Insegnare il pensiero computazionale

strumenti, metodologie e riflessioni per la scuola

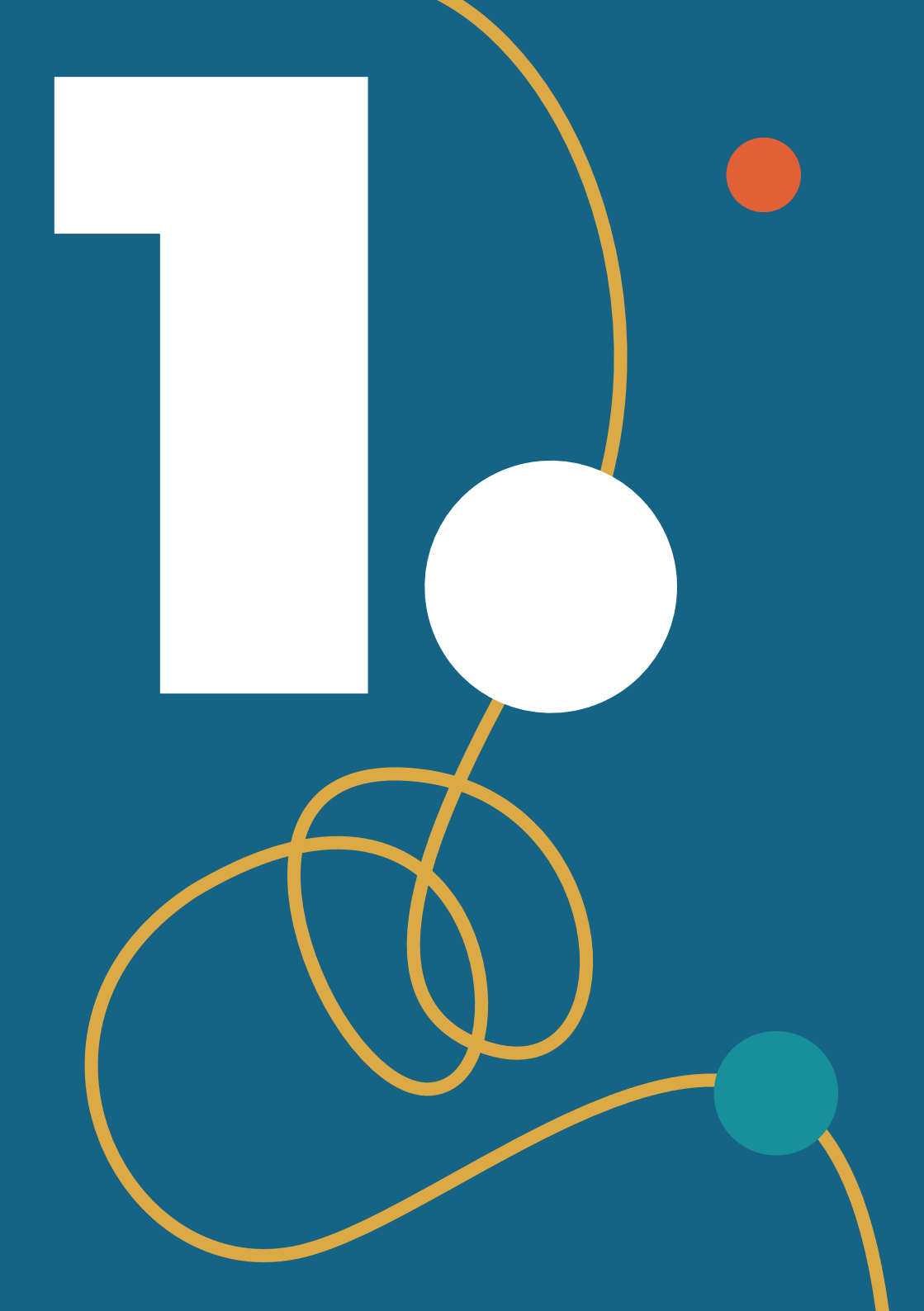
Co-funded by the European Union

Autori: Giorgia Bissoli
e Alberto Montresor

Questo lavoro è il frutto della nostra esperienza in anni di corsi di aggiornamento per insegnanti, a tutti i livelli scolastici, promossi dall'Università di Trento in collaborazione con gli enti del terzo settore Verona Fablab e Glow. Vuole accompagnare tali corsi dando una cornice teorica e pedagogica che spiega l'importanza di introdurre il pensiero computazionale nelle scuole, senza dimenticare di fornire esempi e spunti pratici per approfondire, partecipando ai nostri corsi o sfruttando le tante risorse che si trovano in rete.

Indice

1

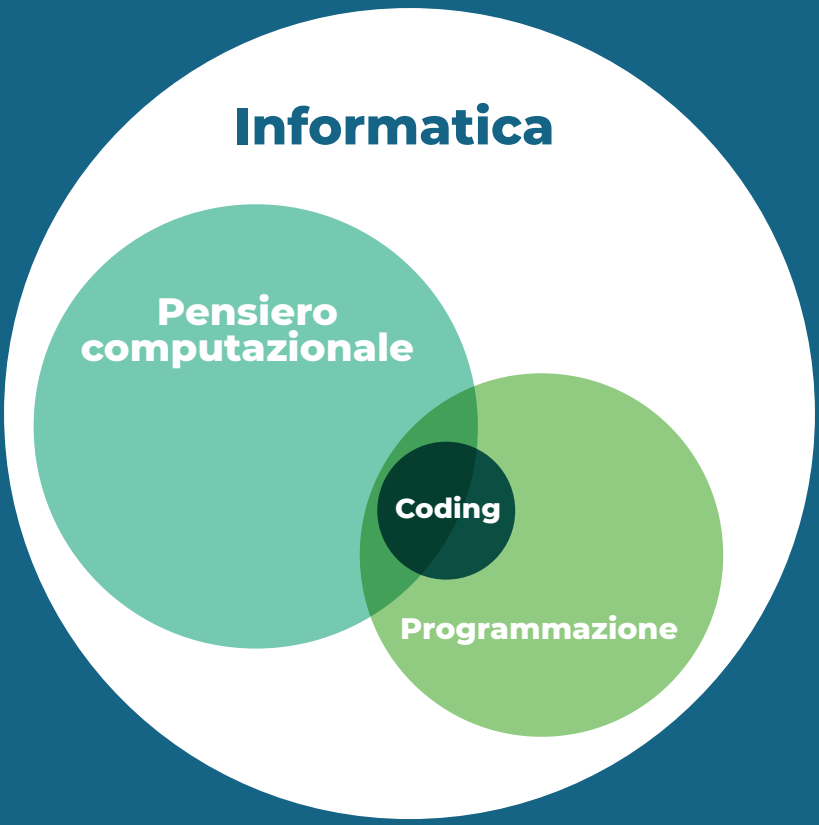


Introduzione

La società moderna è profondamente influenzata dalla tecnologia digitale e dalla cultura che ne deriva. Le discipline scientifiche e ingegneristiche si avvalgono dell'informatica per modellizzare e analizzare informazioni, ad esempio tramite la simulazione numerica di sistemi fisici, il sequenziamento del DNA e la progettazione digitale. Anche le arti visive sono state rivoluzionate dalle infinite possibilità offerte dalle tecnologie digitali. Infine, la crescita incessante dell'economia globale è sostenuta dalla capacità di interconnettere i mercati attraverso la rete.

Nonostante l'informatica sia alla base di molte innovazioni nella nostra società, non è ancora integrata nei programmi scolastici della maggior parte dei paesi europei. Purtroppo, la riforma dei curricula in alcuni paesi è stata lenta, con pochi progressi concreti oltre i proclami ufficiali. Tuttavia, si intravede una speranza di cambiamento grazie al recente **Digital Education Action Plan 2021-2027 (DEAP 2021)**, promosso dalla Commissione Europea, che **riconosce l'importanza dell'educazione informatica e la considera prioritaria come strumento per potenziare le abilità e le competenze necessarie per affrontare la rivoluzione digitale in corso.**

L'integrazione dell'**informatica** nei curricula è una sfida relativamente recente, oggetto di intense discussioni e ricerche. Pur essendo una disciplina scientifica indipendente e consolidata, il suo insegnamento richiede una comprensione precisa dei concetti da introdurre nelle scuole, del momento adeguato per farlo e delle modalità didattiche più efficaci. Negli ultimi anni, il termine "**pensiero computazionale**" (*computational thinking)* è diventato sempre più popolare come modo di descrivere l'insieme di competenze informatiche che tutti – non solo gli informatici – dovrebbero possedere per essere membri attivi e produttivi della società digitale. I sostenitori del pensiero computazionale sostengono che apprendere queste abilità possa aiutare gli studenti a diventare migliori risolutori di problemi, sia nella vita quotidiana sia nella loro futura carriera. Sebbene tali affermazioni possano sembrare ambiziose, cresce il consenso sull'importanza di integrare l'informatica e il pensiero computazionale nei curricula scolastici.


Informatica
Pensiero computazionale
Coding
Programmazione

Il documento di lavoro della Commissione Europea che accompagna il Piano d'Azione per l'Istruzione Digitale 2021-2027 fornisce una definizione di pensiero computazionale, evidenziandone le peculiarità e le differenze rispetto ad altri concetti affini. Secondo il documento,

"il pensiero computazionale, la programmazione e il coding sono spesso usati in modo intercambiabile nei contesti educativi, ma sono attività distinte. In particolare, la ***programmazione*** *si riferisce all'attività di analizzare un problema, progettare una soluzione e implementarla, mentre il* ***coding*** *significa implementare soluzioni in un particolare linguaggio di programmazione. D'altra parte, il* ***pensiero computazionale*** *si riferisce alla capacità di comprendere i concetti e i meccanismi alla base delle tecnologie digitali per formulare e risolvere problemi".*

Descrivere l'essenza del pensiero computazionale in poche righe non rende giustizia alla sua complessità intrinseca. Per comprenderne appieno la portata, è essenziale capire cosa significhi nella pratica e approfondire le teorie pedagogiche che lo supportano. Questo manuale ha l'ambizione di offrire proprio questo: una presentazione dei concetti fondamentali del pensiero computazionale, arricchita da esempi ed esperienze provenienti dal mondo scolastico. Inoltre, fornisce numerosi consigli pratici e riferimenti per ulteriori letture.

Si, ma...

Mentre scriviamo, immaginiamo il lettore che pensa:

🥱 Oddio, l'ennesima "moda" didattica; speriamo che sparisca presto!

😤 Il pensiero computazionale è solo un modo diverso per dire problem solving. La matematica si occupa del problem solving da... sempre, quindi non capisco di cosa stiamo parlando!

🤔 Io insegno scienze, non vedo la relazione con la mia disciplina!

😳 Io insegno materie umanistiche, la mia disciplina è del tutto estranea!

O forse riconosce l'importanza del pensiero computazionale, ma è pieno di dubbi e domande:

🧐 Faccio già fatica a coprire il mio programma, dove trovo il tempo per il pensiero computazionale?

🤔 Non ho alcuna formazione su questo argomento, non saprei da dove partire...

😞 Ho partecipato ad un corso di aggiornamento, è stato davvero bello e divertente, ma non capisco come portare queste idee in classe...

😰 Capisco l'importanza del pensiero computazionale, ho seguito corsi di formazione,ma non mi sento sicuro. Aspetterò finché non sarò pronto...

L'obiettivo di questo documento è dare una risposta a queste domande.
Forniamo qui un breve riassunto, una sorta di antipasto per il resto del documento.

Questa non è una moda!

Potresti percepire il pensiero computazionale come una moda recente, ma in realtà si parlava di didattica dell'informatica già negli anni '70. Tuttavia, all'epoca i computer erano poco diffusi. La società digitale è ADESSO, e dobbiamo preparare gli studenti per il loro futuro, non per il nostro presente.

Il pensiero computazionale è interdisciplinare

Introdurre il pensiero computazionale nei programmi di matematica e scienze può offrire agli studenti un modo divertente e coinvolgente di esplorare queste materie. Ad esempio, utilizzare il piano cartesiano per descrivere i movimenti dei personaggi di un gioco può aiutare gli studenti a visualizzare concetti matematici in modo più concreto. Le possibilità di integrazione sono moltissime e renderanno la tua materia più accessibile e stimolante!

È più semplice di quanto pensi!

Ci si può avvicinare al pensiero computazionale gradualmente, iniziando con un approccio senza computer e successivamente utilizzando software che permettono di imparare a programmare in modo visuale. Se questi strumenti sono adatti ai ragazzi, allora lo sono anche per noi – l'importante è buttarsi!

Problem solving ≠ Pensiero computazionale

Il problem solving e il pensiero computazionale non sono la stessa cosa. Il pensiero computazionale aggiunge una componente autoriflessiva alla risoluzione dei problemi, chiedendo di descrivere e formalizzare le soluzioni. Tuttavia, siamo d'accordo che il pensiero logico/matematico sia fondamentale per sviluppare il pensiero computazionale!

Riguarda anche le discipline umanistiche!

Le materie umanistiche possono svolgere un ruolo cruciale nello sviluppo del pensiero computazionale, poiché richiedono agli studenti di esprimere soluzioni in modo chiaro e comprensibile, sia per un computer che per un essere umano. Per farlo in modo efficace, è essenziale che gli studenti sviluppino solide capacità comunicative, sia scritte che orali. Per questo motivo, le materie umanistiche risultano fondamentali!

È un'occasione per innovare la tua didattica

Il pensiero computazionale è strettamente collegato a numerose teorie pedagogiche, come il costruttivismo (Piaget), il costruzionismo (Papert) e l'apprendimento creativo (Resnick). Alcuni dei loro principi possono arricchire la tua didattica in modi significativi, anche al di là dell'uso dei computer!


2

Organizzazione del manuale

Inizieremo con un'analisi su cosa sia il pensiero computazionale, partendo da un esempio concreto che evidenzi i concetti fondamentali correlati. Successivamente, proporremo una definizione pratica da adottare come "cartina di tornasole" per distinguere le attività che possono essere considerate pensiero computazionale da quelle che non lo sono.

Esamineremo quindi le **pratiche** e le **competenze trasversali** legate al pensiero computazionale. Alcuni di questi aspetti potrebbero sorprenderti! Ad esempio, l'informatica è spesso vista una disciplina poco inclusiva, a causa dello stereotipo del "nerd solitario" che lavora da solo in una stanza buia. In realtà, però, collaborazione e cooperazione tra individui sono aspetti fondamentali dell'informatica!

In seguito, esploreremo le **teorie pedagogiche** che hanno contribuito a dare forma al concetto di pensiero computazionale. Siamo consapevoli che la pedagogia viene spesso presentata in modo astratto, ma ci impegneremo a renderla quanto più concreta possibile, offrendo numerosi esempi pratici!

Infine, esamineremo alcuni modi in cui il pensiero computazionale può essere portato in classe.

1.

Inizieremo con il **Tinkering**, un approccio educativo che promuove l'apprendimento attraverso l'esplorazione, la sperimentazione e la costruzione creativa. Tinkering significa "imparare facendo e riprovando," utilizzando le mani per comprendere concetti scientifici, tecnologici o artistici. Vedremo esempi adatti alla scuola primaria, ma questo approccio può essere efficacemente applicato anche ai livelli scolastici successivi.

2.

Il passo successivo riguarda le attività **Unplugged**. L'idea è semplice: i concetti fondamentali dell'informatica possono essere insegnati anche con carta, penna e lavagna. Questa metodologia è adatta a partire dal secondo anno della scuola primaria, ma poiché incoraggia a riflettere sui principi alla base dell'informatica, è valida per tutte le età, fino alla scuola superiore.

3.

Successivamente, esploreremo i concetti relativi alla programmazione utilizzando esempi basati su **Scratch**. Non aspettarti un manuale con istruzioni ed esercizi pronti per la classe: il nostro obiettivo è illustrare i concetti chiave e il loro legame con le pratiche pedagogiche descritte in precedenza. Troverai comunque numerosi riferimenti utili per avere esempi pratici e spunti su come iniziare.

4.

Infine, daremo uno sguardo alla **connessione tra il pensiero computazionale e altre discipline**, discutendo di narrazione, scienza, matematica, e così via.

Il manuale si conclude con una lettera scritta da una docente che sta cercando di introdurre il pensiero computazionale nella sua classe. È una persona che ammette apertamente i suoi limiti, ma riconosce l'importanza e l'impatto che il pensiero computazionale può avere nell'educazione dei suoi studenti e butta il cuore oltre l'ostacolo!

COME SI LEGGE IL MANUALE?

Devi leggerlo due volte:

la prima volta come punto di partenza per orientarti;

la seconda dopo averlo sperimentato nella pratica didattica, per comprendere la portata pedagogica del pensiero computazionale.

Pensiero computazionale

Il termine pensiero computazionale è stato usato per la prima volta nel 1980 da Seymour Papert, un matematico, informatico, ma soprattutto educatore e pedagogista, il primo a studiare l'impatto delle nuove tecnologie sull'apprendimento.
Nei suoi scritti, Papert usò la locuzione pensiero computazionale en passant, senza dare una definizione precisa. Il termine fu portato alla ribalta nel 2006 da Jeannette Wing, professoressa di informatica all Carnegie Mellon University, che in un articolo su Communications of the ACM (la rivista della principale associazione professionale di informatici) proponeva di insegnare il "modo di pensare degli informatici" in tutte le scuole. Da allora, sono stati scritti innumerevoli articoli nel tentativo di offrire una definizione più precisa, senza tuttavia giungere a un consenso unanime. Tuttavia, molte definizioni proposte concordano sul fatto che il pensiero computazionale non si limiti a metodi e pratiche tecniche, ma includa soprattutto processi mentali e competenze trasversali come creatività, collaborazione, tolleranza dell'ambiguità, resilienza e molto altro.

