproposte in classe, rendendo sempre più autonomi i bambini nell'organizzazione L'entusiasmo, ma anche le difficoltà incontrate in Cogito, li ha resi generalmente più attenti nella scelta di un personale metodo di studio. Senza dubbio Cogito ha permesso a bambini con difficoltà nella letto-scrittura di emergere in un linguaggio "altro" adottato nel percorso, aumentando la propria autostima e valorizzazione nel gruppo. Nel contempo Cogito permette alle eccellenze di mettersi in gioco senza prevaricare o concorrere nel gruppo. Le attività in Li2Lab hanno anche mostrato ai bambini un

modo costruttivo di utilizzo dei devices, in cui non necessariamente si deve parlare di matematica, in cui labilità è legata al pensiero e dove impegno e creatività viaggiano insieme al gusto di apprendere nuovi linguaggi.

4. Pensi che il pensiero computazionale aiuti i bambini nelle attività didattiche? Se sì, perché?

   - Perché permette ai bambini di acquisire un nuovo modo di affrontare le situazioni di apprendimento: gli dà la certezza che si possa semplificare ogni "problema", li rende consapevoli della necessità di organizzare procedure e informazioni per raggiungere un obiettivo; gli consente di ripescare e replicare acquisizioni funzionali già adottate;

5. Quali problemi o sfide si dovranno affrontare in futuro?

   - Il problema maggiore sarà quello culturale. Trovare cioè un equilibrio di vedute tra i due fronti che oggigiorno si contrappongono anche all'interno della scuola: tecnologia sì/tecnologia no. Sarà una conquista condividere che uno strumento tecnologico possa essere positivo e funzionale, anche nel mondo della scuola, perché l'obiettivo dei progetti da realizzare non sarà lo strumento in sé ma ciò che veicola, il pensiero che sottende: non sono il tablet o il PC o lo smartphone il nostro scopo... restano solo strumenti utili in mano all'uomo e/o al bambino che li usa. Anche sperimentare e modificare strategie nella gestione dei gruppi classe sarà necessario e determinante per rinnovare realmente una scuola dove spazi e tempi sembrano sempre più ristretti. Il tema della valutazione e dell'errore saranno sicuramente necessitanti di riflessione approfondita. Noi sogniamo una scuola che non omologa, una scuola dove tutti i bambini ("tutti e non uno di meno") trovino spazio e opportunità, nelle carenze e nelle eccellenze.

6. Consigli per il futuro

   - Dubbi e perplessità sono leciti e indispensabili anche per noi coinvolti nel Progetto. Ogni esperienza necessita di coinvolgimento ma anche di esitazione che ci permette l'attenzione Ogni evoluzione della storia dell'uomo ha fatto un passo avanti ripescando all'indietro I cambiamenti esigono attenzione, riflessione, capacità di mettersi in gioco, grande professionalità e collaborazione. Siamo convinti

che solo così anche le avventure più spericolate possano trovare un approdo significativo.

Conclusioni

In questa tesi si presenta il concetto di pensiero computazionale e coding, inizialmente studiati e proposti da Seymour Papert padre del costruzionismo, e di come questi siano tornati all'attenzione della società grazie all'articolo di Janette Wing "Computational Thinking". Il pensiero computazionale viene proposto come quarta abilità di base oltre al calcolo, alla lettura e alla scrittura. Diversi promotori stanno cercando di portare questo concetto all'interno delle scuole primarie e non solo, con lo scopo di aumentare l'interesse verso il mondo dell'informatica grazie anche al supporto di movimenti nazionali e internazionali che intervengono con percorsi e strumenti di supporto alla didattica e agli insegnanti. In questo contesto il principale contributo si colloca all'interno del progetto COGITO che sperimenta l'integrazione tra il pensiero computazionale e il pensiero pedagogico.