3.1 Un esempio

Per comprendere meglio il concetto di pensiero computazionale, vi consigliamo di guardare il video "Exact Instruction Challenge", di Josh Darnit, su Youtube.
Josh propone ai suoi figli, Johanna ed Evan, una sfida: scrivere una lista di istruzioni per preparare un sandwich al burro di arachidi e marmellata. Una volta scritte, Josh le eseguirà alla lettera.

I ragazzi iniziano con una serie di istruzioni incomplete, che non permettono di raggiungere l'obiettivo, e man mano le perfezionano. Dopo alcuni tentativi, Johanna riesce nell'impresa, mentre Evan, più piccolo, si arrabbia.

Particolarmente interessanti sono alcuni momenti del video:

1.
Evan ha scritto: "Prendi una fetta di pane e spalmala in giro con il coltello." In queste istruzioni, però, non viene menzionato il burro di arachidi. Quando Josh glielo fa notare, Evan esclama "Hold on" (aspetta!), e sul suo volto emerge la consapevolezza che le istruzioni non sono complete.

2.
Johanna ha scritto: "Con il coltello, prendi un po' di burro di arachidi," e, quando sente il padre leggere la frase ad alta voce, commenta: "Un po' significa un sacco." Alla domanda sorpresa di Josh, "Un po' significa un sacco?", lei risponde: "Nel mio mondo."

3.
Più avanti, Josh legge l'istruzione: "Inserisci il coltello nel barattolo" e lo inserisce con la lama rivolta verso l'alto. Lo stupore e il disappunto si leggono sul volto di Johanna: dopo vari tentativi, pensava di aver formulato la soluzione in modo preciso, ma si rende conto che c'era ancora un passaggio ambiguo.

3.2 Alcuni aspetti chiave del pensiero computazionale

Usiamo il video per introdurre alcuni aspetti chiave del pensiero computazionale.

Riflettere sulla soluzione dei problemi

Josh non chiede ai propri figli di preparare un panino al burro di arachidi e marmellata, ma propone loro la sfida di descrivere una soluzione (una ricetta) al problema di preparare questo tipo di panini. Non chiede, quindi, di risolvere una particolare **istanza del problema** (fammi un panino), ma di descrivere una **soluzione generale**. Questo sposta il problem solving ad un "livello superiore": non basta **risolvere il problema**, ma bisogna pensare al procedimento per risolverlo.

Questa è la differenza tra il pensiero computazionale e il semplice problem solving: sviluppare la capacità di astrarre la soluzione in modo che sia applicabile a una classe di problemi, una competenza trasferibile a ogni campo del sapere.

Esprimere una soluzione

In questa sfida, descrivere una soluzione (una ricetta) richiede di **mettersi nei panni dell'altro da sé**. È necessario scomporre la soluzione in una sequenza di passi elementari che possano essere compresi, in modo non ambiguo, dall'esecutore. Quando si descrive un procedimento (o qualsiasi argomento) è fuorviante fare riferimento al "proprio mondo", come fa Johanna; bisogna invece avere un'idea chiara dei **passi elementari** (e dei concetti) che l'esecutore è in grado di comprendere.
In altre parole, è necessario considerare il "livello di comprensione" dell'esecutore. Il padre "fa il finto tonto", cercando deliberatamente i "bug" nella descrizione. Non avendo compreso pienamente le regole del gioco, Evan si arrabbia perché sa che l'esecutore è intelligente e conosce già la soluzione; perciò si aspetta che il padre interpreti le sue istruzioni, anche se ambigue.

Nel contesto del pensiero computazionale, il computer è l'**esecutore più "stupido" di tutti**: conosce solo pochissimi passi elementari, ma ha il vantaggio di eseguirli sempre allo stesso modo, senza ambiguità. Spiegare un procedimento a un computer richiede quindi una comprensione estremamente precisa del problema e della soluzione, favorendo l'abitudine a considerare il livello di comprensione del proprio interlocutore.

Decomporre il problema e le sue soluzioni

Nel video, si nota che Johanna individua subito la necessità di **scomporre il problema** in due parti: imburrare una fetta e spalmare la marmellata nell'altra, per poi **ricomporre le soluzioni parziali** nel panino completo. Evan, più giovane, all'inizio fa più fatica; nella sua seconda soluzione, infatti, viene fuori un pasticcio, poiché il problema non è stato scomposto in modo efficace.

È interessante notare che la necessità di decomporre problemi in sotto-problemi non è un'esigenza del computer, ma serve al cervello umano per affrontare una complessità che va oltre alle sue capacità cognitive. In questo senso, riuscire a mettere in atto tale processo decomposizione è una competenza fondamentale per chiunque.

Ottenere un feedback dai propri errori

Ogni volta che la ricetta viene "eseguita", i ragazzi ricevono un **feedback** sulla loro soluzione: osservano che alcune parti funzionano, mentre altre richiedono ulteriori aggiustamenti. La possibilità di testare e migliorare le proprie soluzioni è un aspetto cruciale di questa sfida, poiché consente ai ragazzi di imparare dai propri errori. Senza l'opportunità di imparare dagli errori, infatti, non può esserci un vero apprendimento.

In informatica, questa attività è nota come **debugging**; ancora una volta, la capacità di analizzare i propri errori è una competenza fondamentale in ogni campo del sapere, dalla matematica alla letteratura. In informatica, tuttavia, il debugging è un processo più immediato, poiché il feedback arriva subito e gli studenti possono rapidamente verificare se il loro programma funziona come previsto e apportare correzioni. Al contrario, stabilire se un'espressione matematica sia corretta non è sempre semplice: ad esempio, se il risultato di un'espressione è -113/11, come si può dire con certezza se è corretto o meno?

DOMANDE PER GLI INSEGNANTI DELLE MATERIE SCIENTIFICHE

Pensa a quando proponi ai tuoi studenti di risolvere un problema:

- **Dai loro la possibilità di sviluppare e descrivere un proprio procedimento di soluzione?**
- **Oppure devono seguire passo-passo le istruzioni che hanno ricevuto da te?**
- **Dai loro la possibilità di spiegare il procedimento ad un compagno?**
- **Li incoraggi a scomporre i problemi in parti indipendenti?**
- **Che tipo di feedback dai loro quando sbagliano?**

Domande per gli insegnanti delle materie umanistiche

I concetti di problema e soluzione sembrano specifici delle materie scientifiche. Ma nel pensiero computazionale gli aspetti linguistici e sociali sono estremamente importanti: saper esprimere una soluzione "per l'altro" è fondamentale, vuoi che sia un computer, vuoi che sia un essere umano.

- **Hai mai evidenziato l'importanza dell'interlocutore nella composizione scritta e nel parlato?**
- **Hai mai chiesto ad uno studente di fornire feedback sul lavoro di un altro?**

3.3 Una definizione del pensiero computazionale

Dal 2006 sono state proposte numerose definizioni di pensiero computazionale. La definizione che proponiamo si basa sui contributi di Jeannette Wing e Alfred Aho:

Il pensiero computazionale è il processo mentale coinvolto nella formulazione di un problema e nell'esprimere le sue soluzioni in una forma che può essere efficacemente eseguita da un esecutore di elaborazione delle informazioni (umano o macchina).

Quanti concetti importanti in questa frase! Esploriamoli con un esempio concreto (questa è una storia vera!). Stai viaggiando a New York con i tuoi figli, Federica e Francesco, e hai bisogno di trovare un percorso sulla mappa della metropolitana per spostarti da un punto all'altro della città: come fare? Questo problema rappresenta un'ottima opportunità educativa e, anche se può richiedere un po' di tempo, puoi collaborare con i tuoi figli per risolverlo. Non è un compito semplice, data la complessità della rete metropolitana, ma rispetto a quelle di Londra e Parigi potrebbe risultare fattibile!

Processo mentale

Non si tratta solo di trovare il percorso migliore per questo viaggio specifico l'obiettivo è che i ragazzi sviluppino un **approccio sistematico** per trovare il percorso migliore.
Mentre i tuoi bambini guardano la mappa, incoraggiarli a **pensare ad alta voce** oppure, se trovano difficoltà, verbalizza tu stesso il processo mentre sviluppi una strategia per risolvere il problema.
Rendere "visibili" i processi mentali è il modo migliore per abituare al ragionamento!

? **Chiedi mai ai tuoi studenti di rendere visibile il loro modo di pensare?**

? **E tu? Rendi mai visibile ai tuoi studenti il tuo modo di pensare?**

Formulazione del problema

Che tipo di percorso stiamo cercando: il più corto, il più veloce, quello con il minor numero di cambi, o quello con la minor distanza a piedi? È importante **definire chiaramente il nostro obiettivo**, tenendo presente che in molte situazioni ci sono tante possibilità. Dobbiamo creare più opportunità per gli studenti di **definire i propri obiettivi in maniera autonoma**.

? **Presenti mai agli studenti scenari aperti, in cui devono definire i loro obiettivi?**

Cercare una soluzione

Una volta definito il nostro obiettivo, dobbiamo capire come individuare il percorso migliore in base a quel criterio. Se fossimo informatici, useremmo l'algoritmo di Dijkstra, sviluppato nel 1956 e ancora oggi ampiamente utilizzato per la sua efficienza. Ma il nostro scopo è diverso: vogliamo trovare un **metodo sistematico per esplorare vari percorsi** e valutarli rispetto al criterio scelto. Forse ci vorrà un po' più di tempo, ma cercheremo di esaminare tutte le possibili soluzioni per scegliere quella che preferiamo.

? **Quando proponi agli studenti di risolvere un problema, chiedi mai loro di elencare diverse soluzioni e confrontarle tra loro?**

Nello scrivere questo testo, siamo stati tentati di fornire subito una soluzione, "istruendo" il lettore. Tuttavia, sarebbe stato un errore. Come una volta disse Seymour Papert:
"Lo scandalo dell'educazione è che ogni volta che insegni qualcosa, privi uno studente del piacere e del beneficio della scoperta".

Senza arrivare all'estremo di una classe completamente non strutturata, quest'altra citazione di Papert chiarisce meglio come impostare il lavoro di insegnamento:
"Il ruolo dell'insegnante è quello di creare le condizioni per la scoperta piuttosto che fornire conoscenze già pronte".

PRIMA DI CONTINUARE...

Come risolveresti il problema del percorso in metropolitana a New York?
Scrivi qui la tua soluzione:

Esprimere una soluzione

Come detto in precedenza, vogliamo concentrarci sul processo di risoluzione dei problemi, piuttosto che limitarci a fornire una soluzione. Consideriamo due criteri per trovare il percorso migliore: un massimo di un cambio di treno e la minimizzazione della distanza da percorrere a piedi. Ecco un esempio di approccio passo-passo:

- Localizza il punto di partenza e scegli la destinazione finale.
- Identifica due o tre stazioni vicine al punto di partenza.
- Prima proviamo con le linee dirette: per ogni stazione:
  • segui le linee metro che passano attraverso di essa nella direzione della destinazione finale;
  • cerca le stazioni più vicine alla tua destinazione finale;
  • valuta la distanza a piedi per ogni percorso.
- Ripeti il processo, considerando la possibilità di effettuare un cambio:
  • ogni volta che incontri una nuova linea ferroviaria, seguila nella direzione della destinazione finale.

? Ti convince la procedura descritta?

? Puoi provare a descrivere la tua soluzione nello stile utilizzato sopra?

? Rileggi ora la nostra soluzione: puoi provare a migliorarla?

L'esecutore

Il punto fondamentale è questo: non si tratta subito di scrivere un programma per computer, ma di **riflettere su chi leggerà la nostra soluzione e su come interpreterà le istruzioni**. Se ci rivolgiamo a una persona, è essenziale pensare a chi eseguirà il compito, che sia semplice o complesso. Se, invece, lavoriamo con un computer, abbiamo a disposizione solo un insieme limitato di istruzioni, ciascuna delle quali svolge un'azione semplice. Sta quindi a noi scomporre un'attività complessa in una sequenza di passi elementari. Coinvolgere gli studenti in attività simili li aiuta a capire l'importanza di esprimersi in modo chiaro e preciso.

Il ruolo dell'esecutore è fondamentale: potremmo dire che lo stile informatico di risoluzione dei problemi implica sempre un agente esterno, a cui bisogna "spiegare" la procedura per la soluzione.

3.4 Altre definizioni

Esistono altre definizioni di pensiero computazionale, ma spesso risultano complesse e poco pratiche. Queste tendono a concentrarsi sui metodi, ossia sugli approcci operativi comunemente utilizzati dai professionisti dell'informatica. La tabella seguente, tratta da un documento della Commissione Europea sul pensiero computazionale, elenca alcuni degli aspetti ad esso legati presenti nella letteratura.

Pensiero computazionale associato al problem solving generico

- Astrazione
- Raccolta dei dati
- Rappresentazione dei dati
- Analisi dei dati
- Decomposizione
- Efficienza
- Valutazione
- Generalizzazione
- Logica
- Riconoscimento di pattern
- Ripetizione di pattern
- Simulazione
- Pensiero sistemico
- Visualizzazione

Pensiero computazionale associato al calcolo e alla programmazione

- Pensiero algoritmico
- Progettazione algoritmi
- Automazione
- Logica booleana
- Calcolo automatico
- Modellazione computazionale
- Condizioni
- Tipi di dati
- Eventi
- Funzioni
- Iterazioni
- Cicli e ripetizioni
- Modularità
- Parallelismo
- Sequenze
- Testare e debuggare

Sai dare definizioni operative dei concetti elencati sopra? Se non hai esperienza con il pensiero computazionale, è probabile che la risposta alla domanda precedente sia "no" per la maggior parte dei concetti. E se hai già praticato attività di pensiero computazionale, la risposta potrebbe comunque essere ancora "no."

È un problema? Ancora una volta "no"! Le definizioni sono solo una parte della comprensione del pensiero computazionale, e spesso è molto più efficace discuterne attraverso esempi concreti piuttosto che cercare di spiegarlo con definizioni astratte. Ecco perché ci concentreremo su attività ed esempi pratici, per rendere questi concetti intuitivi e applicabili.

? Ripensa all'esempio della metro di New York e, prima di proseguire nella lettura, mettiti alla prova: riesci a identificare "Astrazione", "Decomposizione", "Valutazione", "Condizioni" e "Cicli e ripetizioni"?

Ecco le nostre risposte:

- Le mappe in generale, e le mappe della metropolitana in particolare, sono ottimi esempi di **astrazione**: un territorio complesso è rappresentato in modo molto astratto.

- Abbiamo **decomposto** il problema in due sotto-problemi: la ricerca di connessioni dirette e la ricerca di percorsi con un cambio.

- Ogni nuovo percorso che viene scoperto viene **valutato** secondo alcuni criteri.

- Se un nuovo percorso è migliore del miglior percorso precedente, allora ricordiamo il nuovo. Il *se-allora* è una **condizione**.

- **Ripetiamo** l'operazione su ogni possibile percorso, per ogni possibile intersezione.

Questa introduzione al pensiero computazionale è un buon punto di partenza. Ora è il momento di riassumere i concetti chiave e proporre un insieme di linee guida pratiche per integrare queste idee in classe, a prescindere dalla materia.

Pensiero computazionale

1.

Dai agli studenti tempo sufficiente per riflettere sul problema, permettendo loro di discuterlo e di esplorarne diverse varianti. Evita di presentare il problema in modo diretto; lascia invece spazio alla loro curiosità, affinché possano formulare domande e ipotesi. Questo approccio li aiuterà a sviluppare una comprensione più profonda e personale del problema e a generare soluzioni creative.

2.

Concedi agli studenti il tempo necessario per affrontare il problema in autonomia, senza offrire soluzioni già pronte. Incoraggiali a discutere le loro idee e a confrontare le soluzioni, valutando quali siano più efficaci in base ai criteri e alle preferenze che loro stessi hanno stabilito. Questo approccio li aiuta a sviluppare capacità di valutazione critica e di auto-riflessione, rafforzando la fiducia nelle loro capacità di risoluzione di problemi.

3.

Incoraggia gli studenti a spiegarsi reciprocamente le proprie soluzioni, sia che siano scritte su carta sia sotto forma di programmi informatici. Questo scambio di idee li aiuta a chiarire il proprio pensiero, a identificare eventuali punti deboli nelle loro soluzioni e a sviluppare competenze comunicative essenziali.

3.5 Buone pratiche e competenze trasversali

Oltre a rappresentare un metodo per sviluppare abilità di risoluzione problemi, molte definizioni di pensiero computazionale includono anche “buone pratiche” e “competenze trasversali” che esso tipicamente promuove.

Buone pratiche

- **Sperimentare, iterare, riprovare**: nello sviluppo di un progetto informatico, l’approccio tipico è basato sulla costruzione passo dopo passo, attraverso ripetute fasi di progettazione, creazione e verifica. Questo processo è riassunto dalla parola tinkering, che significa progettare attraverso un processo di tentativi ed errori, esplorando e sperimentando.

- **Testare e correggere**: verificare se le soluzioni funzionano mettendole alla prova; identificare e risolvere eventuali problemi (o bug) in una soluzione.

- **Riutilizzare e remixare**: creare le proprie soluzioni partendo da progetti, idee o materiali già esistenti, costruendo su basi già sperimentate.

Competenze trasversali

- **Creare**: progettare e costruire progetti, hardware o software, utilizzando l’informatica come strumento per esprimere creatività e individualità.

- **Comunicare e collaborare**: connettersi con gli altri e lavorare insieme verso un obiettivo comune, migliorando le soluzioni attraverso la condivisione e il lavoro di gruppo.

- **Riflettere, apprendere, meta-riflettere**: usare l’informatica per riflettere su processi e per comprendere gli aspetti computazionali che caratterizzano il mondo.

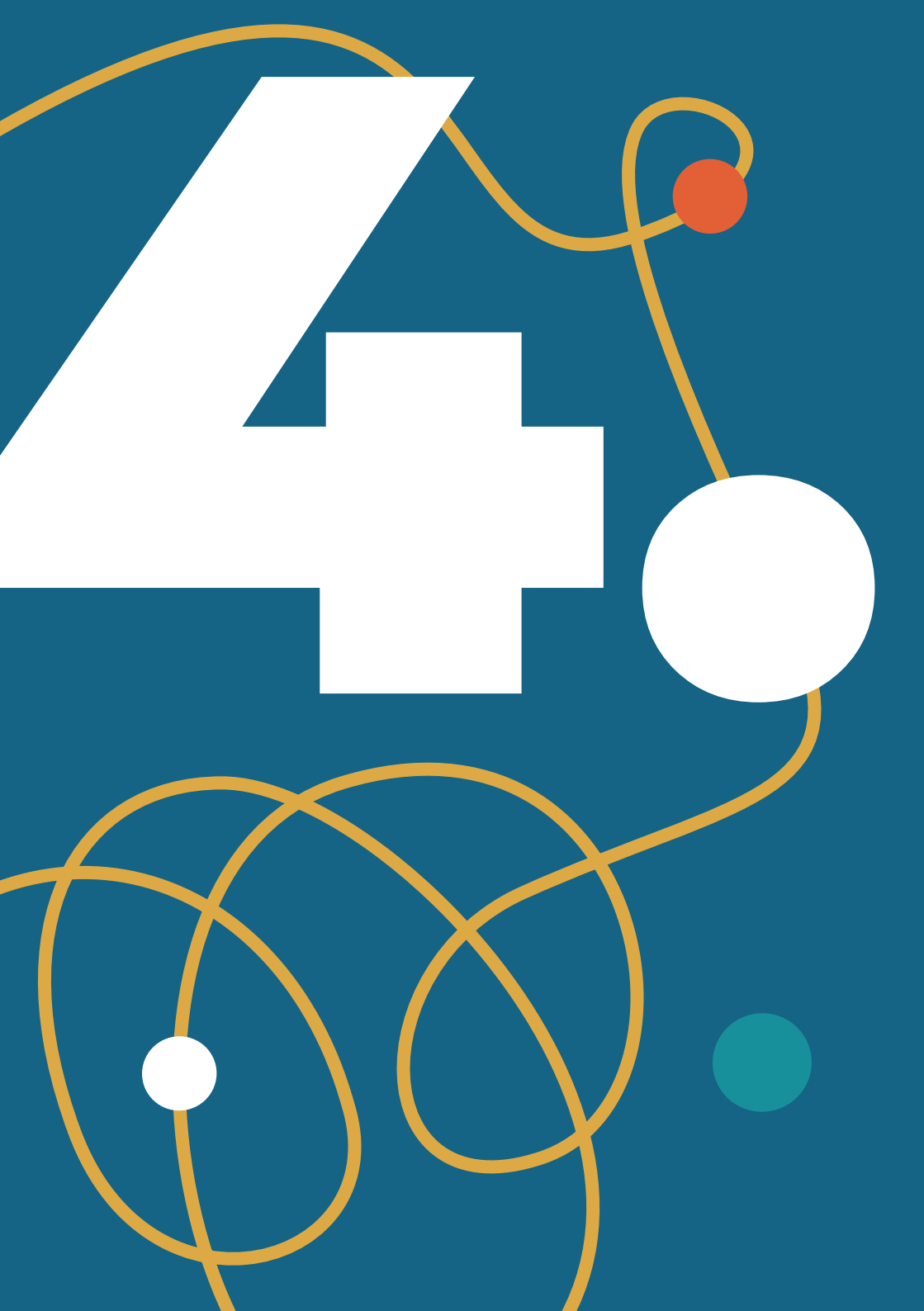
- **Essere tolleranti alle ambiguità**: saper affrontare problemi reali, che spesso sono aperti e poco definiti.

- **Essere persistenti**: sviluppare fiducia e determinazione nel lavorare su sfide difficili, perseverando con resilienza e tenacia.

Approfondimenti

Michael Lodi, Simone Martini, Enrico Nardelli. " *Abbiamo davvero bisogno del pensiero computazionale?*". Mondo Digitale, 16(72), 2017

Michael Lodi, "*Pensiero Computazionale: dalle "scuole di samba della computazione" ai CoderDojo*". DIDAMATICA 2018.



I fondamenti pedagogici

Per capire quali sono i fondamenti teorici del pensiero computazionale, dobbiamo introdurre alcune teorie pedagogiche, ognuna delle quali fornisce un tassello del pensiero computazionale. Andremo dal più teorico al più pratico.

4.1 Costruttivismo e Piaget

Secondo la teoria del **costruttivismo** di Piaget, **l'individuo che impara costruisce modelli mentali per comprendere il mondo che lo circonda**; contrariamente all'istruttivismo, vede **l'apprendimento come una ricostruzione piuttosto che una trasmissione di conoscenza**.

Nella sua teoria, due processi chiamati **assimilazione** e **accomodamento** si alternano continuamente nel corso della vita degli individui. L'**assimilazione** è il processo che ci permette di acquisire costantemente nuove conoscenze, esperienze, informazioni che collochiamo dentro di noi in schemi progressivamente sempre più complessi. L'**accomodamento**, invece, è la continua modifica di quegli stessi schemi per poter giungere ad una visione sempre più coerente ed in "equilibrio" rispetto ai nuovi saperi assimilati. Piaget incoraggiava:

"L'uso di metodi attivi che danno ampio spazio alla ricerca spontanea del bambino o dell'adolescente e richiedono che ogni nuova verità da apprendere sia riscoperta o almeno ricostruita dallo studente, e non semplicemente impartitagli".

4.2 Costruzionismo e Papert

Papert elabora il costruttivismo di Piaget nella sua teoria dell'apprendimento, il **costruzionismo**. Condivide con il costruttivismo l'idea della costruzione attiva della conoscenza attraverso l'esperienza, ma aggiunge un elemento fondamentale: l'apprendimento è particolarmente efficace quando chi apprende è coinvolto nella creazione attiva di **artefatti significativi** per lui o per la comunità di cui fa parte. Questa costruzione richiede materiali, che possono essere **concreti** o **astratti**, "**oggetti con cui pensare**" che facilitano la costruzione di modelli mentali.

Nell'introduzione del suo libro più famoso, Mindstorms, Papert afferma che da bambino era affascinato dagli ingranaggi – smontava qualunque oggetto meccanico per capirne il funzionamento. Il suo modello mentale sugli ingranaggi gli ha permesso di comprendere concetti matematici più complessi, come le proporzioni, le frazioni, fino alle equazioni differenziali:

"Credo che il tempo passato a giocare con gli ingranaggi sia stato più efficace per la mia comprensione della matematica di quanto lo siano stati tutti gli insegnamenti ricevuti alla scuola elementare. Gli ingranaggi, fungendo da modelli concreti, hanno permesso alla mia mente di assimilare idee che altrimenti sarebbero rimaste astratte".

Nella scelta di questi "oggetti con cui pensare", però, non c'è solo un aspetto conoscitivo: è sempre in gioco una fondamentale **componente affettiva**. La passione di Papert per gli ingranaggi è personale e non riproducibile.

"Una Maria Montessori dei nostri giorni potrebbe, se convinta dalla mia storia, proporre di creare un set di ingranaggi per bambini, affinché possano vivere la mia stessa esperienza. Tuttavia, nutrire questa speranza significherebbe trascurare l'essenza della storia: io mi ero innamorato degli ingranaggi. Questo è qualcosa che non può essere ridotto a una semplice questione cognitiva".

? Quali sono gli ingranaggi della tua infanzia?

? Nella tua esperienza di apprendimento, c'è qualche oggetto che ha plasmato la tua comprensione del mondo? Qualcosa per cui hai avuto o hai ancora una vera passione?

Il computer, per Papert, diventa il più flessibile fra questi "oggetti con cui pensare", una sorta di oggetto universale, in grado di rappresentare e simulare qualsiasi sistema o concetto. Come una volta gli ingranaggi hanno offerto a lui stesso una comprensione intuitiva di concetti complessi, ora **il computer può fornire a ogni studente un insieme flessibile di strumenti con cui esplorare e comprendere idee astratte**. Il computer non è quindi solo un mezzo per eseguire calcoli o simulazioni, ma uno spazio di esplorazione in cui gli studenti possono creare e modificare in prima persona.

"La mia tesi può essere sintetizzata così: ciò che gli ingranaggi non possono fare, il computer può. Il computer è il Proteo delle macchine, con un'essenza radicata nella sua universalità e nel suo potere di simulazione. Essendo in grado di assumere migliaia di forme e di svolgere migliaia di funzioni, può soddisfare una vasta gamma di interessi"

Nel costruzionismo di Papert, l'obiettivo è **ribaltare il processo tradizionale di insegnamento**, passando da un approccio in cui il docente trasferisce conoscenze a uno in cui lo studente diventa protagonista del proprio apprendimento. Invece di fornire informazioni già pronte, **l'insegnante ha il compito di creare condizioni che favoriscano la scoperta**. Per il costruzionismo, quindi, l'apprendimento è un autentico processo esplorativo, in cui il ruolo dell'insegnante è quello di offrire agli studenti esperienze e strumenti che consentano loro di approfondire concetti e costruire attivamente la propria conoscenza. Il computer ha anche un altro ruolo nell'apprendimento: consente agli studenti di vedere immediatamente il risultato delle proprie azioni - ogni comando o modifica genera una risposta visibile e immediata, che stimola ulteriori tentativi e incoraggia l'auto-correzione. Questo ciclo di azione-feedback-miglioramento rende l'apprendimento più efficace, permettendo agli studenti di sperimentare. In questo senso, il **feedback immediato** del computer diventa un potente "oggetto con cui pensare", uno strumento che aiuta gli studenti a interiorizzare e applicare concetti in un processo di scoperta continua. Questo aspetto rende l'uso del computer particolarmente adatto alla pedagogia costruzionista, favorendo un apprendimento in cui il discente può iterare, riflettere e progredire autonomamente nel proprio percorso di scoperta.

È più facile spiegare il costruzionismo partendo da un documento di Gary Stager, allievo di Papert, che descrive le “8 grandi idee del costruzionismo”.

Rielaboriamo ogni regola descrivendola sia dal punto di vista del docente che dello studente.

1.

Lascia spazio alla scoperta personale
Impara facendo

Per imparare, non c’è metodo migliore che “fare” qualcosa: lavorare a una sfida, a un progetto, alla costruzione di un artefatto. Apprendiamo al meglio quando usiamo ciò che apprendiamo per creare qualcosa di significativo per noi.

2.

Lasciali esprimere attraverso la tecnologia
Usa la tecnologia per esprimerti

Se usata bene, **la tecnologia è una nuova forma espressiva**: permette di essere creativi e di creare artefatti più interessanti. Questo è particolarmente vero per le tecnologie digitali, che permettono di rendere interattivo quanto creiamo.

3.

Sfidali al punto giusto
Divertiti imparando (hard fun)

Apprendiamo meglio e lavoriamo meglio se ci piace cosa stiamo facendo e ci stiamo divertendo. Ma piacere e divertimento non significano “facilità”. Il divertimento migliore è il divertimento impegnativo (**hard fun**) – lo puoi riconoscere quando vedi che un tuo studente non vuole smettere quello che sta facendo e che alla fine dell’attività dice di essersi divertito nonostante la difficoltà!

4.

Stimola gli studenti a riflettere sul proprio apprendimento
Impara ad imparare

Molti studenti hanno l’idea che l’unico modo per imparare è essere “insegnati”. Questo è ciò che li fa fallire a scuola e nella vita. **Nessuno può insegnarti tutto ciò che devi sapere**. Devi prendere il controllo del tuo apprendimento.

5.

Lascia loro abbastanza tempo
Prenditi il tempo necessario

Molti studenti a scuola sono abituati a essere controllati. Ogni pochi minuti, gli si dice: fai questo, poi fai quello, ora fai la cosa successiva. Se qualcuno non gli dice cosa fare, si bloccano. La vita non è così. Per fare qualcosa di importante **devi imparare a gestire il tuo tempo**. Questa è la lezione più difficile.

6.

Celebra l’errore
Non puoi avere successo senza sbagliare

Non si può ottenere il risultato giusto senza sbagliare. Nulla di impor-

tante funziona la prima volta. L'unico modo per ottenere il risultato giusto è guardare attentamente cosa è successo quando qualcosa è andato storto. Per avere successo, hai bisogno della libertà di fare errori lungo il percorso. Come insegnanti, dobbiamo avere il coraggio di cogliere la natura educativa degli errori, mostrandoli senza reprimerli.

7.

Fai da esempio

Non smettere mai di imparare

Anche se siamo insegnanti, non smettiamo mai di imparare. Imparare è difficile e faticoso; non è un processo lineare, anzi: è pieno di errori, deviazioni, ripensamenti. Ogni difficoltà che incontriamo è un'opportunità per imparare. La lezione più importante che possiamo dare ai nostri studenti è renderli consapevoli che anche noi insegnanti impariamo, e **rendere visibile il processo di apprendimento**.

8.

Riconosci che gli studenti devono apprendere in un mondo diverso dal nostro

Preparati per un mondo digitale.

Oggi, conoscere la tecnologia è tanto importante quanto leggere e scrivere. Quindi, imparare come funzionano i computer è essenziale per il futuro dei nostri studenti, ma lo scopo più importante è usarli per imparare tutto il resto!

? **Pensa al tuo essere insegnante: ti vengono in mente alcuni modi, anche inconsapevoli, in cui hai applicato le otto regole del costruzionismo? Puoi descrivere come?**

Approfondimenti

- *Seymour Papert. "Le otto grandi idee del costruzionismo di Papert".*
- *Seymour Papert. "Gli ingranaggi della mia infanzia".*
- *Seymour Papert. "Hard Fun".*
- *Seymour Papert. "Mindstorms: Bambini, computer e creatività". 1980. Versione gratuita in Inglese.*

4.3 Apprendimento creativo e Resnick

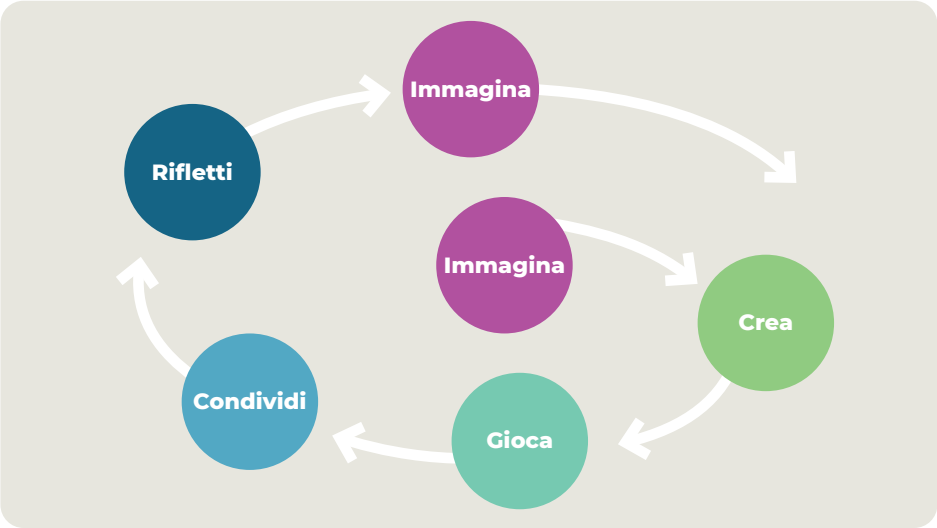
L'Apprendimento Creativo è un approccio metodologico introdotto da Mitch Resnick, coordinatore del gruppo Lifelong Kindergarten presso il MIT Media Lab. Come suggerisce il nome del laboratorio, l'obiettivo è indagare su come le persone imparano durante i primi anni e applicare questo metodo di apprendimento per tutta la vita. All'asilo, i bambini esplorano il mondo e acquisiscono conoscenze manipolando oggetti, sperimentando, costruendo cose e verificando la loro funzionalità. Ragionano creando prototipi e identificando errori, che li aiutano a sviluppare conoscenze delle leggi fondamentali dell'ambiente in cui vivono. Secondo Resnick, questo stile di apprendimento è essenziale per aiutare persone di qualsiasi età a sviluppare le competenze creative necessarie per prosperare nella società in rapida evoluzione di oggi. All'interno del laboratorio stesso, questo metodo viene utilizzato per realizzare obiettivi di ricerca e per questa ragione ha guadagnato una reputazione internazionale per la creatività e l'innovazione. Resnick concettualizza il **processo di apprendimento creativo** come una spirale, un circolo virtuoso di gioco e divertimento che incoraggia l'innovazione e mira al costante miglioramento. Ci sono quindi **cinque fasi di apprendimento**:

- **Immagina**: nella prima fase, i bambini immaginano cosa costruire
- **Crea**: nella seconda fase, i bambini passano immediatamente alla parte pratica, iniziando a creare ciò che hanno immaginato
- **Gioca**: una volta costruito qualcosa, è il momento di giocarci
- **Condividi**: durante il gioco ciò che è stato creato viene condiviso con gli altri bambini
- **Rifletti**: giocando insieme agli altri, i bambini esplorano e riflettono sul proprio progetto, immaginando nuovi modi per migliorarli e da qui iniziano di nuovo a immaginare il nuovo progetto, tornando alla prima fase della spirale.