Attraverso percorsi costruiti grazie anche al contributo delle insegnanti è stato possibile proporre attività che coadiuvassero i concetti e principi del pensiero computazionale con i concetti e i principi delle tradizionali discipline scolastiche. I destinatari di questo percorso sono stati gli alunni di una scuola primaria di Cesena, che attraverso le attività proposte hanno costruito in modo incrementale micromondi disciplinari integrando sia le conoscenze informatiche che quelle scolastiche. I risultati mostrano come i principi informatici vengano fatti propri e utilizzati senza problemi, d'altro canto però non è ancora stato possibile valutare quanto, dal punto di vista pedagogico, questo percorso aiuti i bambini ad acquisire le competenze descritte all'interno della tesi. Per il futuro sicuramente uno dei primi obiettivi sarà capire come poter valutare l'acquisizione delle competenze sia dal punto di vista informatico che dal punto di vista didattico/pedagogico.

Infine come suggeriscono le insegnanti, la più grande conquista sarebbe quella di trovare un equilibrio dal punto di vista culturale tra i due fronti che si contraddistinguono anche (e non solo) all'interno della scuola: tecnologia sì/tecnologia no.

Appendice

Quest'appendice ha lo scopo descrivere alcuni esempi del percorso fatto durante l'anno. Si prendono due Micromondi d'esempio: TartaMatematica e TartaParole. Per ogni attività verrà fornita una scaletta con tutti i punti/passi affrontati in Li2Lab e una scheda con i concetti visti a lezione distribuita poi ai bambini.

4.1 Micromondo TartaMatematica

Il Micromondo TartaMatematica è stato il primo Micromondo affrontato sia con le Terze che con le Quarte. In totale sono state svolte 6 Lezioni su questo argomento: tre in Li2Lab e tre in aula di consolidamento. Verrà presentato di seguito il percorso sviluppato nelle 3 lezioni in Li2Lab e alcune delle schede distribuite agli alunni.

4.1.1 Li2Lab

Lezione 1

Dopo un primo ripasso dei concetti visti l'anno prima (algoritmo, programma, linguaggio) è stato introdotto il Micromondo TartaMatematica che permette di scrivere programmi che risolvono problemi di matematica (Scheda 1).
La chiave narrativa per i bambini è stata: insegniamo alla tartaruga a risolvere problemi di matematica fatto questo, la tartaruga aiuterà noi a risolverli. Il Micromondo si presenta con una lavagna come stage, una tartaruga come sprite e i blocchi necessari per la scrittura dello script.

I primi programmi hanno esplorato le funzionalità del micromondo iniziando a scoprire il funzionamento dei blocchi.
La tartaruga sa scrivere sulla lavagna grazie al blocco "scrivi sulla lavagna"
La tartaruga oltre a scrivere sulla lavagna le stringhe, può scrivere anche i numeri.
La tartaruga non solo sa scrivere, numeri e parole ma sa anche fare i conti attraverso i blocchi della categoria "operatori".

Dopo prime prove ed esperimenti sull'uso dei blocchi descritti nei punti precedenti, si è introdotto il concetto di **variabile** (Scheda 1). Le variabili servono per memorizzare valori ("contenuto della variabile") associati ad un nome ("etichetta della variabile"). Il contenuto di una variabile può cambiare nel corso dell'esecuzione di un programma.

Con i bambini è stato utilizzata la metafora delle scatole: una variabile come la scatola ha un etichetta e un contenuto e attraverso dei foglietti di carta con su scritto un valore si è simulato l'assegnamento di una variabile e la lettura del contenuto. Sono stati infine introdotti i blocchi forniti da Snap! per lavorare con le variabili: creazione nuova variabile, "porta variabile a" (per inizializzare o cambiare il valore).

Il passo successivo è stato introdurre come un problema di matematica possa essere rappresentato mediante un programma. Il concetto di algoritmo in questo passaggio è importante per rappresentare la sequenza dei passi risolutivi.Il primo problema è stato molto semplice, riportato qui di seguito:

Esempio - Problema mele
Luca va al mercato, compra 3kg di mele rosse e 3kg di mele verdi. Quante mele compra in tutto Luca?