? Ti viene in mente una situazione in cui hai sperimentato il processo descritto da Resnick?

Approfondimenti

Mitch Resnick. "Come i bambini: IMMAGINA, CREA, GIOCA e CONDIVIDI. Coltivare la creatività con il Lifelong Kindergarten del MIT". Erickson, 2018.
Alcuni estratti: Progetti, Passione, Pari, Gioco.



4.4 Scaffolding

In pedagogia e psicologia, il termine *scaffolding* (impalcatura) si riferisce al supporto fornito da una persona per aiutare un'altra a imparare come svolgere un compito. Si tratta di una **metodologia didattica in cui l'insegnante offre livelli decrescenti di sostegno**, partendo da un supporto significativo e riducendolo progressivamente man mano che lo studente sviluppa nuove abilità o concetti. L'obiettivo dello scaffolding è alleggerire il carico cognitivo degli studenti, permettendo loro di diventare gradualmente più autonomi nel loro apprendimento.

All'inizio del processo, l'insegnante fornisce un ambiente fortemente guidato, in cui lo studente è attivamente coinvolto ma con possibilità limitate e un carico cognitivo ridotto. Con il progredire delle abilità dello studente, l'insegnante riduce il supporto, consentendo gradualmente un esercizio più autonomo, anche attraverso il lavoro di gruppo, fino a quando lo studente non è in grado di affrontare e risolvere problemi da solo, senza ulteriore assistenza.

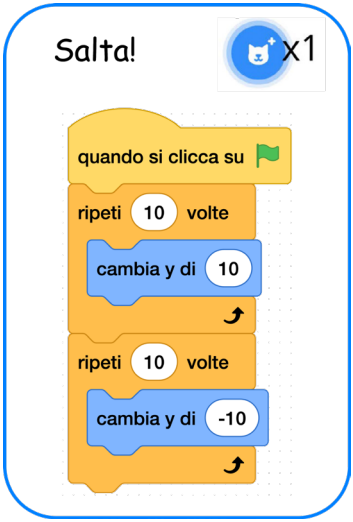
Nel caso dell'apprendimento del pensiero computazionale e della programmazione, è importante notare che il carico cognitivo derivante dall'apprendimento di un nuovo linguaggio e dalla risoluzione dei problemi può essere molto elevato. Spesso agli studenti viene chiesto di risolvere problemi complessi troppo presto, prima che abbiano pienamente compreso il linguaggio. È come chiedere a qualcuno di scrivere senza insegnargli prima come leggere.

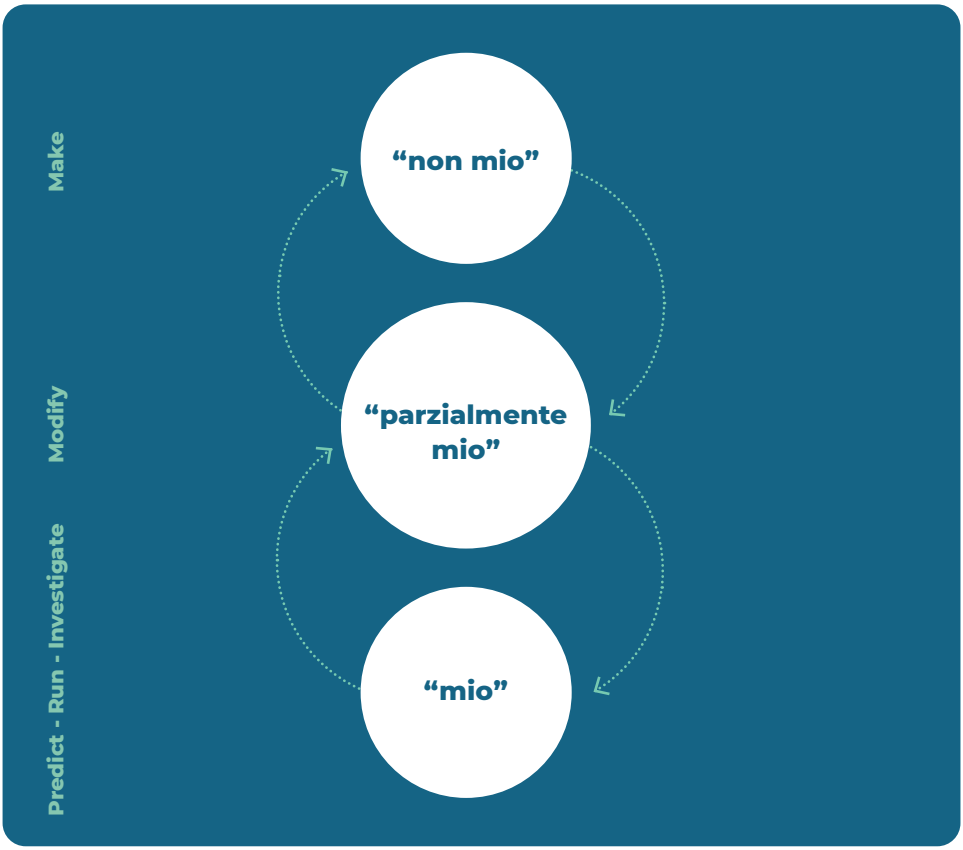
Partendo da questa idea, sono state sviluppate due metodologie per l'insegnamento della programmazione: **Use-Modify-Create (UMC)** e **Predict-Run-Investigate-Modify-Make (PRIMM)**.

“non mio”		“mio”	
Usa	**Modifica**	**Crea**	**Test**
All’inizio, agli studenti viene presentato del codice da eseguire o un modello da esaminare, e agiscono semplicemente come consumatori. Analizzano programmi già scritti per comprenderne il funzionamento e lo scopo, ricevendo nel frattempo indicazioni sotto forma di domande o suggerimenti per facilitare le loro esplorazioni.	Una volta che gli studenti familiarizzano con il codice, vengono incoraggiati a modificarlo. Inizialmente, possono modifiche semplici come il numero di vite in un gioco o il nome di un personaggio. Man mano che si esercitano, possono passare a modifiche più significative, come l’aggiunta di nuove funzionalità.	Man mano che gli studenti affrontano compiti sempre più impegnativi, acquisiscono maggiore fiducia nella loro capacità di programmazione. **Lavorano su sfide di programmazione e completano i propri progetti**, utilizzando le competenze che hanno coltivato durante le fasi Usa e Modifica.	Analizza → Perfeziona

All’inizio, lo studente sente che il codice non è “**mio**”. Man mano che progrediscono attraverso le fasi di Modify e Create, passano attraverso le fasi di “**è parzialmente mio**” e “**è mio**”. In tutte le fasi, è importante assicurarsi che i progetti scelti abbiano un livello di sfida appropriato. Rimuovere il supporto troppo rapidamente può aumentare l’ansia, mentre fornire un compito troppo semplice può ridurre l’interesse.

Ad esempio, considera la scheda Scratch a lato, che consente a uno sprite di Scratch di “saltare” in alto. Durante la fase di **Usa**, gli studen-





ti possono copiarla e utilizzarla per comprendere il ruolo delle diverse istruzioni. Durante la fase di **Modifica**, possono cercare di cambiare i valori numerici per effettuare diversi tipi di salti.
Durante la fase di **Crea**, possono integrare questo comportamento nel proprio progetto.

Un modello più complesso è **PRIMM**, che include le seguenti fasi:

- **Prevedi**: gli studenti esaminano, a coppie o in piccoli gruppi, piccoli segmenti di codice e cercano di indovinare cosa faranno, discutendone insieme.
- **Esegui**: il codice viene eseguito per confrontare l’esecuzione effettiva con la previsione; eventuali differenze rappresentano un ottimo punto di partenza per discutere con l’intera classe e risolvere eventuali fraintendimenti.
- **Indaga**: vengono presentati segmenti di codice simili, per esempio contenenti errori, e agli studenti viene chiesto di apportare piccole modifiche.

- **Modifica**: viene chiesto di modificare il codice in modo più esteso, per esempio cambiandone lo scopo.

- **Crea**: alla fine, gli studenti creano un nuovo programma, prendendo in prestito pezzi di codice precedentemente visti, ma con obiettivi diversi.

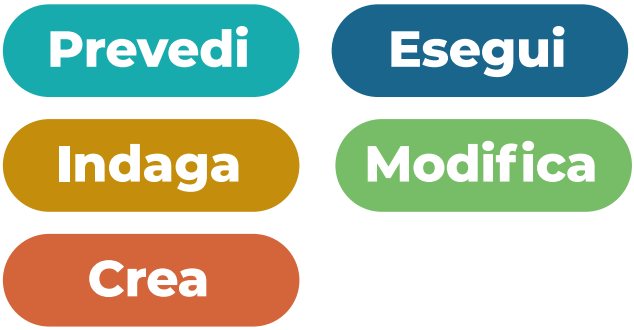
In entrambi i modelli, i singoli passaggi e il loro ordine non devono essere seguiti alla lettera. Per ciascun argomento, è possibile ripetere i passaggi intermedi secondo necessità prima di arrivare alla fase di creazione. Inoltre, alcune fasi possono essere omesse, sia perché non sono rilevanti per l'argomento trattato, sia per mancanza di tempo. È importante sottolineare, tuttavia, che tutte le fasi possono essere svolte attraverso il lavoro di gruppo. Avere l'opportunità di discutere con i compagni permette agli studenti di apprendere più rapidamente.

Il punto centrale di entrambi i modelli è l'idea di partire da un ambiente inizialmente limitato, che si espande gradualmente. In tutte le fasi è richiesto un atteggiamento attivo da parte degli studenti, che culmina nelle fasi finali dove diventa fondamentale stimolare la loro creatività.

C'era una volta in classe!

UMC e PRIMM richiedono un cambiamento radicale per quanto riguarda il coinvolgimento degli studenti, che devono diventare partecipanti attivi nel processo di apprendimento. Il nostro gruppo di ricerca ha condotto un esperimento didattico basato sul modello PRIMM. Uno degli insegnanti partecipanti ci ha raccontato che alcuni studenti, all'inizio, apparivano riluttanti a partecipare: durante le discussioni restavano passivi, limitandosi a chiedere all'insegnante le risposte corrette. Non è stato semplice, ma col tempo il loro atteggiamento è cambiato. Abituati a un approccio passivo e a istruzioni rigide, gli studenti hanno compreso che non c'erano risposte giuste o sbagliate e che non sarebbero stati valutati durante la discussione. Hanno così iniziato a cogliere l'opportunità di confrontarsi e migliorare insieme il proprio apprendimento.

Come spesso accade nella didattica, il passaggio dalla teoria alla pratica non è semplice. Negli esempi che seguiranno, utilizzeremo le etichette:



per evidenziare le opportunità di applicare questo modello nelle attività proposte. È interessante notare che questo approccio è compatibile con tutte le attività, incluse quelle che non richiedono l'uso del computer, come le attività unplugged.

? Secondo la tua esperienza, cos'è che fa capire che il livello di supporto fornito agli studenti è quello giusto?

? Durante lo svolgimento delle attività in classe, quanto spazio concedi per i "momenti creativi"?


5

Tinkering

Il termine *tinkering* deriva dalla parola inglese *tinker*, che indicava originariamente un artigiano ambulante specializzato nella riparazione di utensili. Con il tempo, tinkering ha assunto un significato più ampio, indicando qualsiasi tentativo sperimentale o improvvisato di migliorare o riparare qualcosa.

Di cosa si tratta?

Oggi il termine viene usato per descrivere l'atto di "smanettare" o fare piccoli aggiustamenti, spesso in modo informale e creativo, ed è diventato sinonimo di un **approccio sperimentale** all'apprendimento e alla risoluzione di problemi, che consiste nel provare diversi metodi per stimolare la creatività e scoprire soluzioni. **È un modo di esplorare** la scienza e l'ingegneria **attraverso attività pratiche e costruttive, che incoraggiano l'indagine, l'esplorazione e l'innovazione**. Il tinkering può avvenire in contesti diversi, dai laboratori specializzati alla propria casa, utilizzando materiali facilmente reperibili o addirittura riciclati. L'obiettivo è imparare facendo, sperimentando liberamente e coltivando curiosità e capacità di risolvere problemi.

Come iniziare?

Il sito dell'Exploratorium di San Francisco, un museo interattivo dedicato alla scienza, offre una sezione specifica per le attività di tinkering. Questa area include anche risorse educative, utili per replicare alcune delle attività in classe.

Perché è interessante?

Il tinkering è un approccio non tradizionale all'apprendimento che mette l'accento sul learning by doing: **aiuta** gli aspiranti creatori **a "pensare con le mani"** ed è un metodo educativo coinvolgente per avvicinare bambini e ragazzi alle discipline STEM in modo pratico e divertente.

Per chi è adatto?

Anche se spesso associato alle prime esperienze di apprendimento nella scuola primaria, l'approccio del tinkering è estremamente utile e stimolante per studenti di età più avanzata, incluse le scuole secondarie e persino i contesti universitari.

C'era una volta in classe!

Si potrebbe pensare che attività di tinkering siano adatte solo a studenti delle prime classi. Tuttavia, esistono numerose attività di tinkering che vengono svolte anche all'università. Un esempio è il *popsicle engineering* (ingegneria dei bastoncini da gelato), dove viene chiesto agli studenti di risolvere un problema ingegneristico (come costruire un ponte in grado di reggere pesi robusti) utilizzando solo bastoncini da gelato.

5.1 Un esempio divertente: lo scarabot!

1ª - 5ª elementare

www.neoconnessi.it/magazine/costruire-uno-scarabot-a-casa/

www.adrianoparracciani.it/doc/ScaraBot-Co-un-esperienza-di-tinkering.pdf

Un video, un'idea

How to Make a Scribbling Robot | Do Try This at Home | We The Curious [Youtube]

Gli ***Scarabot*** sono piccoli robot artigianali che, grazie a un motore elettrico e a semplici materiali di recupero, sono in grado di muoversi in modo irregolare e creativo, tracciando linee sul foglio o sul pavimento. Questo movimento caotico permette allo Scarabot di "disegnare" autonomamente, lasciando un tracciato che varia in base alla configurazione del motore e alla posizione dei pennarelli. Gli Scarabot sono un esempio divertente e pratico di tinkering, che avvicina i partecipanti a principi di robotica e meccanica in modo ludico, stimolando creatività e curiosità scientifica.

Materiali

- **Corpo del robot**: contenitori leggeri come bicchieri di carta, vasetti di yogurt o altri materiali di recupero.
- **Motore elettrico**: piccolo motore a corrente continua.
- **Batteria**: ad esempio, una pila stilo da 1,5V.
- **Propulsore**: oggetto da fissare all'asse del motore per generare movimento (ad esempio, un pezzo di gomma o plastica).
- **Pennarelli**: da fissare al corpo del robot per tracciare il percorso.
- **Materiali di fissaggio**: elastici, nastro adesivo, colla a caldo.
- **Fili elettrici**: per collegare la batteria al motore.

Organizzazione

- Procurati tutti i componenti necessari, preferibilmente materiali di recupero per promuovere la sostenibilità.
- Suddividi gli studenti in piccoli gruppi per favorire la collaborazione e lo scambio di idee.

Fasi di lavoro

1. Mostra ai ragazzi uno scarabot già montato, perché abbiano un'idea del prodotto finale.

Prevedi Esegui

2. In generale, per costruire uno Scarabot, è necessario fissare il motore elettrico al corpo del robot, collegarlo alla batteria tramite fili elettrici e montare un "propulsore" sull'asse rotante per generare movimento. Successivamente, i pennarelli vanno fissati alla base del robot in modo che tocchino il suolo, consentendo allo Scarabot di disegnare mentre si muove.

3. Incoraggia ogni gruppo a ideare il proprio design dello Scarabot, stimolando la creatività e l'innovazione.

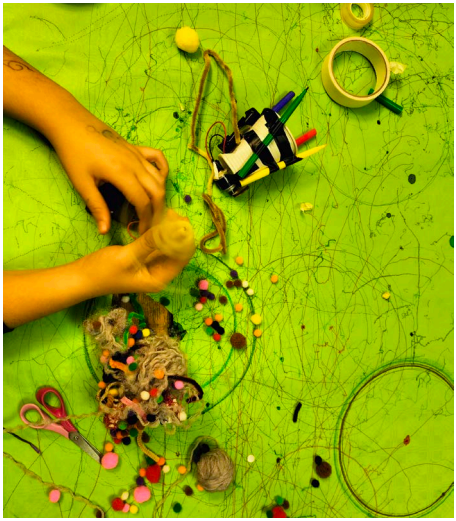
Crea

4. Una volta costruiti, i robot vanno testati; vedendo il lavoro degli altri, è facile che si scateni una competizione per chi costruisce il robot più buffo.

Modifica Indaga

5.2 Può esistere una ricetta per il tinkering?

Lo Scarabot è servito come esempio. Ma dare una serie di istruzioni precise per il tinkering non ha molto senso, poiché **questo approccio si basa sulla sperimentazione e sulla scoperta personale**. Piuttosto, forniamo suggerimenti su come gestire le attività di tinkering dal punto di vista pedagogico, offrendo supporto per creare un ambiente di apprendimento aperto e stimolante.



Inizia a piccoli passi:

introduci gradualmente le attività di tinkering e permetti agli studenti di abituarsi al processo prima di affrontare progetti più complessi. Inizia con attività semplici che richiedono materiali minimi, come la creazione di un circuito semplice con batterie e LED.

Innanzitutto, mostra degli esempi...

Nelle attività di tinkering, avere un progetto finito come esempio può essere molto utile. Può essere difficile per i ragazzi concepire ciò che devono costruire, quindi presentare un prototipo funzionante può essere vantaggioso.

Prevedi Esegui

Anche se l'attenzione è sull'immaginazione e la creatività, avere una comprensione concreta delle possibilità può rendere più agevoli le fasi iniziali del processo creativo.

...poi, utilizza stimoli aperti:

invece di fornire agli studenti un progetto specifico da completare, offri loro uno stimolo aperto che promuova l'esplorazione e la creatività. Ad esempio, chiedi loro di progettare una macchina che risolva un determinato problema, senza specificare l'aspetto della macchina o il suo funzionamento.

Modifica Crea

Incoraggia la collaborazione:

il tinkering può essere un'ottima opportunità per gli studenti di lavorare insieme e imparare gli uni dagli altri. Favorisci la collaborazione assegnando progetti di gruppo o offrendo opportunità agli studenti di condividere il loro lavoro e fornirsi feedback reciproco.

Poni l'accento sul processo anziché sul prodotto:

il tinkering si concentra sul processo di esplorazione e sperimentazione, piuttosto che sul prodotto finale. Sottolinea l'importanza di imparare dagli errori e incoraggia gli studenti a iterare e migliorare continuamente i loro progetti.

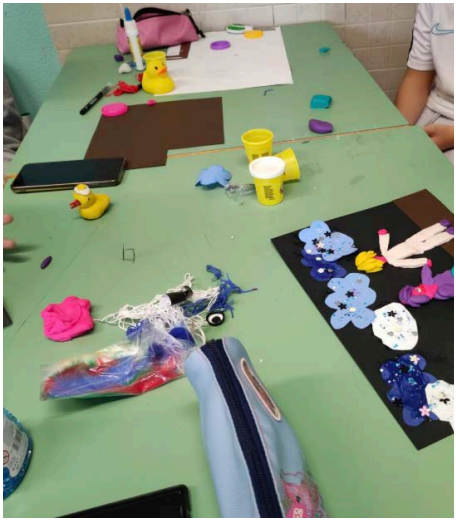
Crea

Fornisci una varietà di materiali:

il tinkering richiede una vasta gamma di materiali, quindi assicurati di offrire diverse opzioni che gli studenti possano utilizzare per creare i loro progetti. Gli studenti possono essere incoraggiati a partecipare al processo di ricerca dei materiali adatti, il che potrebbe comportare l'utilizzo di materiali riciclati, come tappi di bottiglia, cartone, rotoli di carta igienica e barattoli di plastica. Coinvolgendo gli studenti nella selezione dei materiali, vengono stimolati a pensare in modo creativo alle possibili applicazioni dei materiali, anche prima di iniziare il progetto.

Parti da problemi del mondo reale:

le attività di tinkering possono essere utilizzate per esplorare problemi e sfide del mondo reale. Collegando il tinkering a questioni importanti, gli studenti possono percepire la rilevanza di ciò che stanno imparando e sviluppare una comprensione più profonda della materia.



Incorpora riflessione e discussione:

dopo aver completato un'attività di tinkering, dedica del tempo affinché gli studenti possano riflettere sulla loro esperienza e discutere su ciò che hanno imparato. Questo può contribuire a consolidare il loro apprendimento e incoraggiarli a pensare criticamente sul processo e sui risultati ottenuti.

Attività unplugged

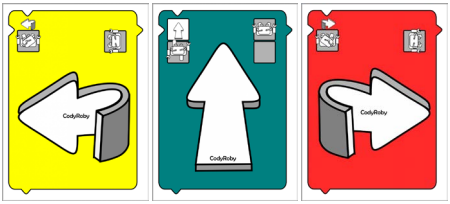
Imparare l'informatica senza computer, un approccio detto anche "*unplugged*" (senza fili), è uno dei modi più efficaci per introdurre gli studenti ai concetti di informatica e pensiero computazionale. Questo approccio è stato introdotto per la prima volta nella metà degli anni '90 da Tim Bell, professore di didattica dell'informatica presso l'Università di Canterbury in Nuova Zelanda.

Di cosa si tratta?

L'approccio di base consiste nel partire da un concetto informatico astratto, come la programmazione, l'ordinamento dei dati o la rappresentazione delle informazioni, e trasformarlo in un'attività concreta attraverso giochi o rompicapi. Queste attività prevedono l'uso di oggetti fisici che gli studenti possono toccare, manipolare e descrivere, facilitando così la comprensione del concetto sottostante. I materiali più comuni includono carta, cartone, pennarelli, materiali per costruzioni e gesso. In alcuni casi, vengono proposte anche attività fisiche (cinestetiche) per rendere l'esperienza ancora più coinvolgente.

Come iniziare?

Su Internet è disponibile una vasta gamma di attività unplugged, spesso accompagnate da descrizioni dettagliate degli obiettivi di apprendimento, dei concetti informatici alla base dell'attività e delle riflessioni per l'insegnante. Queste attività includono solitamente istruzioni passo-passo, materiali stampabili e, talvolta, video dimostrativi. Ad esempio, il concetto astratto di 'istruzioni' può essere illustrato utilizzando una carta da gioco che rappresenta visivamente una sequenza di istruzioni di movimento, come avviene con le carte CodyRoby.



Perché è interessante?

Partecipare ad attività unplugged consente agli studenti di esplorare concetti complessi e di porre domande su argomenti difficili da comprendere. Attraverso una rappresentazione fisica, gli studenti possono discutere l'attività usando esempi pratici, anziché limitarsi a ragionare su concetti astratti. Questo approccio è particolarmente utile per gli studenti con difficoltà di apprendimento, che trovano impegnativi i concetti

astratti e beneficiano di modalità di apprendimento diversificate. Le attività unplugged possono infatti integrare vari stimoli sensoriali, come il movimento fisico, la musica, la manipolazione di oggetti o il disegno.

Per chi è adatto?

Le attività unplugged possono essere utilizzate già dalla scuola primaria, ma alcune si rivelano utili anche con studenti più grandi, persino a livello universitario! Di seguito proponiamo alcuni esempi di attività senza computer adatte a diverse età e contesti.

6.1 Computer Science Unplugged

Il sito *Computer Science Unplugged*, creato da Tim Bell a metà degli anni '90, è considerato il capostipite di tutte le attività unplugged. Attualmente, sono disponibili due versioni:

- Classic CS Unplugged, un documento PDF unico, tradotto in numerose lingue e sviluppato fino al 2016, che include anche una Versione italiana tradotta da Renzo Davoli, Giovanni Michele Bianco e Piergiovanna Grossi.
- CS Unplugged, il sito web della versione moderna di CS Unplugged, che offre una gamma più ampia di materiali. Sebbene non sia disponibile in italiano, rappresenta comunque una risorsa preziosa per chi conosce l'inglese.

Kidsbot

1ª - 3ª elementare

www.csunplugged.org/en/topics/kidbots/rescue-mission/

Iniziamo con l'attività più classica e semplice, Kidsbot, che è un ottimo modo per introdurre concetti di base della programmazione come sequenza e ripetizione.

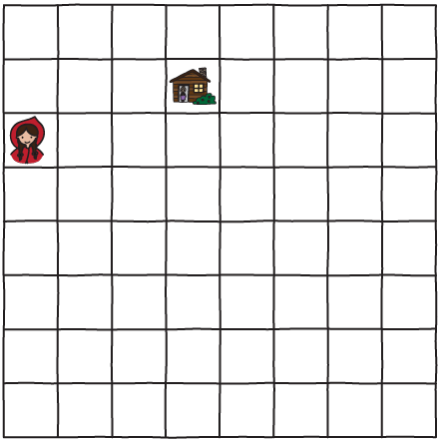
Un video, un'idea

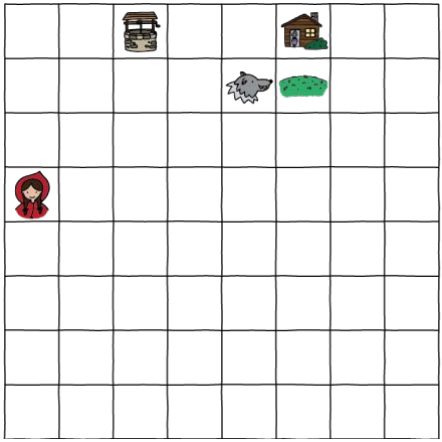
CodyRoby - Il nuovo kit e l'ora di codice unplugged [Youtube]

Materiali

Abbiamo bisogno:

- Una "**scacchiera**" - una griglia regolare di quadrati abbastanza grande da contenere i piedi di uno studente. Può essere creata sul pavimento usando lo scotch o nel cortile usando il gesso per disegnare per terra.





- Un insieme di istruzioni, che possono assumere la forma di un **set di carte**, come quelle di CodyRoby mostrate sopra, o più semplicemente comandi scritti sulla lavagna.

- Dei **segnaposto** per rappresentare punti di partenza e di arrivo o ostacoli; nelle griglie soprastanti vediamo Cappuccetto Rosso che deve raggiungere la casa della nonna.

Organizzazione

Ci sono due ruoli:

- al **programmatore** viene presentato un problema che deve risolvere creando un algoritmo che consiste in una sequenza di azioni di movimento;

- l'**esecutore**, noto anche come robot, deve quindi seguire le istruzioni passo dopo passo così come sono scritte.

Fasi di lavoro

1. All'inizio del gioco, è importante stabilire le regole. Per iniziare, puoi dare istruzioni "ambigue", per esempio "muoviti in avanti fino a quando non raggiungi la casa della nonna". Questo incoraggia gli studenti a fare domande e li spinge a pensare al ruolo della griglia. Attraverso la discussione e l'esplorazione, lavoreranno insieme per raggiungere un accordo sulle possibili istruzioni.

2. Successivamente, l'insegnante può fornire un insieme di istruzioni semplici, come "muoviti in avanti di una casella", "ruota a destra" e "ruota a sinistra". È importante notare che queste istruzioni avranno in seguito delle corrispondenze nei linguaggi di programmazione a blocchi, come Scratch, che prevedono concetti più complessi come gli angoli.

3. Prevedi Esegui A questo punto, l'insegnante può proporre alcuni programmi di esempio molto semplici e chiedere agli studenti di prevedere dove il robot arriverà. Successivamente, il robot cercherà di eseguire il programma e gli studenti potranno osservare se le loro previsioni erano corrette.

4. **Indaga** Il passo successivo è quello di proporre problemi più complessi, come ad esempio quelli che coinvolgono ostacoli, da risolvere da parte del programmatore e da testare con il robot. I ruoli degli studenti possono essere alternati. È anche possibile discutere il problema di trovare il percorso più breve.

5. **Crea** Infine, si potrebbe incoraggiarli a progettare problemi più complessi, come quello di un postino che deve uscire dall'ufficio postale, visitare diverse case evitando ostacoli come cani arrabbiati e buche nell'asfalto, e poi tornare all'ufficio.

L'idea informatica che ci sta dietro

Man mano che gli studenti acquisiscono esperienza con le attività unplugged, specialmente con problemi più complessi, potrebbero voler passare all'utilizzo di un linguaggio più avanzato, come ad esempio "muoviti in avanti di cinque caselle". Questo li aiuta a comprendere il concetto di ripetizione in modo più naturale.

È anche importante far notare agli studenti che ci sono possibilità aggiuntive oltre ai comandi di base. Ad esempio, i comandi "muovi in avanti/indietro" e "muovi a destra/sinistra" (senza angoli) hanno la stessa capacità espressiva. Ciò incoraggia la riflessione sul linguaggio come costrutto che facilita la comunicazione tra esseri umani o tra esseri umani e computer.

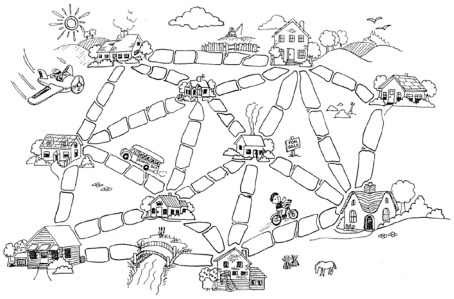
La città fangosa

4ª elementare
3ª media

classic.csunplugged.org/documents/books/italian/csunplugged-it.2015.1.0.pdf (pagina 111)

Tanto tempo fa c'era una città che non aveva strade. Gironzolare nella città era particolarmente difficile dopo un temporale perché tutt'intorno c'era solo fango: le macchine si impantanavano e le persone si sporcavano sempre gli stivali. Il sindaco della città decise così che qualche strada doveva essere pavimentata ma non voleva spendere più soldi del necessario perché voleva costruire anche una piscina. Il sindaco specifico allora due condizioni:

1. devono essere pavimentate abbastanza strade da rendere possibile per tutti andare dalla propria casa ad un'altra casa qualsiasi,
2. la pavimentazione deve costare il meno possibile.



Quella sopra è la mappa della città. Il numero di tavole (pietre) tra le case rappresenta il costo per pavimentare quel tratto. Trova il miglior percorso per connettere tutte le case, usan-

do il minor numero di tavole (pietre) possibile (il ponte non conta, non ha necessità di essere pavimentato). Quale strategia puoi usare per risolvere il problema?

Materiali

Abbiamo bisogno:

- Copie stampate della mappa
- Matite e gomme (per sbagliare e correggere!)

Organizzazione

Gli studenti vengono divisi in piccoli gruppi di 2-3 persone. Potrebbe essere utile ricordare loro che si tratta di un problema di ottimizzazione in cui devono trovare la soluzione che richiede il minor numero di pietre tra tutte le soluzioni possibili, garantendo che siano pavimentate abbastanza strade per consentire a tutti di spostarsi da casa propria a qualsiasi altra casa solo lungo strade pavimentate.

Fasi di lavoro

1. Gli studenti vengono invitati a risolvere il problema, utilizzando la matita e la gomma, tentando e riprovando. Esegui Indaga

2. Agli studenti non viene comunicato il numero minimo di pietre richieste, quindi quando trovano una soluzione, non sono sicuri se sia corretta o meno. È comune che gli studenti chiedano agli altri gruppi quale numero hanno ottenuto. Lasciali interagire!

3. Alla fine, uno dei gruppi potrebbe ottenere il valore minimo, magari per caso, e presto tutti i gruppi lo verranno a sapere. Assicurati che si scambino solo il numero, non l'intera soluzione. Ciò suscita una discussione tra gli altri gruppi: quale strategia ha utilizzato il gruppo con il valore più piccolo? Cosa abbiamo fatto di sbagliato? Come possiamo correggere le nostre soluzioni?

4. Fornisci agli studenti più copie del problema e consenti loro di modificare il numero di pietre tra le case rompendone alcune (disegnando) in due parti. Possono anche aggiungere connessioni aggiuntive con un numero significativamente grande di pietre. Successivamente, chiedi loro di risolvere il nuovo problema. Modifica

5. Incoraggia gli studenti a discutere e riflettere sulle loro strategie. Non si tratta solo di trovare la soluzione corretta, ma anche di identificare una strategia che possa funzionare in modo coerente. Spiegare come funziona questa strategia agli altri è un passo importante nel ragionamento sulle capacità di risoluzione dei problemi.

6. Sfida gli studenti a disegnare città completamente nuove e scambiarle con altri gruppi. La sfida potrebbe essere: creare città estremamente difficili da risolvere. Crea

Il costruzionismo in pratica

- **Chiediamo a noi stessi quanto chiediamo loro**.
  È il tuo turno! Prova a risolvere il problema. Modifica il problema per renderlo diverso. Sviluppa una strategia. Crea nuove città. Divertiti!

- **Prenditi il tuo tempo - il tempo necessario.**
  Questo è in realtà un problema molto complesso, risolto da due informatici, Kruskal e Prim, negli anni '60. Sebbene scrivere un algoritmo possa essere difficile, sviluppare una strategia è più facile, ma richiede comunque tempo. È importante concedere loro il tempo per pensare e discutere delle loro proposte.

- **Non puoi avere successo senza sbagliare.**
  Potresti trovare la soluzione corretta per caso, ma non imparerai molto da essa. I gruppi che scoprono che la loro soluzione non è corretta hanno una grande opportunità di apprendimento: avranno la possibilità di "debuggare" la loro strategia finché non otterranno il risultato giusto!