Si vede quindi l'implementazione del problema tramite pseudocodice, successivamente viene risolto tramite lo script visto nella Figura 31. Si possono considerare i dati del problema come delle variabili dove verrà memorizzato il valore e poi fare le operazioni utilizzando quelle e non i valori numerici.
numero-mele-rosse:=3
numero-mele-verdi:=2
numero-mele-in-totale:= numero-mele-rosse + numero-mele-verdi

Lezione 2
La lezione successiva ha visto l'introduzione del concetto di **input**, come possibilità di far inserire all'utente i dati per risolvere il problema senza dover ogni volta modificarli nello script (Scheda 2).

Lezione 3
L'ultima lezione sul Micromondo Matematica è stata sviluppata sotto forma di valutazione proposta ai bambini come Sfida! Ad ogni gruppo è stato assegnato un problema preparato dalle insegnati e in modo autonomo i bambini hanno scritto lo script risolutivo.

Nelle pagine successive si trovano le schede distribuite ai bambini con le spiegazioni di quanto detto e i relativi esempi.

Micromondo Tarta-Matematica

Insegnamo alla tartaruga a risolvere i problemi di matematica, poi lei aiuterà noi a risolverli

STAGE ---------->

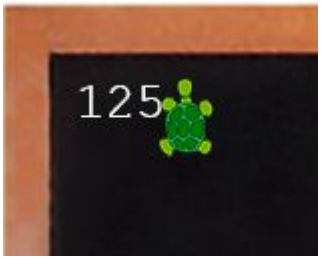
<---------- SPRITE

INIZIAMO…

>>Scriviamo "ciao"

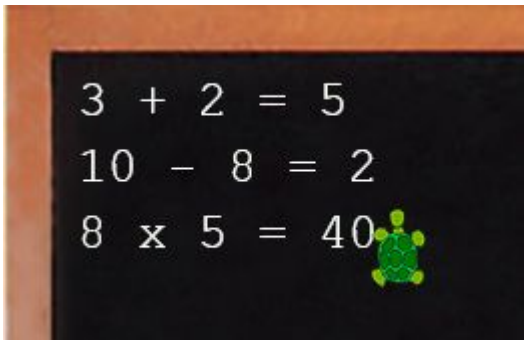
"Ciao" è un testo o una sequenza di caratteri.
In informatica una sequenza di caratteri si chiama STRINGA

>>Possiamo scrivere anche i numeri



I numeri rappresentano dei VALORI

>> Facciamo fare un po' di conti alla nostra tartaruga



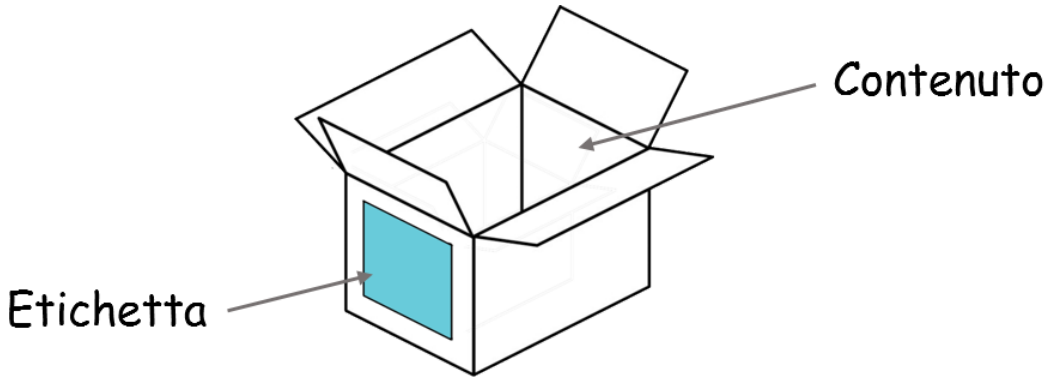
"+" , "-" ," x" , vengono detti OPERATORI

Un'ESPRESSIONE aritmetica, in matematica, è un insieme di numeri legati da operatori