L'idea informatica che ci sta dietro

Il problema descritto qui è chiamato "Minimum Spanning Tree" (Albero di copertura minimo). È un problema importante utilizzato, fra l'altro, per progettare circuiti elettronici. La strategia utilizzata per risolverlo è semplice: selezionare le strade in ordine crescente di costo, garantendo al contempo che non vengano creati cicli. Dovrebbe sempre esserci un solo percorso da una casa all'altra. Se ci sono due percorsi diversi, allora esiste un ciclo. Se viene trovato un ciclo, una delle strade può essere rimossa e le case rimarranno connesse.

6.2 Bebras

Bebras è un'iniziativa internazionale che mira a promuovere l'informatica e il pensiero computazionale tra gli studenti di tutte le età. I partecipanti sono generalmente supervisionati dagli insegnanti che possono integrare la sfida di Bebras nelle loro attività didattiche. La sfida viene svolta a scuola utilizzando computer o dispositivi mobili. In Italia, Bebras è promosso dal Gruppo Aladdin dell'Università Statale di Milano: *Bebras Italia*.

Segnali luminosi

Lo storico greco Polibio (circa 206-118 a.C.) descrisse un sistema di comunicazione basato su torri di segnalazione. Anche in Giappone, durante l'era degli shogunati (governi militari), fu costruito un sistema di torri per comunicare con segnali di fumo in caso di emergenza. Nella figura a lato, il cerchio rosso indica la torre principale dello shogunato e i cerchi grigi indicano le altre torri di segnalazione. Due torri sono collegate da una linea se sono visibili tra loro. Supponiamo, inoltre, che i responsabili di ogni torre accendano il fuoco esattamente un minuto dopo aver avvistato il segnale proveniente da un'altra torre. Quanto tempo ci vuole affinché un segnale partito dalla sede dello shogunato raggiunga le torri più remote dell'impero e tutti i fuochi siano accesi?

Materiali

Abbiamo bisogno di:

- Copie stampate delle mappe
- Matite e gomme

Organizzazione

Gli studenti vengono divisi in piccoli gruppi (2-3 persone). A ciascun gruppo viene consegnata una copia della mappa.

Fasi di lavoro

1. Chiedi agli studenti di risolvere il problema, utilizzando la matita e la gomma, tentando, sbagliando e riprovando.

Esegui Indaga

2. Chiedi loro di partire da una torre diversa, ad esempio da quelle in basso o in alto, e poi chiedi come cambia il tempo necessario rispetto al problema originale.

Indaga

3. Chiedi agli studenti di trovare il nodo più "centrale", ovvero il nodo che richiede meno tempo per accendere tutti i fuochi delle torri.

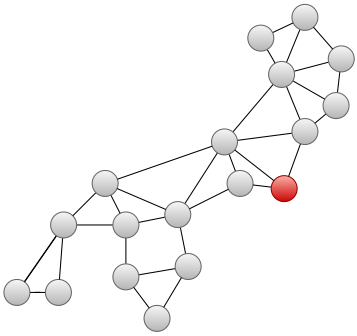
Indaga

4. Chiedi agli studenti di individuare altre situazioni in cui questo problema può essere applicato. Ad esempio, l'idea dei "sei gradi di separazione" si basa sullo stesso principio: si dice che tutte le persone siano collegate tra loro da sei o meno connessioni sociali.

Modifica

5. Chiedi agli studenti di disegnare altre mappe e chiedere loro di scambiarsi le mappe tra i gruppi. L'obiettivo è creare città che siano davvero difficili da risolvere.

Crea



L'idea informatica che ci sta dietro

Il problema descritto qui si chiama 'Breadth-First Search' (Ricerca in ampiezza); è un problema importante utilizzato per misurare la distanza su mappe (grafi) composte da oggetti (nodi) collegati da linee. La strategia utilizzata per risolverlo è molto semplice: si marca il nodo di origine con distanza 0. Quindi si marcano tutti i nodi direttamente collegati ad esso con distanza 1. Successivamente, si marcano tutti i nodi direttamente collegati ai nodi che hanno distanza 1 e non sono stati marcati con una distanza inferiore, con distanza 2. Si continua in questo modo finché tutti i nodi sono stati marcati. Il valore più grande è il tempo richiesto per raggiungere tutti i nodi, in questo caso 4.

Il pensiero computazionale in pratica

Spesso agli studenti viene chiesto di risolvere un problema e basta. Quello che stiamo suggerendo qui è di approfondire il problema provando diverse strategie, discutendo le strategie con gli altri, applicando la tua strategia a diverse versioni del problema e cercando di progettare un problema impegnativo per testare la tua strategia.

7

Scratch e apprendimento creativo

Scratch è un ambiente di programmazione gratuito progettato per consentire agli studenti di esprimere la propria creatività attraverso l'informatica. Utilizzando un linguaggio di programmazione visivo e intuitivo, Scratch permette agli studenti di creare giochi, animazioni e storie interattive, facilitando un primo approccio alla programmazione. Sviluppato dal gruppo *Lifelong Kindergarten* presso il MIT Media Lab, Scratch si basa sulla teoria costruzionista dell'apprendimento.

Di cosa si tratta?

Scratch si distingue come un **ambiente di programmazione accessibile e versatile**. Gli utenti programmano utilizzando blocchi colorati che rappresentano istruzioni, evitando la complessità della sintassi di codici testuali. Questo approccio aiuta i principianti a concentrarsi sulla logica e sulla progettazione di soluzioni, consentendo loro di creare facilmente una vasta gamma di progetti, dai giochi alle animazioni. La filosofia di Scratch si basa sulle "4 P": *Peers* (collaborazione con i pari), *Project* (orientamento al progetto), *Passion* (passione come motore dell'apprendimento) e *Play* (imparare attraverso il gioco).

Perché è interessante?

Il design user-friendly di Scratch lo rende adatto sia agli studenti che agli insegnanti e ai genitori. È uno strumento versatile che può essere utilizzato sia in ambito educativo che ricreativo, adattandosi facilmente a vari contesti e obiettivi. Scratch non è l'unico strumento disponibile per l'apprendimento della programmazione; alternative come Programma il Futuro offrono percorsi simili. Tuttavia, Scratch è spesso preferito per la sua combinazione unica di semplicità e potenzialità: gli studenti possono iniziare con progetti molto semplici e, col tempo, sviluppare competenze avanzate in modo graduale.

Come iniziare?

Iniziare con Scratch è facile e ci sono molte risorse online per imparare le basi. Oltre alla possibilità di partecipare ad un corso di aggiornamento, puoi partire con il nostro corso online, che vuole dare una semplice introduzione sulla programmazione in Scratch, oppure seguire corsi più lunghi come il corso Udemy su Scratch. Se invece preferisci imparare tramite un libro, ti consigliamo il testo di Maurizio Boscaini, Imparare a Programmare con Scratch.

Queste risorse sono pensate per insegnare a utilizzare Scratch, ma non offrono informazioni specifiche per gli insegnanti. In questo testo, invece, non vogliamo fornire un manuale completo, ma discutere gli aspetti didattici e pedagogici da considerare nell'uso di Scratch in classe, arricchendo il tutto con esempi e racconti di esperienze vissute.

Per chi è adatto?

Scratch è progettato per essere accessibile a un pubblico ampio, a partire dagli studenti della scuola primaria fino agli adulti che desiderano approcciarsi alla programmazione per la prima volta. È adatto per chiunque voglia esplorare il coding in modo creativo e pratico, senza necessità di competenze pregresse. Grazie alla sua flessibilità, Scratch può essere usato in classi di diversi livelli scolastici e, attraverso progetti sempre più complessi, continua a stimolare e sfidare anche gli utenti più avanzati.

7.1 Le quattro P dell'apprendimento creativo

Progetti (Project)

Attività come il tinkering e la programmazione attivano un processo creativo che facilita l'apprendimento, reso ancora più efficace quando è orientato verso la realizzazione di un progetto definito dagli studenti stessi. In questo contesto, **il progetto dovrebbe essere l'unità di lavoro principale**, sia per Scratch che per altre attività, permettendo agli studenti non solo di apprendere concetti di programmazione, ma anche di riflettere sul processo di trasformazione di un'idea vaga in un artefatto funzionante.

Questo approccio stimola gli studenti a sviluppare strategie creative per generare idee, testarle sul campo e migliorare i propri progetti grazie ai feedback degli altri. Sebbene risolvere

piccoli rompicapo aiuti a sviluppare il pensiero computazionale, lavorare su progetti porta un valore aggiunto: gli studenti diventano non solo "problem solver" (risolutori di problemi), ma anche "problem poser", capaci di definire i propri obiettivi. Dalla nostra esperienza come educatori, l'aspetto più affascinante è che non esistono due progetti uguali!

SUGGERIMENTI PRATICI

Quando insegniamo, siamo spesso costretti a seguire il famigerato "programma". È già difficile trovare il tempo per coprire tutto, come potremmo trovare tempo anche per l'apprendimento basato sui progetti? Ecco alcuni suggerimenti:

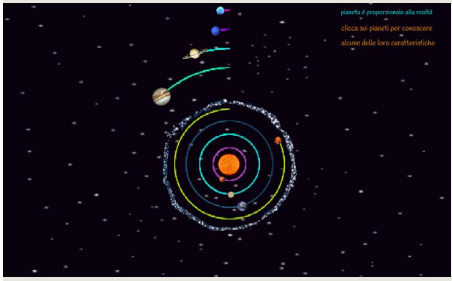
- **Integra le attività con la tua programmazione**: cerca progetti che si allineino ai tuoi contenuti e assicurati che gli obiettivi di apprendimento siano chiari sia per te che per i tuoi studenti. Puoi anche sviluppare tu stesso progetti che integrino i contenuti richiesti.

- **Pianifica in anticipo**: sviluppa un piano che definisca l'ambito del progetto, la sua durata e gli obiettivi di apprendimento.

- **Utilizza problemi autentici**: scegli progetti che si riferiscano a problemi del mondo reale o situazioni rilevanti per la vita degli studenti. Questo li aiuterà a vedere la rilevanza di ciò che stanno imparando e renderà il progetto più significativo.

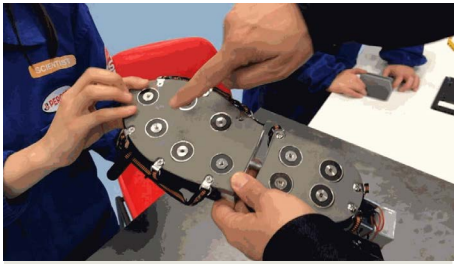
- **Rifletti sull'apprendimento**: alla fine del progetto, incoraggia gli studenti a riflettere su ciò che hanno imparato e su come si sono sviluppati come apprendisti. Questo li aiuterà a vedere il valore del progetto e a identificare aree di miglioramento per il futuro.

C'era una volta in classe!



Progetto Scratch

Camilla, una studentessa di 11 anni, ha creato con Scratch un progetto per illustrare il funzionamento del sistema solare ai suoi compagni di classe. Il progetto non solo rappresentava accuratamente l'organizzazione dei pianeti, ma simulava anche le orbite programmate di ciascun pianeta. Utilizzando le variabili, Camilla è riuscita a regolare le traiettorie in base alla distanza e alla velocità di rotazione, rendendo la simulazione realistica. Inoltre, ha inserito informazioni dettagliate su ogni pianeta, dimostrando una comprensione approfondita dell'argomento. Il progetto ha superato le aspettative, poiché Camilla ha cercato sul web e ha aggiunto curiosità su ciascuno degli otto pianeti del sistema solare.



C'era una volta in classe!

Nel 2019, il team di robotica MyCol-Lego della scuola di Bosco Chiesanuova (Verona) ha partecipato alla *First Lego League* con un progetto altamente innovativo: una suola progettata per rimanere ancorata al pavimento anche in assenza di gravità. Durante questo percorso, i membri del team hanno sviluppato competenze collaborative avanzate, lavorando attivamente con partner esterni per completare il progetto con successo.
Grazie alla loro creatività, il team ha condotto una serie di esperimenti, condividendo idee e ipotesi. La dedizione e l'impegno degli studenti, oltre alla buona volontà degli insegnanti, hanno permesso di instaurare una preziosa collaborazione con il Comando dell'Aeronautica Militare. Inoltre, il team ha contattato la fabbrica di scarpe Gaibana, un'azienda locale che fornisce calzature agli astronauti dell'Agenzia Spaziale Europea (ESA) per l'addestramento in grotta. Gli studenti hanno presentato la loro suola elettromagnetica Chaspy al proprietario dell'azienda, il quale, entusiasta del progetto, ha collaborato con loro alla realizzazione del primo prototipo.

Passione (Passion)

La motivazione svolge un ruolo fondamentale nell'apprendimento, e non è un segreto che impariamo più velocemente quando ci piace ciò che stiamo imparando. Implementando strategie che coinvolgono gli studenti in esperienze significative di apprendimento e permettendo loro di lavorare sulle proprie passioni, possiamo aiutarli a sviluppare l'amore per l'apprendimento. Resnick si basa sul concetto di **"Hard fun" (divertimento impegnativo)** di Papert, che sottolinea che il divertimento non significa sempre facilità. Infatti, affinché un'esperienza sia significativa, l'attività deve essere sia divertente che stimolante. Scratch è stato progettato con questi principi in mente, rendendolo uno strumento didattico efficace con tre caratteristiche fondamentali:

- **"Low Floor" (pavimento basso)**, il software è molto facile da approcciare e gli utenti possono ottenere risultati soddisfacenti con il minimo sforzo. Questo grazie, in parte, all'interfaccia grafica che elimina la complessità della sintassi di programmazione.

- **"High ceilings" (soffitto alto)**, una volta che gli utenti imparano a programmare, possono creare progetti molto complessi e aumentare gradualmente le proprie competenze. Questo offre agli utenti opportunità di sfidarsi e superare i propri limiti.

- **"Wide walls" (pareti ampie)**, è possibile utilizzare Scratch per sviluppare progetti di tipo diverso, partendo da elementi comuni. Ciò incoraggia

la creatività e la sperimentazione, che possono portare a creazioni uniche e innovative.

Introducendo attività di coding in classe e incoraggiando gli studenti a scegliere progetti che apprezzano, possiamo permettere loro di prendere il controllo del proprio apprendimento. Ciò stimola un amore duraturo per l'esplorazione e la scoperta, e aiuta gli studenti a sviluppare una passione per la risoluzione dei problemi e la creatività.

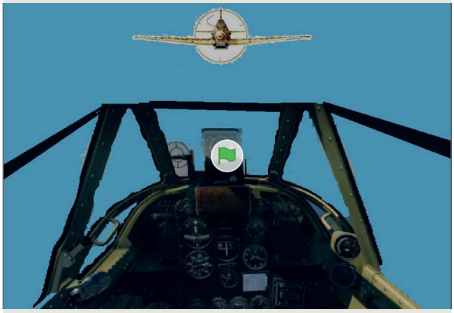
SUGGERIMENTI PRATICI

- **Lascia liberi gli studenti**: offri agli studenti la possibilità di scegliere ciò che vogliono imparare e come desiderano farlo. Fornisci opportunità per perseguire i loro interessi e passioni all'interno della tua programmazione. Ad esempio, se stai insegnando un'unità sulle antiche civiltà, lascia scegliere agli studenti quale civiltà vogliono studiare e presentare.

- **Offri opportunità per lo studio indipendente:** permetti agli studenti di perseguire i loro interessi attraverso lo studio indipendente. Fornisci risorse e orientamento, ma lascia che siano loro a prendere in mano il proprio apprendimento.

- **Utilizza esempi reali:** mostra agli studenti come i loro interessi e passioni possano essere applicati nel mondo reale. Utilizza esempi di persone che hanno seguito le loro passioni e ottenuto successo.

C'era una volta in classe!



Progetto Scratch

Matteo, uno studente di 10 anni, ha deciso di creare il videogioco "The War of Sky" nel tempo libero. Era appassionato di giochi di guerra e si è ispirato ai videogiochi che aveva visto in passato. Durante la creazione del gioco, ha imparato a utilizzare e gestire le variabili, a sincronizzare gli script su diversi livelli di gioco e ad applicare i principi della prospettiva per ottenere l'effetto di movimento dell'aereo in volo.

Pari (Peer)

L'apprendimento non è un'attività solitaria, ma sociale. I compagni di classe svolgono un ruolo fondamentale nella spirale dell'apprendimento creativo ponendo domande e fornendo risposte sui progetti realizzati.
In questa prospettiva, l'insegnante agisce come catalizzatore dell'attività, fungendo da consulente esterno per la realizzazione di un progetto e come connettore all'interno del gruppo dei pari per lo scambio di buone pratiche

e la condivisione di progetti.
Inoltre, Scratch ha una comunità con cui gli utenti possono condividere i propri progetti, esplorare, collaborare e remixare per trarre ispirazione dai progetti degli altri. I migliori progetti nascono dalla collaborazione con gli altri.

C'era una volta in classe!

Il "Coolest Project Award" è un concorso annuale organizzato da CoderDojo Verona, un club di coding gestito da volontari dove i ragazzi possono imparare a programmare. Il concorso prevede la votazione dei migliori progetti creati durante l'anno. Federica e Camilla hanno riconosciuto una necessità all'interno della comunità di CoderDojo e hanno creato un sistema di voto in tempo reale che consente agli utenti di votare per i migliori progetti.