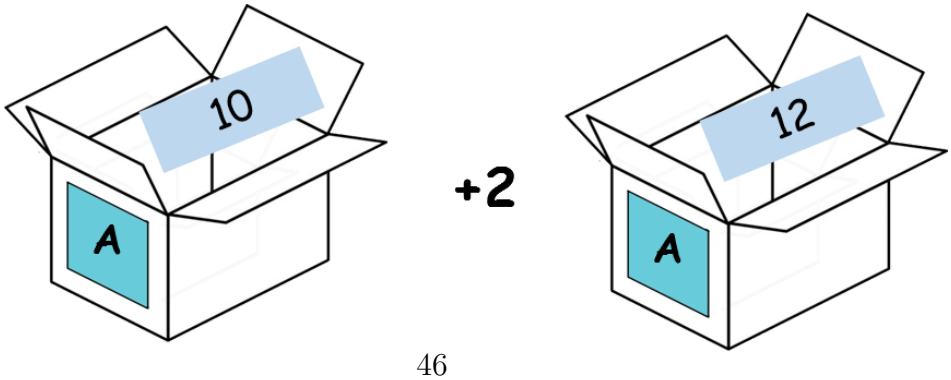
Le Variabili

Una VARIABILE è un contenitore (scatola) che ha:

1. Un'etichetta, che è il nome della variabile
2. Un contenuto, che può cambiare nel tempo



ESEMPIO:



E SE VOGLIAMO CAMBIARE I DATI DEL PROBLEMA?

...senza modificare tutte le volte il programma!

Introduciamo le azioni di INPUT:

AZIONI CON CUI IL PROGRAMMA IN ESECUZIONE CHIEDE IN INGRESSO UN DATO

Su Snap!: BLOCCO CHIEDI

Il blocco CHIEDI chiede in ingresso il dato all'utente che sta usando il programma e lo inserisce nella variabile RISPOSTA

4.2 Micromondo TartaParole

Il Micromondo TartaParole è stato l'ultimo micromondo affrontato prima di passare al videogioco "La pioggia nel Pineto". Sono state svolte due lezioni in totale: una in Li2Lab e una in aula. Verrà presentato di seguito il percorso sviluppato nella lezione in Li2Lab e alcune delle schede distribuite agli alunni.

Il micromondo TartaParole è stato utilizzato per far emergere l'importanza degli algoritmi non solo in ambiti prettamente matematici ma anche in quelli "letterari". E' stato inoltre utilizzato per approfondire il concetto del costrutto "se" e per introdurre i blocchi necessari a manipolare le stringhe.

Dopo aver presentato il micromondo, che si presenta a livello grafico esattamente come quello matematico, con una lavagna come stage e una tartaruga come sprite, sono stati introdotti i blocchi necessari a manipolare le stringhe. Questi si trovano nella categoria **"operatori"**, in particolare sono stati fatti esempi ed esercizietti con il blocco *"lunghezza di A"*, *"lettera in posizione X di Y"*, *"unione di A e B"*. Attraverso l'utilizzo di questi blocchi i bambini hanno costruito degli algoritmi per svolgere i compiti richiesti.

Nelle pagine successive si trovano le schede distribuite ai bambini con spiegazioni ed esempi.

MICROMONDO TARTA-PAROLE

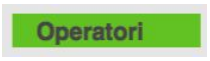
In questo micromondo la tarta ha a disposizione dei blocchi (istruzioni) per manipolare e giocare con le parole o frasi. La Tarta-Parole usa il termine **stringa** per intendere una sequenza di lettere che formano una parola o una frase.

BLOCCO CHIEDI

Anzitutto, come nel micromondo Tarta-Traduttrice, c'è il blocco chiedi per chiedere in **input** una stringa:

La stringa inserita in input viene messa nella variabile predefinita **risposta**

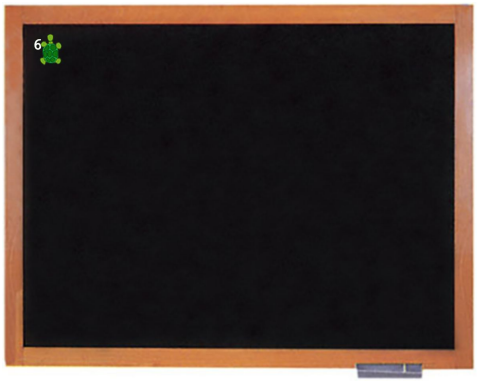
BLOCCO "LUNGHEZZA DI _"

Poi nella categoria 

c'è l'operatore 

, utile per calcolare la lunghezza della stringa specificata come parametro.