Gioco (Play)

Ci sono vari tipi di giochi che possono essere sviluppati con Scratch, che incoraggia il gioco non strutturato, offrendo a tutti l'opportunità di esprimersi promuovendo un approccio ludico all'apprendimento. I blocchi di Scratch ricordano i mattoncini Lego nella forma e nel modo in cui si collegano tra loro. Il nome Lego deriva dalla combinazione di due parole danesi - "leg" e "godt" - che significano "gioca bene". In Danimarca, si usano due termini distinti per descrivere due tipi differenti di gioco: "spil" per il gioco strutturato basato su regole, e "lege" per il gioco libero, da cui i famosi mattoncini derivano il loro nome. Allo stesso modo, Scratch è stato progettato per fornire "mattoncini" di base per permettere il gioco libero e la creazione di vari programmi.

C'era una volta in classe!

A un gruppo di studenti delle scuole elementari è stato affidato un progetto in cui dovevano creare una storia usando Scratch. Sono state fornite loro le istruzioni di base su come utilizzare il software e i diversi blocchi a loro disposizione, ma sono stati incoraggiati a sperimentare e giocare con diverse combinazioni di blocchi per vedere cosa possono creare.
Sara era particolarmente interessata a creare una storia su una principessa e un drago. Ha iniziato creando diversi sprite per i personaggi e sfondi per le scene. Quando inizia a programmare la storia, si è resa conto che poteva utilizzare i diversi blocchi di movimento e rilevamento per creare più elementi interattivi. Ad esempio, ha aggiunto un blocco che fa ruggire il drago quando l'utente fa clic su

di esso e un blocco che fa agitare la mano alla principessa quando l'utente sposta il mouse su di lei.

Man mano che Sara continua a sperimentare e giocare con i diversi blocchi, la sua storia diventa sempre più coinvolgente e interattiva. Sara aggiunge anche un minigioco in cui l'utente deve aiutare la principessa a raccogliere gemme per sconfiggere il drago. Alla fine del progetto, Sara ha creato una storia interattiva completa che non è solo divertente, ma mette in mostra la sua creatività e capacità di risoluzione dei problemi. Attraverso questo progetto, Sara e i suoi compagni di classe hanno imparato che il gioco e la libera sperimentazione possono portare a creazioni davvero interessanti e innovative.

7.2 I quattro stadi dell'apprendimento creativo, in pratica!

Immagina! Mostra esempi per accendere l'immaginazione!

Una pagina bianca o uno schermo vuoto possono intimorire, specialmente per chi non è abituato ad avere libertà d'azione; troppa libertà di scelta spesso porta alla reazione: "Non so cosa fare!" Prima di iniziare le attività, può essere utile mostrare alcuni progetti completati per offrire ispirazione e far vedere ciò che è possibile realizzare, oltre a presentare progetti di base che mostrino come muovere i primi passi.

Prevedi Esegui

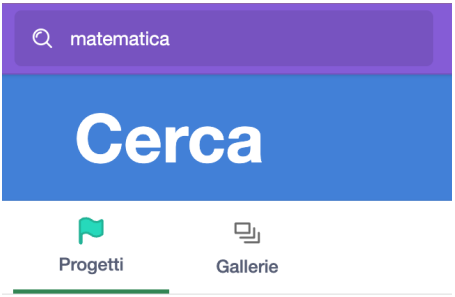
All'inizio, alcuni studenti potrebbero semplicemente copiare il progetto mostrato, e questo va benissimo! È importante, però, incoraggiarli a personalizzare e modificare gli esempi, aggiungendo il loro tocco personale.

Indaga

Ponendo domande come "Cosa succede se...?" oppure "Come lo faresti in modo diverso?", puoi stimolare la loro creatività e far sì che il progetto diventi davvero unico e personale.

Modifica Crea

SUGGERIMENTI PRATICI



Scratch non è solo un ambiente di sviluppo, ma anche una vivace comunità che offre risorse e ispirazione. Sul sito di Scratch, gli utenti possono esplorare una vasta gamma di progetti condivisi da altri membri utilizzando lo strumento di ricerca al centro della pagina. Oltre ai singoli progetti, è possibile accedere a gallerie tematiche cliccando sul pul-

sante *Gallerie*, che raccoglie progetti organizzati per argomento, offrendo così una visione completa dei lavori su temi specifici.
Ad esempio, se si vuole incoraggiare gli studenti a sviluppare un gioco a tema matematico, si può cercare una galleria con esempi pertinenti per mostrare ciò che altri studenti hanno creato. Una di queste gallerie è Math Game Everyone. Esplorando queste gallerie e progetti, gli utenti possono trarre ispirazione, imparare nuove tecniche e collaborare con altri nella comunità.

Quando inizi una nuova attività, fornisci esempi di progetti che hai già completato? Quanti? E come si diversificano?

Utilizzi strategie per aiutare gli studenti a stimolare la loro immaginazione? Quali?

Crea
Incoraggiali a "pasticciare"!

Maria Montessori affermava che "Le mani sono gli strumenti propri dell'intelligenza dell'uomo". Per lei, il pensiero creativo non si esaurisce nell'immaginazione, ma richiede anche un impegno pratico. Per aiutare gli studenti a generare idee, è fondamentale incoraggiarli a sperimentare e mettersi in gioco con attività concrete. Che si tratti di coding, Scratch o making, è importante chiarire fin dall'inizio che nella tua classe non solo è permesso, ma è incoraggiato esplorare, toccare e sperimentare, e che fare errori è non solo accettato, ma parte integrante dell'apprendimento. Dedica un momento della lezione a spiegare che gli errori sono una tappa cruciale nel processo di risoluzione dei problemi: provare e riprovare è fondamentale per sviluppare il pensiero creativo. Come ha detto Ken Robinson, esperto di educazione e creatività, "**Se non sei disposto a fare errori, non arriverai mai a creare qualcosa di originale.**"

SUGGERIMENTI PRATICI

Con Scratch, è possibile organizzare attività che incoraggiano l'esplorazione e la creatività, come per esempio: Utilizzo di materiali come le Scratch Cards per aiutare gli studenti nella fase iniziale dell'uso dei blocchi. Da qui, puoi suggerire agli studenti di modificare blocchi, personaggi, valori o combinare varie carte per creare progetti più complessi. Prevedi Esegui

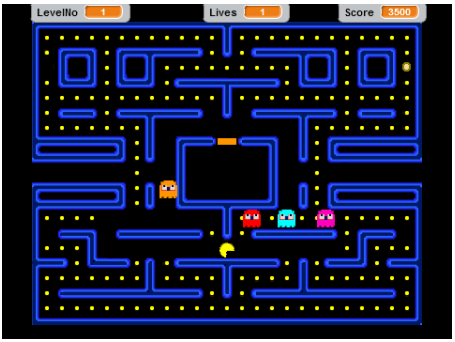
Introduzione di attività semi-strutturate come l'attività scratch a 10 blocchi, in cui si chiede agli studenti di creare qualcosa utilizzando solo 10 blocchi. Indaga

Puoi anche organizzare una Scratch Studio Challenge, in cui gli studenti lavorano in piccoli gruppi per creare una serie di progetti correlati all'interno di uno studio condiviso. Questo incoraggia la collaborazione e può portare a risultati interessanti e creativi.
Modifica Crea

Durante le attività in classe, incoraggi mai i tuoi studenti a pasticciare?

Quando qualcuno commette uno sbaglio, come affronti la situazione?

Gioca
Lasciali divertirsi!



Gli studenti hanno bisogno di tempo per lavorare su progetti creativi, specialmente quando esplorano continuamente nuove idee (come ci auguriamo). È essenziale dare loro spazio non solo per sperimentare e creare, ma anche per giocare con ciò che hanno realizzato. Giocare con le proprie creazioni è una fase importante del processo creativo, tanto quanto il realizzarle. Se, dopo una sessione di coding, gli studenti trascorrono un po' di tempo a giocare con un videogioco che hanno programmato, non è necessario che riprendano subito a programmare. Questo momento di gioco rientra nella spirale dell'apprendimento creativo, permettendo loro di riflettere sul progetto e immaginare nuove possibilità.

? Durante le attività in classe, ci sono dei momenti dedicati in cui gli studenti possano creare, esplorare e provare ciò che hanno realizzato? E lasci loro abbastanza tempo per farlo?

Condividi
Proponiti come mediatore!

Permetti sempre agli studenti di lavorare in gruppo e proponiti come mediatori. Lavorare insieme può essere più impegnativo rispetto al lavoro individuale, ma spesso porta a risultati migliori. Il motto di Scratch, "Immagina, Programma e CONDIVIDI", evidenzia l'importanza della collaborazione e della condivisione dei propri progetti con gli altri.

Quando si lavora in gruppo, è fondamentale stabilire un obiettivo comune e condiviso. Talvolta, gli studenti possono lavorare in coppia ma seguire le proprie idee senza vera collaborazione. Come facilitatori, è importante incoraggiare la comunicazione e il lavoro di squadra, aiutandoli a integrare le idee di ciascuno.

Inoltre, se gli studenti collaborano in modo efficace, sarà inevitabile che parlino e facciano confusione. Un gruppo che funziona bene può sembrare rumoroso, ma è un "rumore creativo" che alimenta la collaborazione e l'impegno. Come educatori, dobbiamo accettare e valorizzare questo tipo di dinamica, evitando di zittire gli studenti solo perché siamo abituati a una classe silenziosa.

SUGGERIMENTI PRATICI

Gli studenti non sono abituati a lavorare in gruppo; soprattutto quando si tratta di computer, sono talmente entusiasti dell'attività che vorrebbero avere sempre il controllo del mouse. Inizialmente, se gli studenti non sono in grado di gestire autonomamente i tempi di lavoro al PC, è utile prevedere

dei momenti per il "cambio di guardia"; i momenti possono essere brevi (ogni 20 minuti) o lunghi (metà lezione). Fornisci loro alcuni esempi concreti di come lavorare insieme:

- Pensare insieme al progetto, mettendo insieme le passioni di ciascuno.
- Se non si è d'accordo sulla storia da creare, ognuno sceglie un personaggio e decide aspetto, dialoghi ecc. purché sia in linea con la storia scelta in precedenza di comune accordo.
- Quando uno dei due usa il computer, l'altro partecipa comunque alla creazione della storia, non si fa gli affari suoi.

? Ogni quanto tempo fai lavorare gli studenti in piccoli gruppi?

? Adotti mai delle strategie per incoraggiare la collaborazione tra gli studenti?

? Quando in classe c'è rumore, come lo affronti?

INCORAGGIARE LA CONDIVISIONE CON TUTTA LA CLASSE

È fondamentale creare un ambiente sicuro e positivo, in cui gli studenti si sentano a proprio agio nel condividere le proprie abilità e nel chiedere aiuto. In qualità di facilitatori, potrete promuovere una mentalità di crescita, in cui gli errori vengono considerati opportunità per apprendere e migliorare. Sottolinea che ciascuno ha punti di forza e aree di miglioramento, e che lavorare insieme e sostenersi reciprocamente può condurre a risultati migliori.

Incoraggiate gli studenti a condividere i loro progetti con altri, sia di persona sia online, e a dare e ricevere feedback. Questo può avvenire attraverso revisioni tra pari o discussioni di gruppo, dove gli studenti possono evidenziare ciò che apprezzano nei progetti degli altri e suggerire possibili miglioramenti. Spunti di discussione come "Qual è stata la tua parte preferita di questo progetto?" possono facilitare un confronto costruttivo e positivo.

Promuovere un ambiente collaborativo e di supporto può aiutare gli studenti a superare la paura di condividere, stimolando una maggiore creatività e un apprendimento più profondo.

SUGGERIMENTI PRATICI

Per incoraggiare la condivisione con tutta la classe, è importante adottare le seguenti misure:

- Specifica fin dall'inizio che è consentito copiare, ma non durante le prove.
- Incoraggia gli studenti ad alzarsi e a dare un'occhiata ai lavori degli altri, per vedere cosa hanno creato i compagni e trarre ispirazione.
- Valorizza gli studenti che hanno completato le attività permettendo agli altri di copiare il loro lavoro e aiuta quelli che hanno difficoltà facendo loro copiare il lavoro degli altri.

Ricorda che gestire una classe in cui ogni gruppo lavora su un proprio progetto può essere impegnativo, quindi è importante avere alleati disposti ad aiutare in aula con voi. Gli studenti stessi possono essere i vostri migliori alleati, quindi non esitate a chiedere agli studenti migliori di aiutarvi a spiegare i concetti ai loro compagni.

? Adotti mai delle strategie per incoraggiare la condivisione tra gli studenti?

Rifletti
Condividi i progetti e le riflessioni con tutti

Al termine delle attività, è importante organizzare un momento specifico per presentare i progetti alla classe. In questo modo, potrai favorire il confronto tra tutti i progetti e una riflessione di gruppo guidata da voi. Da qui, ognuno può trarre ispirazione per migliorare il proprio progetto e immaginarne la continuazione.
Ascoltando le vostre riflessioni, gli studenti saranno più aperti a riflettere sul proprio pensiero e avranno un modello migliore per farlo. Pensa a loro come ad apprendisti del pensiero creativo e aiutali a diventare pensatori creativi mostrandogli i pensieri dietro le tue azioni..

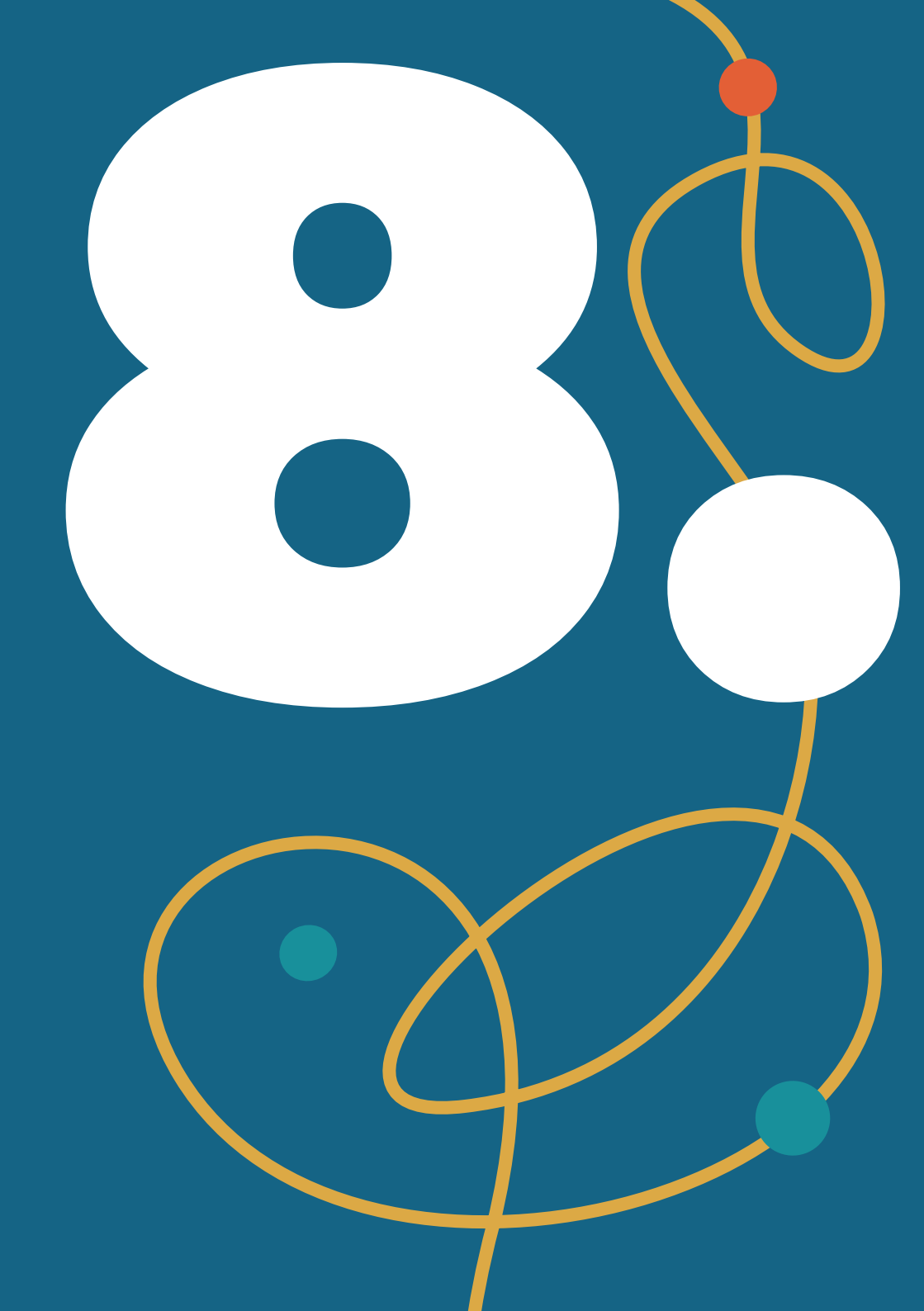
SUGGERIMENTI PRATICI

Incoraggia gli studenti a riflettere sul loro lavoro ponendo loro delle domande.

- Ad esempio, potrai chiedere: “Come ti è venuta l’idea per questo progetto?” per spingerli a pensare a ciò che li ha ispirati.
- Se li vedi in difficoltà, invece di chiedere se hanno bisogno di aiuto (diranno di no), chiedi “Cosa stai facendo?”. Questo spesso li porta a spiegarvi il loro progetto e li aiuta a identificare i punti in cui hanno difficoltà.
- Un’altra domanda da porre è: “Cosa dovrebbe fare il tuo personaggio?”. Questo li aiuta a riflettere sul loro progetto e a capire come risolvere i problemi senza che voi diate loro un input diretto.