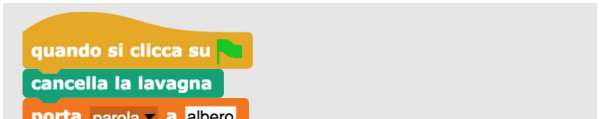
Esempio:

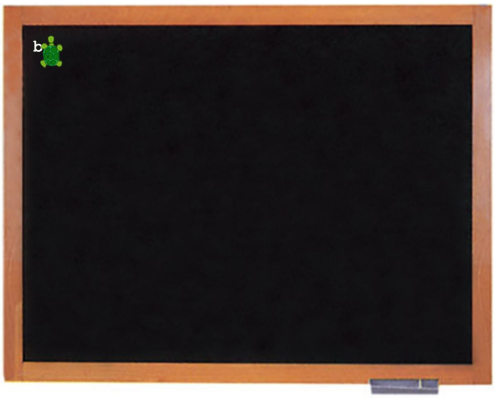


BLOCCO "LETTERA IN POSIZIONE _ DI _"

L'operatore 

invece è utile per trovare le lettere di una stringa. In particolare il blocco funziona con due parametri: una posizione (un numero) e una stringa. Il blocco trova qual è la lettera che si trova alla posizione specificata nella parola. Esempio:

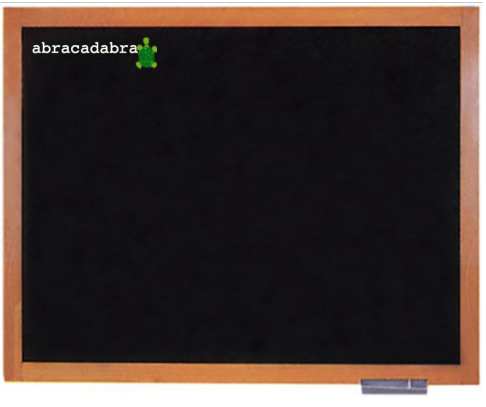




OPERATORE "UNIONE DI _"

Infine l'operatore 

è utile per concatenare insieme stringhe. In particolare il blocco funziona con due o più parametri, che sono le stringhe da concatenare e restituisce la stringa risultato della concatenazione. Esempio:

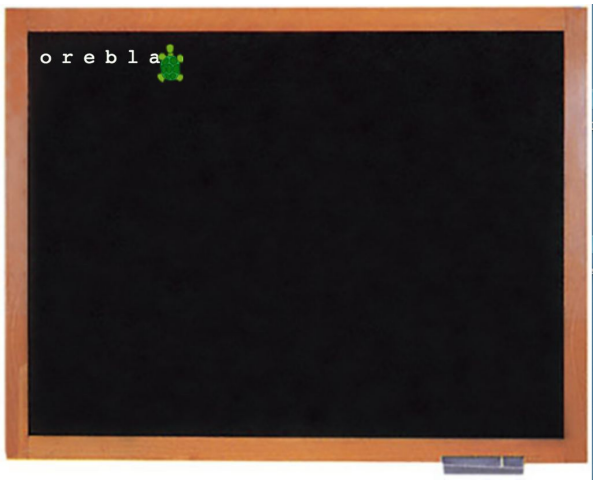


ALGORITMO #1: SCRIVERE TUTTE LE LETTERE DI UNA PAROLA

Il seguente programma legge in input una parola e scrive sulla lavagna tutte le lettere di cui è composta la parola, in ordine:

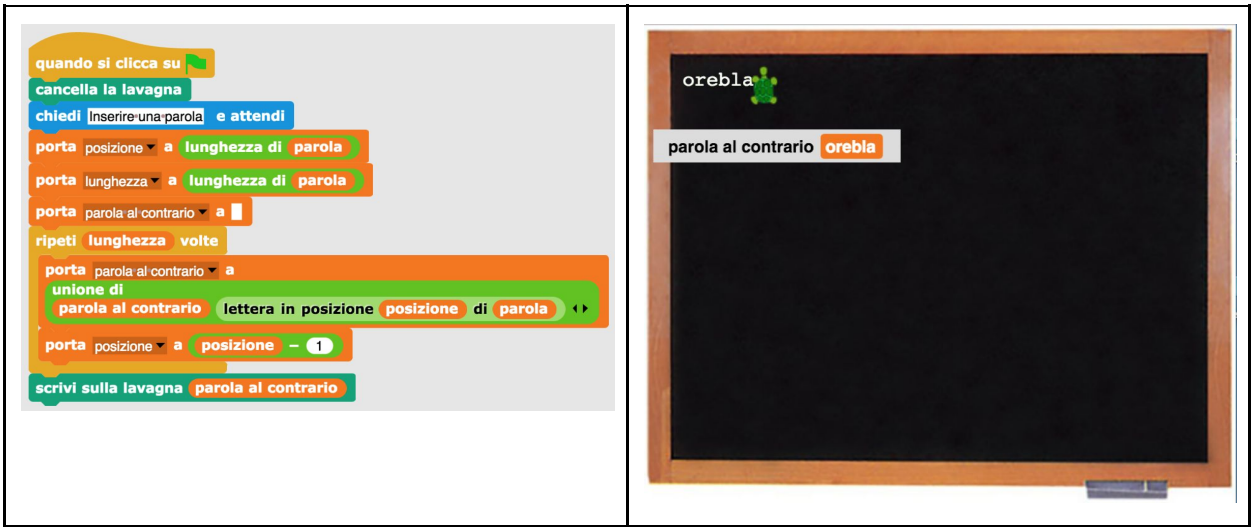
ALGORITMO #2: SCRIVERE TUTTE LE LETTERE DI UNA PAROLA **AL CONTRARIO**

Il seguente programma legge in input una parola e scrive sulla lavagna tutte le lettere di cui è composta la parola, in ordine:



ALGORITMO #3: CREARE LA PAROLA **AL CONTRARIO**

Il seguente programma legge in input una parola e scrive sulla lavagna la parola al contrario:



Bibliografia

[1] Seymour Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., 1980.

[2] Jeannette M Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35.

[3] A Csizmadia, P Curzon, M Dorling, S Humphreys, T Ng, C Selby, and J Woollard. Computational thinking: A guide for teachers. *Computing at Schools*, 2015.

[4] Michael Lodi Rita Marchignoli. *EAS e pensiero computazionale. Fare coding nella scuola primaria*. La Scuola, 2016.

[5] Massimo Capponi. *Un giocattolo per la mente. L'informazione cognitiva di Seymour Papert*. Morlacchi Editore, 2009.

[6] Marvin Minsky and Giuseppe Longo. *La società della mente*. Adelphi, 1990.

[7] Maria Montessori, Maria Montessori, Aerztin Anthropologin, Maria Montessori, Médecin Anthropologue, Pays-Bas Espagne, Maria Montessori, Physician Anthropologist, Italy Pedagogue, and Netherlands Spain. *La mente del bambino: mente assorbente*. Garzanti, 1992.

[8] Idit Ed Harel and Seymour Ed Papert. *Constructionism.* Ablex Publishing, 1991.

[9] Seymour Papert. *The children's machine: Rethinking school in the age of the computer*. Basic books, 1994.

[10] Seymour Papert. Bambini e adulti a scuola con il computer. 1997.

[11] Mitchel Resnick. Learn to code, code to learn. *EdSurge, May*, 2013.

[12] Mark Guzdial. Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics*, 8(6):1–165, 2015.

[13] Miles Berry. Computing in the national curriculum. a guide for primary teachers. *Computing at school*, 2013.

[14] John Stout. After scrath try snap! *Hello World, Jan*, 2017.