Facendo domande, potrai aiutare gli studenti a diventare più consapevoli del loro processo creativo e incoraggiarli a pensare in modo critico al loro lavoro.

? Al termine delle attività che proponi agli studenti, prevedi mai un momento di revisione finale?



Integrare il pensiero computazionale

8.1 Verso un curriculum interdisciplinare

Alcuni insegnanti potrebbero temere che integrare il pensiero computazionale nell'insegnamento possa sottrarre spazio alla propria disciplina. Tuttavia, non è necessariamente così. La natura interdisciplinare del pensiero computazionale consente di esplorare connessioni tra scienze, matematica, tecnologia, ma anche discipline umanistiche, stimolando un pensiero critico e creativo.
Inserendo il pensiero computazionale all'interno delle lezioni e dei curricoli esistenti, gli insegnanti possono arricchire e rafforzare le discipline tradizionali, rendendo l'apprendimento più interdisciplinare e stimolante. Questa integrazione non solo prepara i giovani alle competenze necessarie per il mondo del lavoro e per una cittadinanza digitale responsabile, ma promuove anche un ambiente educativo inclusivo, in cui ogni studente è stimolato a pensare, collaborare e innovare.
Alcune applicazioni pratiche nelle varie discipline:

- **Matematica**: gli insegnanti possono integrare esercizi di programmazione per aiutare gli studenti a visualizzare e manipolare concetti come angoli e coordinate, rafforzando così la loro comprensione dei concetti matematici tradizionali.

- **Competenze linguistiche**: gli studenti possono utilizzare strumenti come Scratch per creare storie articolate, con più personaggi e narrazioni complesse, introducendo anche elementi di interattività. Questo approccio non solo stimola la creatività, ma incoraggia gli studenti a pensare in modo creativo e strutturato, arricchendo il loro uso della lingua e potenziando le abilità narrative.

- **Discipline umanistiche (storia, geografia, etc.)**: queste materie offrono ampie possibilità di applicare il pensiero computazionale invertendo i ruoli tradizionali di valutazione. Anziché valutare gli studenti tramite test a risposta multipla, si può chiedere loro di sviluppare quiz, sfidando i propri compagni e imparando nel processo. Una versione entusiasmante di questa attività è l'escape room. Anche qui si può invertire il ruolo: invece di proporre un'escape room già esistente, gli studenti trarranno maggior beneficio progettando la propria esca-

pe room, un processo che richiede approfondimento della materia e abilità di problem-solving.

- **Scienze**: gli insegnanti possono utilizzare modelli e simulazioni computazionali per facilitare la comprensione di fenomeni scientifici complessi. Gli studenti possono sperimentare con le variabili e osservare i risultati, sviluppando così una comprensione più profonda dei principi scientifici e imparando a fare ipotesi e interpretare i dati.

Integrare il pensiero computazionale nei programmi scolastici significa fornire agli studenti strumenti per comprendere, strutturare e risolvere problemi in modo innovativo e collaborativo. In quest'ottica, tutti gli strumenti e i metodi proposti – come le attività di programmazione, i laboratori unplugged, e il tinkering – possono essere utilizzati in modo scalabile nei diversi gradi scolastici, offrendo così un percorso verticale di apprendimento. Questo approccio aiuta gli studenti a sviluppare progressivamente la capacità di analizzare e scomporre problemi complessi e applicare soluzioni creative in vari contesti disciplinari.

8.2 Alcuni spunti disciplinari

Matematica

Angoli

Scratch dispone di due insiemi di comandi che possono essere utilizzati per disegnare figure matematiche. Il primo è il set Movimento, che permette agli sprite di muoversi sul piano usando comandi come "vai alla coordinata x y" (posizione assoluta), "muovi d passi" (muovi in avanti), "fai -d passi" (muovi indietro), "ruota a sinistra di d gradi" e "ruota a destra di d gradi". Il secondo è il set Penna, che deve essere aggiunto come estensione. La penna ha due comandi principali: "penna giù" (per scrivere mentre si muove) e "penna su" (per smettere di scrivere mentre si muove).
Dopo averli abituati a questi comandi, si può chiedere loro di disegnare figure geometriche.

Storytelling

Harry Potter Story Game!!!

Scratch è un ottimo strumento per la narrazione, che permette agli utenti di creare facilmente storie interattive. La piattaforma offre una vasta gamma di funzioni che facilitano l'aggiunta di personaggi, sfondi ed effetti sonori a una storia, consentendo anche agli utenti di creare grafiche e suoni personalizzati. Scratch include inoltre una serie di blocchi di programmazione che permettono agli utenti di aggiungere interattività alle loro storie, come attivare eventi in base all'input dell'utente o modificare l'esito della storia in base alle scelte fatte dall'utente. Con Scratch,

la narrazione diventa un'esperienza collaborativa, con gli utenti in grado di condividere le loro creazioni con gli altri e ricevere feedback e suggerimenti per miglioramenti.
Ecco alcune proposte che puoi fare ai tuoi studenti per creare le loro storie:

- **Storia animata**: crea una storia animata in cui i personaggi si muovono e interagiscono tra loro. Puoi utilizzare l'editor di sprite di Scratch per progettare i tuoi personaggi e sfondi, e quindi utilizzare il codice a blocchi per animare i loro movimenti.

- **Storia interattiva**: crea una storia interattiva in cui gli utenti possono prendere decisioni che influenzano l'esito della storia. Ad esempio, puoi creare una storia in cui l'utente deve decidere quale percorso il protagonista dovrebbe prendere, e la storia cambia in base alle loro scelte.

- **Gioco di avventura a scelta multipla**: crea un gioco in cui l'utente può scegliere la propria avventura. Ad esempio, puoi creare un gioco in cui l'utente è un avventuriero che esplora una giungla, e deve scegliere quale percorso prendere e quali oggetti portare con sé.

- **Storia collaborativa**: crea una storia collaborativa in cui più utenti possono contribuire. Ad esempio, puoi iniziare una storia con una scena iniziale, e poi invitare altri utenti a aggiungere alla storia creando nuove scene o personaggi.

- **Concorso di narrazione**: organizza un concorso di narrazione in cui gli utenti possono inviare le proprie storie create con Scratch. Puoi fornire un tema o una proposta per ispirare le loro storie, e poi avere una giuria che sceglie le migliori.

Approfondimenti

Le figure di *Propp*, anche conosciute come la Morfologia della Fiaba, sono un insieme di archetipi di personaggi che appaiono in molte fiabe tradizionali. Questi archetipi includono l'eroe, il cattivo, il donatore, il mandante e altri ancora. Integrare queste figure in Scratch può essere un modo divertente e coinvolgente per promuovere la narrazione.

Arte

Earth Day

Scratch può essere utilizzato per sviluppare progetti artistici che includono grafica, disegni, musica, canzoni e molto altro, consentendo agli utenti di creare e programmare grafica e animazioni personalizzate, oppure importare file multimediali esistenti da incorporare nei loro progetti. Scratch dispone anche di un editor audio integrato che consente di creare musica personalizzata e effetti sonori da utilizzare nei propri progetti. Inoltre, la comunità di Scratch consente agli utenti di condividere i propri progetti artistici con altre persone, ricevendo feedback e collaborando con gli altri su nuovi progetti. Complessivamente, Scratch fornisce una piattaforma potente e versatile per gli artisti di tutte le età per esprimere la loro creatività

ed esplorare nuovi mezzi artistici in modo divertente e coinvolgente.

Scienze

Sistema solare

Scratch è uno strumento fantastico per sviluppare progetti scientifici che includono simulazioni e visualizzazioni. Con le sue potenti capacità di programmazione e l'interfaccia user-friendly, Scratch consente agli utenti di creare simulazioni interattive che illustrano concetti scientifici, fenomeni ed esperimenti. Che si tratti di una simulazione di un sistema solare, di un ecosistema o di una reazione chimica, Scratch permette agli utenti di rendere vivi complessi concetti scientifici in modo divertente e coinvolgente.

8.3 Una galleria di esempi

Guarda questi progetti, sviluppati da ragazzi di tutto il mondo. Il loro scopo è ispirarti e fornirti esempi di come i contenuti educativi possano essere portati su Scratch. Una volta aperto un progetto, cliccando sul pulsante "See Inside" in alto a destra, è possibile visualizzare lo script con cui è stato creato il programma e, cliccando sul pulsante verde "Remix", è possibile creare una copia del progetto per modificarlo a proprio piacimento!

Materia	Descrizione
Italiano	Creazione di un libro digitale (Illustrazioni, suoni e didascalie realizzate dagli studenti seguendo un racconto)
Italiano	Narrazione a fumetti di un racconto di Gina Labriola "Il pianeta Faidatè" adattato e illustrato per il libro di testo "I colori dell'arcobaleno 2" ed. Il Capitello
Storia d'arte	Animazione di un quadro di Van Gogh
Storia d'arte	Omaggio a Matisse
Geografia	Presentazione di una regione d'Italia: il Trentino Alto Adige
Geografia	Mappa interattiva di Rimini
Storia	La mummificazione degli Egizi
Inglese	Breve dialogo in lingua inglese
Matematica	Disegnare poligoni regolari
Matematica	Angoli

Progetti più complessi, che richiedono blocchi di più categorie:

Materia	Descrizione
Matematica	Disegno di poligoni
Geografia	La deriva dei continenti
Geografia	La Toscana
Storia	Quiz di storia romana
Storia	Viaggio nel tempo nell'Antica Grecia
Italiano	Quiz sull'analisi logica
Italiano	Creazione di una storia a bivi
Matematica	Tabelline
Matematica	Conversione unità
Sostegno	Appare Tommaso - Scratch per usare in modo intelligente la LIM con un ragazzino autistico.

Gallerie di progetti educativi:

Materia	Descrizione
Progetti Educativi	Contiene molti esempi di Scratch for education.
Arte	Galleria di Arte Generativa.
Scienze	Galleria “Ripassare scienze”.
Matematica	Attività didattiche (in Inglese) dedicate alla matematica e la galleria di progetti associati.

? **Dopo aver dato un’occhiata ai progetti e alle gallerie, pensate alle vostre lezioni: ce ne sono alcune che possono essere proposte utilizzando Scratch?**

Le mie regole di Jennifer Orr

Il testo che segue è una sintesi di un capitolo del libro di Gary Steger, "20 Things to Do with a Computer, Forward 50": una lettera di un'insegnante che condivide le proprie regole, ma anche i propri dubbi e come li ha superati. Consigliamo vivamente di leggere l'intero libro!

Prima di tutto, una confessione: sono un'insegnante di primaria senza una formazione completa in informatica o programmazione. Mi piace mettermi in gioco, e ancora di più vedere i miei studenti fare lo stesso. Spesso sono più bravi di me, anche i più piccoli [...] Mi prendo comunque parte del merito: il mio punto di forza non sono le competenze informatiche, ma la capacità di dare spazio agli studenti. Nel tempo ho sviluppato alcune regole per me e altre per loro, che ritengo essenziali per creare questo spazio. Credo che queste regole — almeno quelle per me stessa — possano essere utili a tutti i docenti, non solo a quelli che insegnano informatica.

Regole per me stessa

1. Mostra agli studenti di non sapere

I bambini piccoli spesso credono che gli adulti sappiano tutto, e vedere quanto si fidano di quello che dici può essere esaltante. Mi ci sono voluti anni per capire che **mostrare agli studenti ciò che non so** non è una debolezza, ma un potente esempio. Se io, adulto e insegnante, non so tutto, allora va bene anche per loro non sapere. Questo li rende più liberi di correre rischi e fare domande.

2. Celebra le domande

Se voglio che i miei studenti facciano domande, devo incoraggiarli e fare attenzione a non scoraggiarli inconsapevolmente. Fare una domanda comporta un rischio: si potrebbe sembrare ingenui o poco preparati. Anche i più giovani imparano presto, spesso proprio a scuola, che dovrebbero sapere le risposte invece di fare domande. **Rispondere con entusiasmo alle loro curiosità** li incoraggia a fare altre domande. Ancora meglio è girare le domande alla classe: chi ha chiesto percepisce il valore della domanda, e tutti sono coinvolti nel trovare una risposta.

3. Evidenzia i tuoi errori

Nei miei primi anni di insegnamento cercavo di nascondere o minimizzare i miei errori. Oggi, invece, ho capito che è importante **mettere in evidenza i propri errori**, mostrando loro come correggerli. Vedere un adulto sbagliare e capire che, il più delle volte, non è un problema, dà agli studenti la libertà di fare errori. Senza errori, non si impara: per questo questa regola è fondamentale.

4. Celebra gli errori
Oltre a sottolineare i propri, è importante **celebrare i loro errori**. Gli studenti sbagliano perché stanno esplorando oltre ciò che è noto e confortevole, e questo coraggio merita di essere riconosciuto. Inoltre, molte risposte errate contengono comunque elementi positivi, che, se evidenziati, incoraggiano gli studenti a costruire su questi punti di forza e ad ampliare le proprie conoscenze.

5. Gioca
(forse questa dovrebbe essere la prima regola)
È facile, per un docente, finire intrappolati nel programma, negli obiettivi di apprendimento, nei piani di apprendimento individuali, nelle griglie di valutazione, e qualunque altra cosa il ministero si inventa, perdendo di vista gli studenti di fronte a noi e le loro capacità. **Dare loro il tempo di giocare con gli strumenti che gli fornisco** è ormai una voce fissa della mia programmazione. Non me ne pento mai. Le cose che gli studenti apprendono attraverso le loro libere esplorazioni sono spesso molto più complesse e difficili di quanto potrei insegnare loro. Dando loro lo spazio per giocare con quanto appreso, gli studenti acquisiscono una nuova capacità di apprendere.

6. Rimuovi le barriere
Più riduco la possibilità di accedere agli strumenti, meno gli studenti apprendono. Per quanto riguarda la programmazione, questo significa che non metto prerequisiti per quanto riguarda gli strumenti che offriamo. Che si tratti di BeeBot, di Scratch, dei Microbit: se uno studente vuole provare, lo lascio. Prima di iniziare, durante le ricreazioni al chiuso, o durante qualunque momento libero, questi strumenti sono sempre a loro disposizione. Quando noi adulti, figure autorevoli per loro, erigiamo barriere all'apprendimento, stiamo allontanando gli studenti. L'obiettivo per me è **fornire gli strumenti ai ragazzi**, gli obiettivi per loro è di avere il tempo per prendere confidenza.

7. Impara dagli studenti
Quando gli studenti prendono confidenza, diventano esperti. Non appena li vedo fare qualcosa di nuovo, **chiedo loro di insegnarmi**. Dato che ho già mostrato loro quanto poco conosco, sono disponibili a darmi una mano. Questo ha anche il vantaggio ulteriore di spingerli a spiegare il loro processo, il che spesso porta a revisioni e miglioramenti.

8. Arruolali come insegnanti
Ogni volta che è possibile, **affido la cattedra ai ragazzi**. Mostro il loro lavoro al proiettore e glielo faccio descrivere. Per i loro pari, sono spesso migliori insegnanti di quanto non sia io. Si capiscono l'uno con l'altro e possono rispondere alle domande in modi che hanno senso per loro. Questo ha effetti positivi sia sugli studenti che ascoltano, sia sullo studente che insegna.

Regole per gli studenti

1. Alzati, guardati in giro, osserva cosa fanno gli altri studenti

Alcuni studenti si alzano senza problemi, se sentono un loro compagno fare un verso di stupore o di gioia. Altri studenti invece hanno imparato troppo profondamente che non ci si alza senza permesso. Io mi avvicino a quelli che si sono alzati, discuto con loro di quello che vedono, e mi entusiasmo con loro. Quindi incoraggio gli altri ad alzarsi e guardarsi in giro. Alcuni studenti sono troppo presi da quello che stanno facendo per alzarsi, e questo ovviamente non è un problema. Ma è importante che gli studenti sappiano che possono alzarsi e vedere cosa succede nella stanza.

2. Non si toccano gli strumenti dei compagni

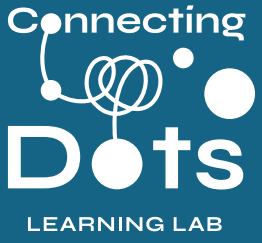
Questa regola deriva dall'osservare gli studenti seguire la regola numero 1, scoprire qualcosa di interessante e quindi chiedere al compagno come si fa. A questo punto, il compagno si reca al computer del richiedente e ripete il processo da solo. Nessuno beneficia da questo. Lo studente A, che ha scoperto qualcosa di nuovo e lo ha mostrato allo studente B, deve accompagnare lo studente A al suo computer e spiegargli il processo, mentre lo studente B esegue i passi così descritti. Questo forza lo studente A a spiegare cosa ha fatto, spingendolo a pensare al processo in maniera più profonda di quanto abbia fatto da solo, e forza lo studente B ad apprendere meglio il processo. Entrambi gli studenti finiscono ad avere competenze più approfondite grazie a questo scambio.

Testi:
Giorgia Bissoli
e Alberto Montresor

Ringraziamenti al Team ConnectingDots:
Francesca Fiore,
Agnese Del Zozzo,
Giulia Paludo,
Marta Valentini,
David Zikovitz

Progetto grafico:
Nadia Groff

Finito di stampare:
maggio 2025,
Trento


Connecting
Dots
LEARNING LAB